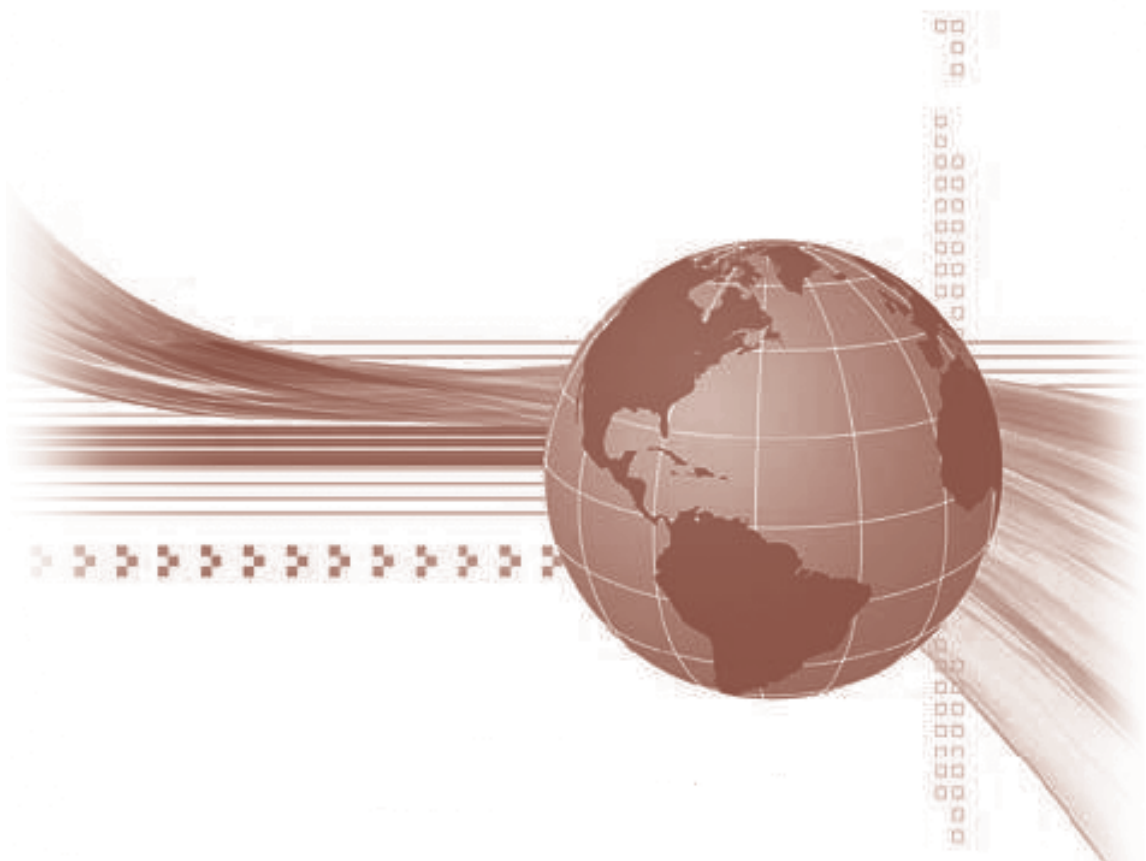
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

2/2015

STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 2/2015

July - December

EDITORIAL BOARD

YEAR **Volume 60 (LX) 2015**
MONTH **DECEMBER**
ISSUE **2**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

2

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

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

A NEW UNSUPERVISED LEARNING BASED APPROACH FOR GENDER DETECTION OF HUMAN ARCHAEOLOGICAL REMAINS

IOAN-GABRIEL MIRCEA, SERGIU-GEORGE LIMBOI, AND MARA-RENATA PETRUŞEL

Abstract. Detecting the gender of human skeletal remains is an important problem within archaeology, since it is essential for understanding the characteristics of past societies. We approach in this paper, from a machine learning perspective, the problem of sex identification of human skeletal remains from bone measurements. In order to partition a group of skeleton remains according to their gender, different clustering algorithms are considered. Computational experiments carried out on publicly available archaeological data sets show a good performance of the proposed clustering approaches with respect to existing similar approaches from the literature.

1. Introduction

Machine learning (ML) [20] is a challenging field of computational intelligence whose goal is to develop computational systems which are able to improve their performance through experience, by learning some specific domain knowledge.

We approach in this paper, from a machine learning perspective, the problem of sex identification of human skeletal remains from bone measurements. Many studies in the literature propose various approaches for detecting the gender of human skeletons, most of which are based on bone measurements, DNA and gene analysis or different statistical methods.

In this paper we aim at introducing a novel machine learning approach based on clustering for solving the sex detection problem. The case studies used in the experiments for evaluating the performance of our model show that

Received by the editors: March 31, 2015.

2010 *Mathematics Subject Classification.* 68T05,62H30.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]:Learning– *Induction*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *Medicine and science*.

Key words and phrases. bioarchaeology, sex determination, machine learning, clustering.

our approach overperforms the existing similar approaches from the literature. To the best of our knowledge, our approach is novel, since there are no similar approaches in the literature.

The rest of the paper is organized as follows. Section 2 presents the problem approached in this paper, highlighting the motivation of our work, as well as different approaches existing in the literature for gender detection of human skeletal remains. The fundamentals of clustering are given in Section 3. Section 4 introduces our novel unsupervised learning based approach for the determination of sex in human skeletons. Experimental evaluations are given in Section 5, while comparisons to related work from the literature are presented in Section 6. Section 7 contains the conclusions of the paper and indicates future research directions.

2. Gender detection of human skeletal remains

Identifying the gender of archaeological human remains is essential for studying the gender differences in past populations [8]. This contributes to a better understanding of the social position and attributions of each gender in society. The sex classification task is a very delicate one and is highly influenced by the historical period and the geographic origin of the skeleton.

Usually the skull and the pelvis measurements are used for determining the gender, but other bones from the body may be used as well. Thus, measurements of the arm and leg bones may be important in the sex detection process. Though, in the case of subadult individuals, sex identification based on bone measurements is not fully accurate.

2.1. **Related work.**

Barrier's idea [3] of elaborating a discriminant function formula for sex determination based on researching forearm bones: radius and ulna, came from the great number of South African skeletal remains. The accuracy obtained was between 76% and 86%, therefore these bones are moderate discriminators for the given assignment.

Wada [25] suggested a distinct discriminant function formula, using for sex determination the radius on 63 instances, both males and females. More precisely, there were 35 correctly classified instances for females (97.2%), respectively 25 for males (92.6%). In order to increase the accuracy of the results, the unclear area of discriminant was diminished. Better results did not fail to appear.

Traditional morphometric analysis cannot solve the sex determination problem for partial and immature skeletal remains. The bests results seem to be provided by a new method built on amplification of the single-copy amelogenin-encoding gene projected by [8]. From a total consisting of 22 skeletal remains, 18 were properly classified. The skeletal remains dated from

periods starting from 200 years to 8000 years ago and, among them, young children skeletons were also studied.

There exists another method that deals with sex determination for incomplete or immature skeletal remains suggested by [16]. This method uses preserved DNA and Polymerase Chain Reaction. The advantage of this technique is that the research can be done even on a single unit of DNA or on burnt or charred material.

Anthropologists accept two distinct methods for gender detection, particularly morphological, or non-metrical, and metrical, including geometric morphometrics. Besides these, molecular techniques concentrating on DNA research have been included. Although every single method is confined by a number of constraints [5], they complement each other and produce better results when implemented in conjunction with each other.

Weiss disagrees in [26] with a statistically important bias in sex determination affecting adult skeletons. The causative factor for the bias is (approximately 12%) in favor of male population. In consequence, it is stated the nature of the secondary sex features in the bone. The size of the skull, the femoral head size and the rugosity of the bone illustrate these features. So, when a specimen has medium size, rugosity or development is discovered, there is an inclination to classify it as male [26]. In support of this hypothesis comes the research on adult sex ratios of skeletal populations from various time periods, cultures and geographic areas and its importance illustrates the need of taking into consideration the bias when sex-specific analysis of skeletal material is tried.

Bruzek [7] proposed a visual technique for sex determination that studies only the os coxae, the bone thought to provide the best accuracy for this problem. The approach was based on a formula-based methodology as an alternative to the simple visual remarks. The described changes involve the minimization of observer subjectivity and a better probability of a correct diagnosis with separate fragments of the bone. Tests have been made on a sample of 402 adults of French and Portuguese origins. A correct sexual determination was acquired in 95% of cases.

Another method for gender detection is odontometrics, which was proposed by Vodanovic [24]. The practicability of this method was doubted by the significance of the teeth in cases of poor maintenance of skeletal remains and the better accuracy obtained by adding the odontometric parameters in the procedure that previously was using only craniofacial characteristics. Nevertheless, there was stated as a disadvantage of this method the lack of referent odontometric values for comparison.

A distinct technique for sex determination was suggested by Vanharová and Drozdová in [23]. Analysis of DNA was applied to skeletal remains of

children, having results consistent with archeological grave remains and body imposition. The molecular method is not restricted by physical fragmentation and can be applied on immature individuals, in case skeletally based identification is not possible. Although, the molecular method is restricted by some constraints including molecular contamination or molecular preservation.

Other techniques that use analysis of the bones have been attempted with success for skeletal gender detection. Measurements of the bones of the hands and feet (metatarsals,metacarpals, phalanges) were used in [22] and [9] with accuracy rates of 80% and 84-92%. Stature and gender were estimated by use of foot measurements in [27] with accuracies of 95.6% for the right foot measurements, respectively 96.4% for the left foot.

Several computational intelligence techniques have also been investigated for the detecting the sex of human skeletons. Neural networks have been applied in [4] to classify archaeological remains based on osteological measurements. Two archaeological databases were investigated and an accuracy ranging between 81-88 % was obtained. Afrianty et al have used in [1] back-propagation neural networks for gender detection on a data set consisting of remains characterized by pelvic and pattela bones measurements. Accuracies of 86.6%- 98.3% were obtained on a data set generated based on statistical values reported in the literature [2].

Stevenson et al have approached in [21] the sex prediction problem using CHAID (Chi-square automatic interaction detection). CHAID is a type of decision tree technique based on the Chi-square test [11] to determine the best next split at each node in the tree. Experiments are performed on 304 remains of American, European and African ancestry who died between 1915 and 1955 and accuracies between 85% and 85.5% were obtained.

3. Background

In this section we are providing a brief background on clustering , the unsupervised machine learning method which will be further used in our approach.

3.1. **Clustering.**

Unsupervised classification, or *clustering* is an important activity within data mining. The goal of clustering is to identify groups (classes or clusters) inside a given data set of instances (objects) [12], and it is considered the most important unsupervised learning problem. The resulting groups of instances are formed such that the instances within each cluster are more similar to one another than the instances belonging to different clusters. An important notion in the clustering process is the *similarity* (or *dissimilarity*)

between the instances. The measure used for expressing the dissimilarity between the instances can be any metric or semi-metric function (e.g. *Euclidian distance*, *Manhattan distance*, *Minkovski distance*, etc).

Most clustering algorithms available in the literature [12], [14]are based on two techniques known as *partitional* and *hierarchical* clustering.

In the following, we present a brief overview of the partitional clustering methods which we will use for gender detection.

Given n instances and a number k, a partitioning technique splits the instance set into k distinct and non-empty groups (clusters). The partitioning process is iterative and it stops when a "good" partitioning is obtained. The partitional clustering algorithms try to minimize a criteria (usually defined as a squared error function) and generally they converge to local optima [14].

The *k-means* algorithm divides a set of n objects into k distinct and non-empty clusters[14]. The algorithm starts with k initial centroids, then the clusters and their centroids are iteratively recalculated (each instance is put into the closest centroid) until convergence is reached.

In the *k-medoids* or *PAM* (Partitioning around medoids) clustering algorithm [15], each cluster is characterized by one of the instances within the cluster, and this representative instance is called *medoid*. The *k-medoid* algorithm starts with k initial medoids for the clusters, then the clusters and their medoids are iteratively recalculated (each instance is put into the closest medoid) until convergence is reached. At a given iteration, a medoid of a cluster is replaced with another object from the cluster, if it reduces the total distance of the obtained clustering [15].

Certainly, the initial medoids or centroids impact the performance of the PAM and $k-means$ algorithms. Thus, there is no guarantee for obtaining an optimal solution. Another drawback of the algorithms is that the number of clusters have to be initially specified.

4. Our approach

This section presents our unsupervised learning approach using *clustering* for determining the gender of human skeletons from the length of long bones.

4.1. **Model.** We are considering a data set $\mathcal{AR} = \{ar_1, ar_2, \ldots, ar_n\}$ in which each instance ar_i represents an archaeological remain. Each skeleton is represented by m characteristics (features) $f_1, f_2, \ldots, f_m$ which correspond to different measurements that were performed on it. Usually, the measurements are numerical values and correspond to several significant bones in the body. Thus, an instance ar_i is characterized by an m-dimensional vector, i.e $ar_i = (ar_{i1}, ar_{i2}, \ldots, ar_{im})$ where ar_{ij} $\forall 1 \leq i \leq n$ represents the value of measurement f_j applied to the skeletal remain ar_i.

4.2. **Data pre-processing and feature selection.** The first step in building the machine learning model is the *data pre-processing* step. During this step, an analysis is performed in order to determine those measurements (features) that highly influence the gender classification task. We will use the *information gain* measure which expresses the expected reduction in entropy determined by partitioning the instances according to a given feature [18].

First, we are computing the information gain measure for each feature from the feature set. For computing the information gain of a feature, the values of the feature are first discretized (the feature domain is divided in ten intervals of equal size). Let us consider that $f_{o_1}, f_{o_2}, \ldots, f_{o_m}$ are the features in the increasing order of their information gain. Then, we are trying to remove subsets of features considering the determined order, i.e $\{f_{o_1}\}$, $\{f_{o_1}, f_{o_2}\}$, etc. In order to decide the impact of removing the subset $\{f_{o_1}, \ldots, f_{o_k}\}$ $(1 \leq k \leq m-1)$ of features from the initial feature set, a *decision tree* is built from the set of instances and the currently selected features. For estimating the performance of the constructed decision tree, a k-fold cross-validation is applied on the training data set. Finally, we decide to remove (from the initial feature set) the subset $\{f_{o_1}, \ldots, f_{o_p}\}$ $(1 \leq p \leq m-1)$ of features for which the corresponding decision tree provided a unique maximum average accuracy during the k-fold cross-validation $(k \geq 10)$.

After the feature selection technique is applied, the input data is scaled to [0,1].

4.3. **Building the model.** After the data set is pre-processed as indicated above, the k-means and PAM clustering algorithms will be applied in order to build the unsupervised learning model.

Before applying the partitional clustering algorithms, the number k of clusters have to be identified. For determining the number of clusters, we are using a cluster validity index, the *Dunn index* [19]. This index expresses the quality of a given partition (clustering). The greater the value of this index, the better a partition is, therefore the *Dunn index* should be maximized in order to obtain better partitions. The clustering algorithms are applied considering different values for the number of clusters (starting from two clusters). The number k of clusters which provided the partition with the highest *Dunn index* will be reported as the correct number of clusters, i.e the number of clusters that have to be identified in data.

After the number of clusters was determined, we are using an heuristic method for selecting two initial representative instances (i.e skeletons), which will be used as initial medoids/centroids in the clustering process. The heuristic is described below.

(i) The first representative skeleton is the most "distant" one from all other skeletons from the data set (i.e the skeleton whose average distance from all other skeletons is maximum).
(ii) In order to select the second representative instance we reason as follows. For each remaining instance (that was not selected), we compute the minimum distance (*dist*) from the instance and the already chosen representative instance. As the second representative instance we will choose the skeleton that maximizes *dist*.

The representative instances identified using the heuristic below will be selected as initial medoids/centroids in the clustering process. After the initial medoids/centroids are selected, the PAM/k-means algorithm will be applied and the output partition is reported. We mention that, if a cluster becomes empty during the iterative process, the number of clusters will be decreased.

After the unsupervised learning model was built, a testing step is performed in order to evaluate its performance. Details about the testing step will be given in Section 5.2.1.

5. Experimental evaluation

This section contains the experimental evaluation of the clustering algorithms (described in Section 4) on three case studies which were conducted starting from two data sets obtained from the literature [3]. This data sets were used for determining the gender of South African skeletal remains based on different measurements of the forearm bones, ulna respectively the radius.

5.1. **Case studies.** The data set from [3] consists of 200 male and 200 female skeletons from the Pretoria Bone and Raymond A. Dart collections. Ten anthropometric measurements were taken from the radius bone and nine measurements were taken from the ulna bone. The skeletal remains represent black South Africans from the 19^{th} and 20^{th} centuries, being born between 1863 and 1996.

In each data set considered for evaluation, the instances (skeletons) within the data sets are labeled as being **male** or *female*.

The first case study we are considering for evaluation consists of human remains identified by ten radial measurements. Thus, there are 10 features characterizing the instances within the data set. The features represent the following radial measurements [3]: maximum length of the radius (F1), distal breadth (F2), circumference at the midshaft (F3), sagittal diameter at midshaft (minimum diameter) (F4), transverse diameter at midshaft (maximum diameter) (F5), vertical radial head height (F6), minimum head diameter (F7), maximum head diameter (F8), circumference of the radial (F9) and circumference at the tuberosity (F10).

As the second case study, nine measurements of the ulna bone are used for the human skeletons. There are 9 features describing the instances within the data set: maximum length of the ulna (F1), maximum length of the ulna measured using the plumbline geniometer method (F2), anterior-posterior diameter (minimum diameter) (F3), medial-lateral diameter (maximum diameter) (F4), circumference at midshaft (F5), minimum circumference of the ulna (F6), olecranon breadth (F7), minimum olecranon breadth (F8) and height of the olecranon (F9).

The previously described data sets were previously used in [3] for the gender identification task. We are considering in this paper as our third case study the data set which contains both radial and ulnar measurements. Consequently, in this data set, each skeleton (instance) will be represented by 19 measurements (features): the first ten are the radial measurements (as for the first case study) and the next nine features represent the ulnar measurements (as for the second case study).

5.2. **Results.** For all the case studies considered for evaluation, the methodology presented in Section 4 will be applied. After the feature selection step, the number of clusters which have to be determined in the data set, as well as the initial centroids/medoids are identified as indicated in Section 4.3. We mention that for all the considered case studies, the number k of desired clusters (heuristically determined using the *Dunn* index) is two. Obviously, two clusters are expected, since the problem we are approaching in this paper is a binary classification problem.

5.2.1. *Evaluation measures.* After two clusters are detected in data using the clustering algorithm (k-means or k-medoids), we identified which is the male cluster and which is the female cluster. Since the labels of the instances within the input data set are known, the cluster which has the number of males greater than the number of females will be considered the male cluster. The remaining cluster will be considered a female cluster.

Since the gender detection task is a binary classification problem, the confusion matrix will be computed. We assume in the following that that the male class is the positive one and the female class is the negative one. Thus, we have to compute the number of true positives (TP - number of males correctly identified), the number of true negatives (TN - number of females correctly identified), the number of false positive (FP - number of misclassified males) and the number of false negative (FN - number of misclassified females).

Two evaluation measures will be further used in order to test the performance of the clustering algorithms for gender identification. These measures are *external evaluation* measures since their computation requires the class label of each instance.

(1) The *accuracy* (denoted by Acc) measures the percentage of instances that are correctly classified, $Acc = \frac{TP+TN}{TP+TN+FP+FN}$.
(2) The *Area under the ROC curve* measure (denoted by AUC) which is considered as one of the best evaluation measures used to compare classifiers [17, 10]. The AUC measure expresses the area under the ROC curve (Receiver Operating Characteristics curve). The ROC curve is obtained by connecting the ($recall$, 1-$specificity$) point to the points at (0,0) and (1,1) [10]. The *recall* of a binary classifier is computed as $recall = \frac{TP}{TP+FN}$ and the *specificity* of the classifier is computed as $specificity = \frac{TN}{TN+FP}$.

Good classifiers have high *accuracy* and AUC values. Thus, these measures need to be maximized in order to obtain better classifiers.

5.2.2. *First case study.* As mentioned in Section 5.1, there are 10 initial features used for the the gender classification task. First, the information gain for the features is determined. The obtained values are depicted in Figure 1.

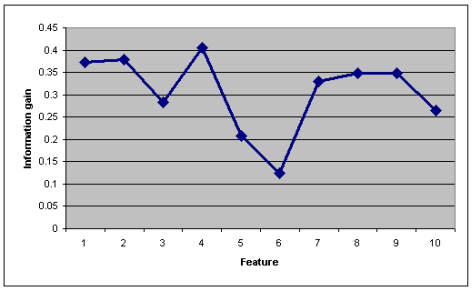


FIGURE 1. Information gain for the features from the first case study

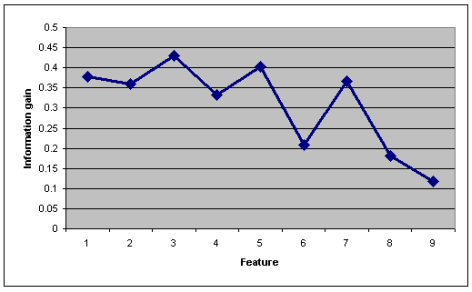


FIGURE 2. Information gain for the features from the second case study

For the first case study, the features in their increasing order of the information gain are $\{F6, F5, F10, F3, F7, F9, F8, F1, F2, F4\}$ (see Figure 1). We applied the feature selection step described in Section 4.2 in order to identify a subset of relevant features. The resulting set of features is $\{F1, F2, F4\}$ (i.e the subset of features which are removed is $\{F6, F5, F10, F3, F7, F9, F8\}$) since it provided a *decision tree* having a maximum accuracy of 86.25% using 10-fold cross validation. Then, the data is scaled to [0,1].

Table 1 presents the values of the evaluation measures (Section 5.2.1) obtained by applying the k-means and PAM algorithms on the first data set,

considering different set of features. The best obtained results are marked with bold. We notice that for the set of features identified after the feature selection step, the values for the *Acc* and *AUC* measures are very close (for both algorithms) to the maximum values. We note that when using only the feature set $\{F2\}$, we obtain only one cluster, which is not our purpose. Thus, we do not report in Table 1 the results obtained when using only feature $F2$ for classification.

Feature set	**k-means**		**k-medoids**	
	Acc	**AUC**	**Acc**	**AUC**
$\{F6, F5, F10, F3, F7, F9, F8, F1, F2, F4\}$	0.837	0.837	0.837	0.837
$\{F5, F10, F3, F7, F9, F8, F1, F2, F4\}$	0.837	0.837	0.837	0.837
$\{F10, F3, F7, F9, F8, F1, F2, F4\}$	0.837	0.837	0.832	0.832
$\{F3, F7, F9, F8, F1, F2, F4\}$	0.835	0.835	0.830	0.830
$\{F7, F9, F8, F1, F2, F4\}$	0.847	0.847	0.842	0.842
$\{F9, F8, F1, F2, F4\}$	0.842	0.842	**0.845**	**0.845**
$\{F8, F1, F2, F4\}$	**0.855**	**0.855**	0.835	0.838
$\{F1, F2, F4\}$	0.850	0.851	0.842	0.843
$\{F2, F4\}$	0.822	0.824	0.825	0.826

TABLE 1. Results obtained for the first case study.

Table 4 depicts the minimum, maximum, mean and standard deviation of the *Acc* and *AUC* values obtained on the first case study.

5.2.3. *Second case study.* For the second case study, the features in their increasing order of the information gain are $\{F9, F8, F6, F4, F2, F7, F1, F5, F3\}$ (see Figure 2). We applied the feature selection step described in Section 4.2 in order to identify a subset of relevant features. The resulting set of features is $\{F7, F1, F5, F3\}$ (i.e the subset of features which are removed is $\{F9, F8, F6, F4, F2\}$) since it provided a *decision tree* having a maximum accuracy of 87.25% using 13-fold cross validation. Then, the data is scaled to [0,1].

Table 2 presents the values of the evaluation measures (Section 5.2.1) obtained by applying the k-means and PAM algorithms on the second data set, considering different set of features. For both algorithms, the best obtained results are marked with bold and correspond to the feature set that was identified during the feature selection step described above.

Table 4 illustrates the minimum, maximum, mean and standard deviation of the *Acc* and *AUC* values obtained on the second case study.

Feature set	k-means		k-medoids	
	Acc	**AUC**	**Acc**	**AUC**
$\{F9, F8, F6, F4, F2, F7, F1, F5, F3\}$	0.865	0.865	0.875	0.875
$\{F8, F6, F4, F2, F7, F1, F5, F3\}$	0.870	0.870	0.875	0.875
$\{F6, F4, F2, F7, F1, F5, F3\}$	0.867	0.867	0.870	0.870
$\{F4, F2, F7, F1, F5, F3\}$	0.872	0.872	0.860	0.860
$\{F2, F7, F1, F5, F3\}$	0.877	0.878	0.880	0.881
$\{F7, F1, F5, F3\}$	**0.877**	**0.878**	**0.882**	**0.884**
$\{F1, F5, F3\}$	0.862	0.862	0.862	0.862
$\{F5, F3\}$	0.840	0.840	0.847	0.847
$\{F3\}$	0.827	0.828	0.835	0.836

TABLE 2. Results obtained for the second case study.

5.2.4. *Third case study.* For the third case study, the features in their increasing order of the information gain are $\{F19, F6, F18, F16, F5, F10, F3, F7, F14, F9, F8, F12, F17, F1, F2, F11, F15, F4, F13\}$. The obtained values for the features information gain are depicted in Figure 3.We applied the feature selection step described in Section 4.2 in order to identify a subset of relevant features. The resulting set of features is $\{F2, F11, F15, F4, F13\}$ (i.e the subset of features which are removed is $\{F19, F6, F18, F16, F5, F10, F3, F7, F14, F9, F8, F12, F17, F1\}$) since it provided a *decision tree* having a maximum accuracy of 88.25% using 15-fold cross validation. Then, the data is scaled to [0,1].

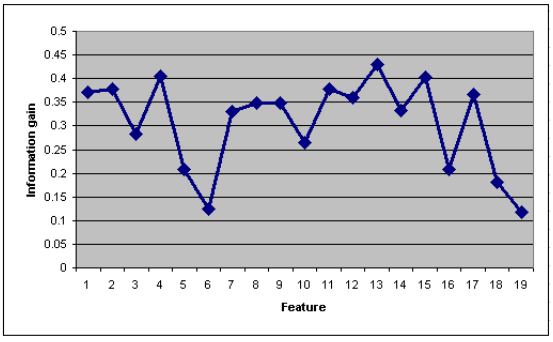


FIGURE 3. Information gain for the features from the third case study

Table 3 presents the values of the evaluation measures (Section 5.2.1) obtained by applying the k-means and PAM algorithms on the third data set,

Feature set	**k-means**		**k-medoids**	
	Acc	**AUC**	**Acc**	**AUC**
$\{F19, F6, F18, F16, F5, F10, F3, F7, F14, F9,$ $F8, F12, F17, F1, F2, F11, F15, F4, F13\}$	0.857	0.858	0.867	0.868
$\{F6, F18, F16, F5, F10, F3, F7, F14, F9, F8, \}$ $F12, F17, F1, F2, F11, F15, F4, F13\}$	0.857	0.858	0.840	0.841
$\{F18, F16, F5, F10, F3, F7, F14, F9, F8, F12, \}$ $F17, F1, F2, F11, F15, F4, F13\}$	0.857	0.858	0.867	0.868
$\{F16, F5, F10, F3, F7, F14, F9, F8, F12, F17, \}$ $F1, F2, F11, F15, F4, F13\}$	0.862	0.863	0.865	0.866
$\{F5, F10, F3, F7, F14, F9, F8, F12, F17, F1, \}$ $F2, F11, F15, F4, F13\}$	0.862	0.863	0.860	0.860
$\{F10, F3, F7, F14, F9, F8, F12, F17, F1, F2, \}$ $F11, F15, F4, F13\}$	0.862	0.863	0.857	0.858
$\{F3, F7, F14, F9, F8, F12, F17, F1, F2, F11, \}$ $F15, F4, F13\}$	0.867	0.868	0.865	0.865
$\{F7, F14, F9, F8, F12, F17, F1, F2, F11, F15, \}$ $F4, F13\}$	0.875	0.877	0.857	0.858
$\{F14, F9, F8, F12, F17, F1, F2, F11, F15, F4\}$ $F13\}$	0.870	0.873	0.872	0.878
$\{F9, F8, F12, F17, F1, F2, F11, F15, F4, F13\}$	0.865	0.865	0.867	0.868
$\{F8, F12, F17, F1, F2, F11, F15, F4, F13\}$	0.880	0.885	0.877	0.885
$\{F12, F17, F1, F2, F11, F15, F4, F13\}$	0.887	0.890	0.870	0.879
$\{F17, F1, F2, F11, F15, F4, F13\}$	0.880	0.882	0.852	0.856
$\{F1, F2, F11, F15, F4, F13\}$	**0.890**	**0.892**	0.882	0.883
$\{F2, F11, F15, F4, F13\}$	0.882	0.884	**0.890**	**0.890**
$\{F11, F15, F4, F13\}$	0.857	0.857	0.860	0.860
$\{F15, F4, F13\}$	0.845	0.845	0.852	0.852
$\{F4, F13\}$	0.860	0.860	0.845	0.845
$\{F13\}$	0.827	0.828	0.835	0.836

TABLE 3. Results obtained for the third case study.

considering different set of features. The best obtained results are marked with bold. We notice that for the k-medoids algorithm, the best results corresponds to the feature set that was identified during the feature selection step described above. For the k-means algorithm, for the set of features identified after the feature selection step, the values for the Acc and AUC measures are very close to the maximum values.

The minimum, maximum, mean and standard deviation of the Acc and AUC values obtained on the third case study are given in Table 4.

Case study	**Algorithm**	**Acc**				**AUC**			
		Mean	Min	Max	Stdev	Mean	Min	Max	Stdev
First	k-means	0.840	0.822	0.855	0.009	0.840	0.824	0.855	0.009
First	k-medoids	0.836	0.825	0.845	0.006	0.836	0.826	0.845	0.006
Second	k-means	0.861	0.827	0.877	0.017	0.862	0.828	0.878	0.017
Second	k-medois	0.865	0.835	0.882	0.015	0.865	0.836	0.884	0.015
Third	k-means	0.865	0.826	0.89	0.015	0.866	0.828	0.892	0.015
Third	k-medoids	0.862	0.835	0.89	0.013	0.863	0.836	0.890	0.014

TABLE 4. Results for all performed experiments.

6. Discussion and comparison to related work

Analyzing the experimental results presented in Section 5 we observe the following.

For the first and the third case studies, the maximum *Acc* and *AUC* values are obtained using the k-means algorithm. The k-medoids algorithm provided the maximum *Acc* and *AUC* values for the second case study. We notice that, for each case study, the values reported by the two clustering algorithms are sensitively equal. From all the performed experiments, the best values were reported for the k-means algorithm, on the third case study (an *Acc* of 0.89 and an *AUC* of 0.892). Moreover, we observe, for each performed experiment, a small value for the standard deviation, which indicates a good precision of the obtained results.

As we have shown in Section 2.1, most of the approaches existing in the literature for determining the sex of skeletal remains are based on bone measurements, DNA and gene analysis or different statistical methods. As far as we know, there are no approaches that use clustering for learning to identify the gender of human skeletons.

There is only one approach in the literature that uses the same data sets as in our paper, a discriminant analysis method which was introduced in [3]. Five discriminant functions were used in this paper for the data set we have considered in our first case study and four functions were used for the data set considered in our second case study. For estimating the performance of the gender prediction task, only the accuracy is reported in [3], thus we will also use for comparison this evaluation measure.

Table 5 comparatively presents the values for the *Acc* measures reported by our clustering algorithms (using the feature selection method indicated in Subsection 4.2), as well as the average accuracy value reported by the discriminant analysis method from [3]. We note that in [3] the evaluation is made using the "leave-one-out" cross-validation method.

Case study	Algorithm	Accuracy
First	k-means	**0.850**
First	k-medoids	0.842
First	Discriminant functions [3]	0.838
Second	k-means	0.877
Second	k-medoids	**0.882**
Second	Discriminant functions [3]	0.843
Third	k-means	0.882
Third	k-medoids	**0.890**
Third	Discriminant functions [3]	-

TABLE 5. Comparative results on the considered case studies.

From Table 5 we observe that our clustering algorithms outperform, for all the considered case studies, the discriminant analysis method from [3] (considering the average accuracy reported). For each case study, the best obtained accuracy is highlighted. We also observe that the maximum value for the accuracy was obtained on the third case study using the k-medoids method. As in the third experiment there was no value for the discriminant functions, in the following graphs, the average of the results obtained in the first and second cases was considered. Figures 6 indicate, for each case study considered in our experiments, the comparative results from Table 5, emphasizing the good performance of our clustering based methods.

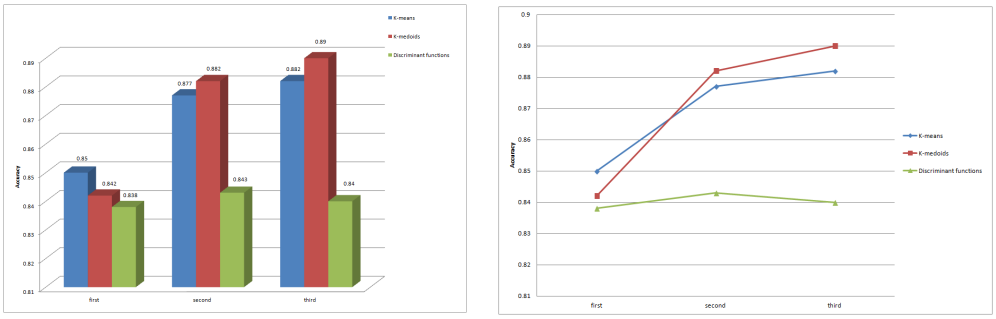


FIGURE 4. Comparative results.

Considering all case studies, an average accuracy of 0.8705 was obtained. The 95% Confidence Interval [6] for the average value is (0.86, 0.89). Thus, there is a 95% confidence that the accuracy of the partition obtained using clustering exceeds the best value reported in the literature with at least 1.7% and at most 4.7%. Other approaches existing in the literature for gender detection perform experiments on various data sets which are different from

the ones we are considering in this paper. Thus, a comparison with these approaches is hard to be conducted.

7. Conclusions and further work

In this paper we proposed an unsupervised machine learning based approach for detecting the gender of human skeletons from bone measurements. The experimental evaluation is performed on two publicly available human skeletal remains data sets and three case studies were conducted in order to test the performance of our technique. The experimental results obtained on two open-source data sets reveal that our approaches outperform the similar approaches from the literature.

Experimental evaluations of the proposed clustering methods on real data sets [13] will be further conducted in order to better test their accuracy. We also plan to investigate the effectiveness of *fuzzy* clustering and decision tree learning [18] for the problem of gender detection of human skeletal remains.

References

[1] I. Afrianty, D. Nasien, M.R.A. Kadir, and H. Haron. Determination of gender from pelvic bones and patella in forensic anthropology: A comparison of classification techniques. In *Artificial Intelligence, Modelling and Simulation (AIMS), 2013 1st International Conference on*, pages 3–7, Dec 2013.

[2] Mitra Akhlaghi, Ardeshir Sheikhazadi, Ali Naghsh, and Gholamali Dorvashi. Identification of sex in iranian population using patella dimensions. *Journal of Forensic and Legal Medicine*, 17(3):150 – 155, 2010.

[3] Isabelle Linda Odile Barrier. Sex determination from the bones of the forearm in a modern South African sample. PhD thesis, University of Pretoria, 2007.

[4] S. Bell and R. Jantz. Neural network classification of skeletal remains. In G. Burenhult and J. Arvidsson, editors, *Archaeological Informatics: Pushing The Envelope. CAA2001*, pages 205–212. Archaeopress, Oxford.

[5] M.A. Bidmos, V.E. Gibbon, and G. Strkalj. Recent advances in sex identification of human skeletal remains in South Africa. *South African Journal Of Science*, 106, 2010.

[6] LD. Brown, TT. Cat, and A DasGupta. Interval Estimation for a proportion. *Statistical Science*, 16:101 – 133, 2001.

[7] J. Bruzek. A method for visual determination of sex, using the human hip bone. *American Journal Of Physical Anthropology*, 117(2):157–68, Feb 2002.

[8] M. Faerman, D. Filon, G. Kahila, C. L. Greenblatt, P. Smith, and A. Oppenheim. Sex identification of archaeological human remains based on amplification of the x and y amelogenin alleles. *Gene*, 167(1-2):327–32, 1995.

[9] A.B. Falsetti. Sex assessment from metacarpals of the human hand. *J Forensic Sci.*, 40(5):774-776, 1995.

[10] Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[11] Priscilla E. Greenwood and Michael S. Nikulin. *A Guide to Chi-Squared Testing*. Wiley Series in Probabilities and Statistics. Wiley, 1996.

[12] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
[13] Institute of Interdisciplinary Research in Bio-Nano-Sciences. http://bionanosci.institute.ubbcluj.ro/.
[14] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.
[15] L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids*. Statistical Data Analysis Based on the L_1Norm and Related Methods, edited by Y. Dodge, North-Holland. pages 405416, 1987.
[16] Jan Kiesslich, Franz Neuhuber, Harald J. Meyer, Max P. Baur, and Jutta Leskovarm. DNA analysis on biological remains from archaeological findings - sex identification and kinship analysis on skeletons from Mitterkirchen, upper Austria. In: *Interpretierte Eisenzeiten. Fallstudien, Methoden, Theorie.*, eds. Raimund Karl - Jutta Leskovar. pages 147–154, 2005.
[17] Nada Lavrac, Branko Kavsek, Peter A. Flach, and Ljupco Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.
[18] Thomas M. Mitchell. *Machine learning*. McGraw-Hill, Inc. New York, USA, 1997.
[19] Malay K. Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik. Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3):487 – 501, 2004.
[20] St. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall International Series in Artificial Intelligence. Prentice Hall, 2003.
[21] JC. Stevenson, ER Mahoney, PL. Walker, and PM. Everson. Technical note: prediction of sex based on five skull traits using decision analysis (CHAID). *Am J Phys Anthropol*, 139(3):434 – 441, 2009.
[22] D. Troy Case and Ann H. Ross. Sex determination from hand and foot bone lengths. *J Forensic Sci.*, 52(2):264-70, 2007.
[23] Michaela Vanharová and Eva Drozdová. Sex determination of 4000 years old children and juveniles skeletal remains from Eneolithic burial site Hostice 1 za Hanou (Czech Republic) by ancient DNA analysis. *Anthropological Review*, 71(1):63-70, 2008.
[24] Marin Vodanovic, Zeljko Demo, Vera Njemirovskij, Jadranka Keros, and Hrvoje Brkic. Odontometrics: a useful method for sex determination in an archaeological skeletal population? *Journal of Archaeological Science*, 34:905–913, Sep 2007.
[25] Yo Wada. Discriminant function for sex determination of ancient Iraqis based on radial measurements. *Anthropological Science*, 102:149-158, 1994.
[26] K.M. Weiss. On the systematic bias in skeletal sexing. *American Journal Of Physical Anthropology*, 37(2):239–49, Sep 1972.
[27] G. Zeybek, I. Ergur, and Z. Demiroglu. Stature and gender estimation using foot measurements. *Forensic Sci Int.*, 181(1-3):54.e1-5, 2008.

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

E-mail address: mircea@cs.ubbcluj.ro, {lsir1296, pmir1335}@scs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

AN INFEASIBLE INTERIOR-POINT METHOD FOR THE CARTESIAN $P_*(\kappa)$ SECOND-ORDER CONE LINEAR COMPLEMENTARITY PROBLEM WITH ONE CENTERING STEP

BEHROUZ KHEIRFAM

Abstract. In this paper, we present a new full step infeasible interior-point algorithm for the Cartesian $P_*(\kappa)$ linear complementarity problem over second-order cones. The algorithm uses only full Nesterov and Todd steps. Each (main) iteration of the algorithm consists of one so-called feasibility step and only one centering step. The algorithm starts with a strictly feasible point of a perturbed problem, after an iteration, the new iterate is still strictly feasible of the new perturbed problem. The algorithm has the same complexity as the best known infeasible interior-point methods.

1. Introduction

In this paper, we consider the second-order cone linear complementarity problem (SOCLCP), which seeks vectors $x, s \in R^n$ such that

$$x \in \mathcal{K},\ s \in \mathcal{K},\ s = \mathcal{A}(x) + q,\ \langle x, s\rangle = 0,$$

where $\langle x, s\rangle := \mathrm{tr}(x \circ s)$ denotes the Euclidean inner product, $q \in R^n$, $\mathcal{A} : \mathcal{K} \to \mathcal{K}$ is a linear transformation, and $\mathcal{K} \subseteq R^n$ is the Cartesian product of several second-order cones, i.e., $\mathcal{K} = \mathcal{K}^1 \times \mathcal{K}^2 \times \cdots \times \mathcal{K}^N$, with

$$\mathcal{K}^j := \left\{(x_1, x_{2:n_j}^T)^T \in R \times R^{n_j-1} : x_1 \geq \|x_{2:n_j}\|\right\},\ \text{where } x_{2:n_j} := (x_2; \ldots; x_{n_j})$$

for each $j = 1, \ldots, N$ and $\sum_{j=1}^N n_j = n$. Since $\mathcal{K}$ has finite dimensional, we can consider matrix representation of the linear transformation $\mathcal{A}(x) = Mx$,

Received by the editors: March 4, 2015.

2010 *Mathematics Subject Classification.* 90C51, 90C33.

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

Key words and phrases. The Cartesian product of second-order cones, Linear complementarity problem, Infeasible interior-point method, Polynomial complexity.

with $M \in R^{n\times n}$. By Lemma 2.2 in [7] we know that $\langle x, s\rangle = 0$ if and only if $x \circ s = 0$. Therefore, we may rewrite SOCLCP in the following form

$$x \in \mathcal{K},\ s \in \mathcal{K},\ s = Mx + q,\ x \circ s = 0. \tag{1}$$

We call SOCLCP the Cartesian $P_*(\kappa)$-SOCLCP if the matrix M has the Cartesian $P_*(\kappa)$-property, i.e., for any $\kappa \geq 0$, the matrix M satisfies

$$\langle x, Mx\rangle \geq -4\kappa \sum_{j\in I_+(x)} \langle x^{(j)}, [Mx]^{(j)}\rangle, \text{ where } I_+(x) = \{j : \langle x^{(j)}[Mx]^{(j)}\rangle \geq 0\}.$$

The concept of the Cartesian $P_*(\kappa)$-property was first introduced by Luo and Xiu [15] in the general Euclidean Jordan algebra. Actually, it is a straightforward extension of the $P_*(\kappa)$-matrix introduced by Kojima et al. [14]. Moreover, the matrix M with the Cartesian $P_*(\kappa)$-property becomes the usual $P_*(\kappa)$-matrix when $\mathcal{K}$ is specified to be R^n_+, correspondingly, the Cartesian $P_*(\kappa)$-SOCLCP reduces to the $P_*(\kappa)$-LCP [15]. Wang and Zhu [25] presented a primal-dual interior-point algorithm for the Cartesian $P_*(\kappa)$-SOCLCP based on a parametric kernel function. The primal-dual full-Newton step feasible IPM for linear optimization (LO) was first analyzed by Roos et al. [18]. Darvay [4] proposed a full-Newton step primal-dual path-following interior-point algorithm for LO which is based on the equivalent algebraic transformation. Achache [1], Wang and Bai [22, 23] and Wang [24] generalized the results for LO in [4] to convex quadratic optimization (CQO), second-order cone optimization (SOCO), symmetric cone optimization (SCO) and monotone LCP over symmetric cone (SCLCP).

The above algorithms enjoy the best known iteration bound. However, they are all feasible IPMs, which start with a strictly feasible interior point and maintain feasibility during the solution process. One may distinguish between feasible IPMs and infeasible IPMs (IIPMs), which start with an arbitrary positive point and feasibility is reached as optimality is approached. In 2006, Roos [17] designed the first full-Newton step primal-dual IIPM with the currently best iteration bound for LO. Following Roos' contribution, Kheirfam and Mahdavi-Amiri [11] and Gu et al. [8] respectively extended both versions of the feasible IPM [18] and IIPM [17] to SCLCP and SCO by using Nesterov and Todd (NT) direction as a search direction and obtained the same iteration complexity bounds. Kheirfam and Mahdavi-Amiri [12] presented a full NT-step IIPM for SCLCP based on modified NT directions, and the corresponding complexity results accord with the currently best-known iteration bound for IIPMs. Based on Darvay's technique [4] extension to SCO in [23], Kheirfam [10] presented a full-NT step IIPM for SCO. Recently, Kheirfam [9] designed and analyzed the full-Newton step IIPM based on a new proximity measure for $P_*(\kappa)$ horizontal linear complementarity problem (HLCP). All

IIPMs mentioned so far consists of one feasibility step and a few - at most three - centering steps. Recently, Darvay et al. [5] presented an improved version of an IIPM for LO in [2], in sense that each iteration of the algorithm consists of one feasibility step and only a centering step.

Motivated by Darvay et al.'s recent work, we present a new full NT-step IIPM for the Cartesian $P_*(\kappa)$-SOCLCP based on the technique introduced in [5] and prove that each main iteration needs to a feasibility step and one centering step in order to get a well-defined algorithm. The new algorithm reduces the searching steps in each iteration and tendering an interesting analysis for iteration complexity.

The remainder of our work is organized as follows. In Section 2, we briefly recall the corresponding Euclidean Jordan algebra to second-order cones. Based on Darvay's technique, we are providing some new results that will be used in the complexity analysis of the algorithm. In Section 3, we introduce the perturbed problem and the new infeasible interior-point algorithm. Then, we provide the complexity analysis of the algorithm and derive the iteration bound. Finally, some conclusions are given in Section 4.

2. Euclidean Jordan algebra and some results

In this section, we first recall some basic concepts of Euclidean Jordan algebra [3, 6], and then we provide some results that will be used for the main purpose of this paper.

A Euclidean Jordan algebra $(\mathcal{J}, \langle\cdot,\cdot\rangle, \circ)$ ($\mathcal{J}$ for short) is an n-dimensional inner product space over R endowed with a bilinear map $\circ : \mathcal{J} \times \mathcal{J} \to \mathcal{J}$ iff for all $x, y, z \in \mathcal{J}, x \circ y = y \circ x, x \circ (x^2 \circ y) = x^2 \circ (x \circ y)$ and $\langle x \circ y, z\rangle = \langle x, y \circ z\rangle$ where $x^2 := x \circ x$. For any $x^j = (x_1^j, x_{2:n_j}^j), s^j = (s_1^j, s_{2:n_j}^j) \in R \times R^{n_j-1}$, the Jordan product of x^j and s^j is defined as

$$x^j \circ s^j = \big((x^j)^T s^j; x_1^j s_{2:n_j}^j + s_1^j x_{2:n_j}^j\big).$$

One can easily verify that $(R^{n_j}, \circ)$ is a Euclidean Jordan algebra, with $e^j = (1; 0) \in R \times R^{n_j-1}$ as identity element. In this algebra, the second-order cone $\mathcal{K}^j$ is the cone of square, i.e., $\mathcal{K}^j = \{x^2 : x \in R^{n_j}\}$ (see, [6]). Given a vector $x^j = (x_1^j, x_{2:n_j}^j) \in R \times R^{n_j-1}$, let

$$L(x^j) := \begin{bmatrix} x_1^j & (x^j)_{2:n_j}^T \\ x_{2:n_j}^j & x_1^j E_{n_j-1} \end{bmatrix},$$

which can be viewed as a linear mapping from R^{n_j-1} to R^{n_j-1}, where E_{n_j-1} denotes the identify matrix. It is not hard to verify that $L(x^j)s^j = x^j \circ s^j$

for any $x^j, s^j \in R^{n_j}$. The eigenvalues of $L(x^j)$ are denoted respectively as $\lambda_{\min}(x^j) = x_1^j - \|x_{2:n_j}^j\|$ and $\lambda_{\max}(x^j) = x_1^j + \|x_{2:n_j}^j\|$. Note that

$$x^j \in \mathcal{K}^j \Leftrightarrow \lambda_{\min}(x^j) \geq 0,\ x^j \in \text{int}\mathcal{K}^j \Leftrightarrow \lambda_{\min}(x^j) > 0,$$

where int$\mathcal{K}^j$ denotes the interior of $\mathcal{K}^j$. For any $x^j \in R^{n_j}, P(x^j) := 2L(x^j)^2 - L((x^j)^2)$ where $L(x^j)^2 = L(x^j)L(x^j)$. The map $P(x^j)$ is called the quadratic representation of x^j. Each $x^j = (x_1^j; x_{2:n_j}^j) \in R^{n_j}$ admits a spectral decomposition, associated with $\mathcal{K}^j$, of the form $x^j = \lambda_{\max}(x^j)c_1 + \lambda_{\min}(x^j)c_2$, where c_1, c_2 are the associated eigenvectors given by

$$c_1 = \frac{1}{2}\big(1; \frac{x_{2:n_j}^j}{\|x_{2:n_j}^j\|}\big),\ c_2 = \frac{1}{2}\big(1; \frac{-x_{2:n_j}^j}{\|x_{2:n_j}^j\|}\big). \tag{2}$$

Moreover, $\text{tr}(x^j) = \lambda_{\max}(x^j) + \lambda_{\min}(x^j) = 2x_1^j$. The natural inner product is given by

$$\langle x^j, s^j\rangle := \text{tr}(x^j \circ s^j) = 2(x^j)^T s^j,\ x^j, s^j \in R^{n_j}.$$

Hence, the norm induced by this inner product, which is denoted by $\|\cdot\|_F$, satisfies

$$\|x^j\|_F = \sqrt{\langle x^j, x^j\rangle} = \sqrt{\text{tr}((x^j)^2)} = \sqrt{\lambda_{\min}(x^j)^2 + \lambda_{\max}(x^j)^2} = \sqrt{2}\|x^j\|.$$

In the sequel, we generalize the above definitions and properties to the case where $N > 1$, when the second-order cone underlying $\mathcal{K}$ is the Cartesian product of N second-order cones $\mathcal{K}^j$. For any $x = (x^1; \cdots; x^N) \in R^n$ with $x^j \in R^{n_j}, j = 1, \ldots, N$, the algebra $(R^n, \circ)$ is defined as a direct product of the Jordan algebras $(R^{n_j}, \circ)$ as

$$x \circ s := (x^1 \circ s^1; \cdots; x^N \circ s^N).$$

Obviously, if $e^j \in \mathcal{K}^j$ is the identity element in the Jordan algebra for the jth second-order cone, then the vector $e = (e^1; \cdots; e^N)$ is the identity element in $(R^n, \circ)$. Moreover, $\text{tr}(e) = 2N$, which is the rank of $(R^n, \circ)$. The matrix $L(x)$ and the quadratic representation $P(x)$ of $(R^n, \circ)$ can be respectively adjusted to

$$L(x) := \text{diag}\big(L(x^1), \cdots, L(x^N)\big),\ P(x) := \text{diag}\big(P(x^1), \cdots, P(x^N)\big).$$

Furthermore

$$\lambda_{\max}(x) = \max_{1\leq j\leq N}\{\lambda_{\max}(x^j)\},\ \lambda_{\min}(x) = \min_{1\leq j\leq N}\{\lambda_{\min}(x^j)\}, \|x\|_F^2 = \sum_{j=1}^{N}\|x^j\|_F^2$$

$$\text{tr}(x) = \sum_{j=1}^{N}\text{tr}(x^j) = \sum_{j=1}^{N}\big(\lambda_{\max}(x^j) + \lambda_{\min}(x^j)\big) := \sum_{j=1}^{2N}\lambda_j(x).$$

Lemma 2.1. (Corollary 2.14 in [22]) *Let $x, s \in R^n$ and $x+s=e$. If $|\lambda_{\min}(s)|$ is small enough, then $\psi(x+s) \approx \psi(x)+\psi^{'}(x)\circ s$, where $\psi(t):[0,\infty)\rightarrow(0,\infty)$ such that $\psi^{'}(t)>0$ for all $t>0$.*

Lemma 2.2. (Lemma 6.1 in [22]) *Let $x(\alpha):=x+\alpha\Delta x$ and $s(\alpha):=s+\alpha\Delta s$ for all $0\leq\alpha\leq 1$. Suppose that $x, s\in \text{int}\mathcal{K}$. If one has*

$$\det\big(x(\alpha)\circ s(\alpha)\big)>0,\ \forall 0\leq\alpha\leq\bar{\alpha},$$

then $x(\bar{\alpha}), s(\bar{\alpha})\in \text{int}\mathcal{K}$.

Lemma 2.3. (Theorem 4 in [20]) *Let $x, s\in\mathcal{K}$. Then*

$$\lambda_{\min}\big(P(x)^{\frac{1}{2}}s\big)\geq\lambda_{\min}(x\circ s).$$

Lemma 2.4. (Lemma 30 in [19]) *Let $x, s\in\mathcal{K}$. Then*

$$\big\|P(x)^{\frac{1}{2}}s-e\big\|_F\leq\big\|x\circ s-e\big\|_F.$$

Luo and Xiu [15] have discussed the existence and uniqueness of the central path of the Cartesian $P_*(\kappa)$ symmetric cone linear complementarity problem ($P_*(\kappa)$-SCLCP). As a special case of the Cartesian $P_*(\kappa)$-SCLCP, the existence and uniqueness of the central path of the Cartesian $P_*(\kappa)$-SOCLCP could be similarly obtained. The main idea of IPMs is to replace the last equation in (1), the so-called complementarity condition, with the parameterized equation $x\circ s=\mu e$, with parameter $\mu>0$. So we consider the following system

$$s=Mx+q,\ x\circ s=\mu e,\ x, s\in\text{int}\mathcal{K}. \tag{3}$$

Throughout the paper, we assume that the Cartesian $P_*(\kappa)$-SOCLCP satisfies the interior-point condition (IPC), i.e., there exists $x^0, s^0\in\text{int}\mathcal{K}$ with $s^0=Mx^0+q$, then the system (3) has a unique solution $(x(\mu), s(\mu))$, for each $\mu>0$ as the μ-center of the Cartesian $P_*(\kappa)$-SOCLCP. The set of μ-centers is called the central path of the Cartesian $P_*(\kappa)$-SOCLCP. If $\mu\longrightarrow 0$, then the limit of the central path exists and since the limit points satisfy the complementarity condition, the limit yields a solution for the Cartesian $P_*(\kappa)$-SOCLCP [26]. Similarly to the LO case [4], we replace the standard centering equation $x\circ s=\mu e$ by $\psi(\frac{x\circ s}{\mu})=\psi(e)$, where $\psi(\cdot)$ is the vector-valued function induced by the univariate function $\psi(t)$. Then, we consider the following system

$$s=Mx+q,\ \psi\big(\frac{x\circ s}{\mu}\big)=\psi(e),\ x, s\in\text{int}\mathcal{K}, \tag{4}$$

Applying Newton's method to the system (4) leads to the following system

$$M\Delta x-\Delta s=0,\ \psi\big(\tfrac{x\circ s}{\mu}+\tfrac{x\circ\Delta s+\Delta x\circ s+\Delta x\circ\Delta s}{\mu}\big)=\psi(e). \tag{5}$$

Neglecting the term $\Delta x \circ \Delta s$, from Lemma 2.1, we can replace the second equation of (5) by

$$\psi\big(\frac{x \circ s}{\mu}\big) + \psi'\big(\frac{x \circ s}{\mu}\big) \circ \big(\frac{x \circ \Delta s + \Delta x \circ s}{\mu}\big) = \psi(e).$$

This enables us to rewrite the system (5) as follows

$$M\Delta x - \Delta s = 0,\ x \circ \Delta s + s \circ \Delta x = \mu\big(\psi'(\tfrac{x\circ s}{\mu})\big)^{-1}\big(\psi(e) - \psi(\tfrac{x\circ s}{\mu})\big). \tag{6}$$

Due to the fact that x and s do not operator commute in general, i.e., $L(x)L(s) \neq L(s)L(x)$, the above system does not always have a unique solution. To overcome this difficulty, the second equation of the system (4) is replaced by the following equivalent scaled equation (cf. Lemma 28 in [19])

$$\psi\Big(\frac{P(w)^{-\frac{1}{2}}x \circ P(w)^{\frac{1}{2}}s}{\mu}\Big) = \psi(e),$$

where $w = P(x)^{\frac{1}{2}}\Big(P(x)^{\frac{1}{2}}s\Big)^{-\frac{1}{2}}\Big[= P(s)^{-\frac{1}{2}}\Big(P(s)^{\frac{1}{2}}x\Big)^{\frac{1}{2}}\Big]$ is the NT-scaling point of x and s. This scaling point was first proposed by Nesterov and Todd for self-scaled cones [16]. Now, we replace the second equation of the system (5) by

$$\psi\Big(\frac{P(w)^{-\frac{1}{2}}(x + \Delta x) \circ P(w)^{\frac{1}{2}}(s + \Delta s)}{\mu}\Big) = \psi(e).$$

Applying Newton's method again and neglecting the term $P(w)^{-\frac{1}{2}}\Delta x \circ P(w)^{\frac{1}{2}}\Delta s$, from Lemma 2.1, we get

$$\begin{aligned} M\Delta x - \Delta s &= 0,\\ P(w)^{\frac{1}{2}}s \circ P(w)^{-\frac{1}{2}}\Delta x + P(w)^{-\frac{1}{2}}x \circ P(w)^{\frac{1}{2}}\Delta s &= \\ \mu\big(\psi'(\tfrac{P(w)^{-\frac{1}{2}}x\circ P(w)^{\frac{1}{2}}s}{\mu})\big)^{-1}\big(\psi(e) - \psi(\tfrac{P(w)^{-\frac{1}{2}}x\circ P(w)^{\frac{1}{2}}s}{\mu})\big).& \end{aligned} \tag{7}$$

In this case, assuming that $\psi(t) = \sqrt{t}$, the system (7) becomes

$$\begin{aligned} \Delta s - M\Delta x &= 0,\\ P(w)^{-\frac{1}{2}}x \circ P(w)^{\frac{1}{2}}\Delta s + P(w)^{\frac{1}{2}}s \circ P(w)^{-\frac{1}{2}}\Delta x &= \\ 2\Big(\big(\mu P(w)^{-\frac{1}{2}}x \circ P(w)^{\frac{1}{2}}s\big)^{\frac{1}{2}} - P(w)^{-\frac{1}{2}}x \circ P(w)^{\frac{1}{2}}s\Big).& \end{aligned} \tag{8}$$

We use the following notations:

$$v := \frac{P(w)^{-\frac{1}{2}}x}{\sqrt{\mu}}\ \Big[= \frac{P(w)^{\frac{1}{2}}s}{\sqrt{\mu}}\Big],\ d_x := \frac{P(w)^{-\frac{1}{2}}\Delta x}{\sqrt{\mu}},\ d_s := \frac{P(w)^{\frac{1}{2}}\Delta s}{\sqrt{\mu}}. \tag{9}$$

It follows from (9) that the system (8) reduces to

$$\overline{M}d_x - d_s = 0,\ d_x + d_s = p_v, \tag{10}$$

where $\overline{M} := P(w)^{\frac{1}{2}} M P(w)^{\frac{1}{2}}$ and $p_v := 2(e - v)$. The new search directions d_x and d_s are obtained by solving (10) so that Δx and Δs are computed via (9). The new iterates are given by

$$\tilde{x} := x + \Delta x, \ \tilde{s} := s + \Delta s. \tag{11}$$

For the analysis of the algorithm, we define a norm-based proximity measure

$$\delta(v) := \delta(x, s; \mu) := \frac{\|p_v\|_F}{2} = \|e - v\|_F. \tag{12}$$

Defining $q_v = d_x - d_s$, we have

$$d_x = \frac{p_v + q_v}{2}, \ d_s = \frac{p_v - q_v}{2}, \ d_x \circ d_s = \frac{p_v \circ p_v - q_v \circ q_v}{4}. \tag{13}$$

Moreover, since $\overline{M}$ has the Cartesian $P_*(\kappa)$-property and $d_s = \overline{M} d_x$ from the first equation in (10), we obtain

$$\begin{aligned}
\langle d_x, d_s \rangle &\geq -4\kappa \sum_{j \in I_+} \langle d_x^{(j)}, d_s^{(j)} \rangle \geq -\kappa \sum_{j \in I_+} \langle d_x^{(j)} + d_s^{(j)}, d_x^{(j)} + d_s^{(j)} \rangle \\
&\geq -\kappa \sum_{j=1}^{N} \langle d_x^{(j)} + d_s^{(j)}, d_x^{(j)} + d_s^{(j)} \rangle = -\kappa \sum_{j=1}^{N} \left\| d_x^{(j)} + d_s^{(j)} \right\|_F^2 \\
&= -\kappa \left\| d_x + d_s \right\|_F^2 = -\kappa \left\| p_v \right\|_F^2 = -4\kappa\delta^2. \qquad (14)
\end{aligned}$$

This implies that

$$\|q_v\|_F^2 = \|p_v\|_F^2 - 4\langle d_x, d_s \rangle \leq 4\delta^2 + 16\kappa\delta^2 = 4(1 + 4\kappa)\delta^2. \tag{15}$$

Using (9) and (11), we obtain

$$\tilde{x} = x + \Delta x = \sqrt{\mu} P(w)^{\frac{1}{2}}(v + d_x), \ \tilde{s} = s + \Delta s = \sqrt{\mu} P(w)^{-\frac{1}{2}}(v + d_s). \tag{16}$$

Since $P(w)^{\frac{1}{2}}$ and its inverse $P(w)^{-\frac{1}{2}}$ are automorphisms of $\mathcal{K}$, then $\tilde{x}$ and $\tilde{s}$ belong to int$\mathcal{K}$ if and only if $v + d_x$ and $v + d_s$ belong to int$\mathcal{K}$, respectively.

Lemma 2.5. *Let $\delta(x, s; \mu) < \frac{1}{\sqrt{1+4\kappa}}$. Then $\tilde{x}$ and $\tilde{s}$ are strictly feasible.*

Proof. Using (15) and an argument similar to that described in the proof of lemma 4.2 [23], the result follows. □

According to (9), the v-vector after the step is given by

$$\tilde{v} := \frac{P((\tilde{w})^{-\frac{1}{2}})\tilde{x}}{\sqrt{\mu}} \ \Big[= \frac{P((\tilde{w})^{\frac{1}{2}})\tilde{s}}{\sqrt{\mu}} \Big],$$

where $\tilde{w}$ is the NT-scaling point of $\tilde{x}$ and $\tilde{s}$.

Lemma 2.6. (Proposition 5.9.3 in [21]) *One has $\tilde{v} \sim \left(P(v + d_x)^{\frac{1}{2}}(v + d_s)\right)^{\frac{1}{2}}$.*

Lemma 2.7. *Let $\delta := \delta(x, s; \mu)$. Then $\lambda_{\min}(\tilde{v}) \geq \sqrt{1 - (1 + 4\kappa)\delta^2}$.*

Proof. From Lemma 2.6, Lemma 2.3 and (15) it follows that

$$\begin{aligned}\lambda_{\min}(\tilde{v}) &= \lambda_{\min}\big((P(v + d_x)^{\frac{1}{2}}(v + d_s)\big)^{\frac{1}{2}}\big) \geq \Big(\lambda_{\min}\big((v + d_x) \circ (v + d_s)\big)\Big)^{\frac{1}{2}} \\ &= \Big(\lambda_{\min}\big(e - \frac{q_v \circ q_v}{4}\big)\Big)^{\frac{1}{2}} \geq \Big(1 - \big\|\frac{q_v \circ q_v}{4}\big\|_F\Big)^{\frac{1}{2}} \geq \Big(1 - \frac{\|q_v\|_F^2}{4}\Big)^{\frac{1}{2}} \\ &\geq \sqrt{1 - (1 + 4\kappa)\delta^2}.\end{aligned}$$

This completes the proof. □

Lemma 2.8. *Let $\delta := \delta(x, s; \mu) < \frac{1}{\sqrt{1+4\kappa}}$. Then*

$$\delta(\tilde{x}, \tilde{s}; \mu) \leq \frac{(1 + 4\kappa)\delta^2}{1 + \sqrt{1 - (1 + 4\kappa)\delta^2}}.$$

Proof. Using Lemma 2.7, (15) and an argument similar to that described in the proof of lemma 4.4 [23], the result follows. □

3. Full NT-step IIPM

3.1. **The perturbed problem.** As usually of IIPMs, we assume that the Cartesian $P_*(\kappa)$-SOCLCP (1) has a solution (x^*, s^*) such that

$$\|x^*\|_\infty \leq \rho_p, \ \|s^*\|_\infty \leq \rho_d, \tag{17}$$

where ρ_p and ρ_d are positive. Furthermore, we define

$$x^0 = \rho_p e, \ s^0 = \rho_d e, \ \mu^0 = \rho_p \rho_d, \tag{18}$$

as the initial starting point. Then, the initial residual as is given $r_q^0 = s^0 - Mx^0 - q$. For any ν with $0 < \nu \leq 1$, we consider the perturbed problem to be

$$s - Mx - q = \nu r_q^0, \ x, s \in \mathcal{K}. \tag{19}$$

Note that if $\nu = 1$, then $(x, s) = (x^0, s^0)$ yields a strictly feasible solution of (19). We conclude that if $\nu = 1$, then (19) satisfies the IPC. More generally, we have the following result.

Lemma 3.1. *Let the Cartesian $P_*(\kappa)$-SOCLCP be feasible and $0 < \nu \leq 1$. Then, the perturbed problem (19) satisfies the IPC.*

Proof. The proof is similar to the proof of Lemma 17 in [11]. □

Let the Cartesian $P_*(\kappa)$-SOCLCP be feasible and $0 < \nu \leq 1$. Lemma 3.1 implies that the perturbed problem (19) satisfies the IPC, for each $0 < \nu \leq 1$, and hence its central path exists. This means that the system

$$s - Mx - q = \nu r_q^0, \ x \circ s = \mu e, \ x, s \in \mathcal{K}, \tag{20}$$

has a unique solution, for every $\mu > 0$. It is the μ-center of the perturbed problem (19). In the sequel, the parameters μ and ν always satisfy the relation $\mu = \mu^0\nu$. The system (20), can be written as follows:

$$s - Mx - q = \nu r_q^0,\ \psi(\frac{x \circ s}{\mu}) = \psi(e),\ x, s \in \mathcal{K}. \tag{21}$$

We assume that (x, s) is a strictly feasible solution of (19). We apply Newton's approach for (21). In fact, we want the new iterates $x + \Delta x$ and $s + \Delta s$ such that

$$\begin{gathered} s + \Delta s - M(x + \Delta x) - q = \nu r_q^0, \\ \psi(\tfrac{x\circ s}{\mu} + \tfrac{x\circ\Delta s + \Delta x\circ s + \Delta x\circ\Delta s}{\mu}) = \psi(e), \\ x + \Delta x, s + \Delta s \in \mathcal{K}. \end{gathered}$$

Neglecting the quadratic term $\Delta x \circ \Delta s$ and using Lemma 2.1, since $s - Mx - q = \nu r_q^0$, we obtain

$$\begin{gathered} \Delta s - M\Delta x = 0, \\ x \circ \Delta s + s \circ \Delta x = \mu\big(\psi'(\tfrac{x\circ s}{\mu})\big)^{-1} \circ \big(\psi(e) - \psi(\tfrac{x\circ s}{\mu})\big). \end{gathered} \tag{22}$$

3.2. **A new algorithm.** Initially, we have $\delta(x^0, s^0; \mu^0) = 0$. In what follows, we assume that at the start of each iteration, just before the μ-update, $\delta(x, s; \mu) \le \tau$. So, this is certainly true at the start of the first iteration. Now suppose that the iterate (x, s) is strictly feasible of (19) for $\mu = \nu\mu^0$ and such that $\delta(x, s; \mu) \le \tau$. We reduce μ to $\mu^+ = (1 - \theta)\mu$ and ν to $\nu^+ = (1 - \theta)\nu$, with $\theta \in (0, 1)$, and find displacements $\Delta^f x$ and $\Delta^f s$ such that

$$\begin{gathered} M\Delta^f x - \Delta^f s = \theta\nu r_q^0, \\ P(w)^{\frac{1}{2}} s \circ P(w)^{-\frac{1}{2}}\Delta^f x + P(w)^{-\frac{1}{2}} x \circ P(w)^{\frac{1}{2}}\Delta^f s = \\ 2\Big(\big(\mu P(w)^{-\frac{1}{2}} x \circ P(w)^{\frac{1}{2}} s\big)^{\frac{1}{2}} - P(w)^{-\frac{1}{2}} x \circ P(w)^{\frac{1}{2}} s\Big), \end{gathered} \tag{23}$$

where w is the NT-scaling point of x and s. It is easily seen that $x^f := x + \Delta^f x$ and $s^f := s + \Delta^f s$ satisfy the affine equation in (19), with $\nu = \nu^+$. Then, just by performing a centering step starting at (x^f, s^f) and targeting at μ^+-center of (19) with $\nu = \nu^+$, we obtain iterates (x^+, s^+) that are strictly feasible for (19) with $\nu = \nu^+$ and $\delta(x^+, s^+; \mu^+) \le \tau$. We define

$$d_x^f := \frac{P(w)^{-\frac{1}{2}}\Delta^f x}{\sqrt{\mu}},\ d_s^f := \frac{P(w)^{\frac{1}{2}}\Delta^f s}{\sqrt{\mu}}. \tag{24}$$

One can easily check that the system (23), which defines the search directions $\Delta^f x$ and $\Delta^f s$, can be written in terms of the scaled search directions d_x^f and

d_s^f as follows

$$\overline{M}d_x^f - d_s^f = \frac{\theta\nu}{\sqrt{\mu}}P(w^{\frac{1}{2}})r_q^0,\ d_x^f + d_s^f = p_v, \tag{25}$$

where $\overline{M} := P(w)^{\frac{1}{2}}MP(w)^{\frac{1}{2}}$ and $p_v := 2(e-v)$. Let $\tilde{p}_v := d_x^f - d_s^f$. Then, we have

$$d_x^f \circ d_s^f = \frac{p_v \circ p_v - \tilde{p}_v \circ \tilde{p}_v}{4}, \tag{26}$$

which implies that

$$\frac{\|\tilde{p}_v\|_F^2}{4} = \frac{\|p_v\|_F^2}{4} - \langle d_x^f, d_s^f\rangle. \tag{27}$$

A formal description of the algorithm is given as follows.

Infeasible interior − point algorithm

Input :
accuracy parameter $\epsilon > 0$;
barraier update parameter $\theta,\ 0 < \theta < 1$;
threshold parameter $\tau > 0$;
initialization parameters $\rho_{\rm p} > 0,\ \rho_{\rm d} > 0$.

begin
$x := \rho_p e;\ s := \rho_d e;\ \mu^0 := \rho_p\rho_d$;
while $\max\left(\mu N, \nu\|r_q^0\|_F\right) > \epsilon$
$(x,s) := (x,s) + (\Delta^f x, \Delta^f s)$;
μ and ν − update :
$\mu := (1-\theta)\mu;\ \nu := (1-\theta)\nu$;
$(x,s) := (x,s) + (\Delta x, \Delta s)$;
end while
end.

3.3. **Analysis of the algorithm.** Let $x^f = x + \Delta^f x$ and $s^f = s + \Delta^f s$ be the iterates obtained after the feasibility step. Then, by using (24), we have

$$x^f = \sqrt{\mu}P(w)^{\frac{1}{2}}(v + d_x^f),\ s^f = \sqrt{\mu}P(w)^{-\frac{1}{2}}(v + d_s^f).$$

Since $P(w)^{\frac{1}{2}}$ and its inverse $P(w)^{-\frac{1}{2}}$ are automorphisms of $\mathcal{K}$, the iterates x^f and s^f belong to int$\mathcal{K}$ if and only if $v + d_x^f$ and $v + d_s^f$ belong to int$\mathcal{K}$, respectively. Moreover, $p_v = 2(e-v)$ implies that

$$v \circ v + v \circ p_v = e - \frac{1}{4}p_v \circ p_v. \tag{28}$$

In what follows, we use the notation $\bar{\omega} := \frac{1}{2}\sqrt{\|d_x^f\|_F^2 + \|d_s^f\|_F^2}$. In the next lemma we give a condition in terms of $\delta(v)$ and $\bar{\omega}$, which guarantees the feasibility of x^f and s^f.

Lemma 3.2. *The iterate (x^f, s^f) is strictly feasible if $\delta(v)^2 + 2\bar{\omega}^2 < 1$.*

Proof. We define $v_x(\alpha) := v + \alpha d_x^f$ and $v_s(\alpha) := v + \alpha d_s^f$, for $0 \leq \alpha \leq 1$. We thus have

$$\begin{aligned} v_x(\alpha) \circ v_s(\alpha) &= v^2 + \alpha v \circ (d_x^f + d_s^f) + \alpha^2 d_x^f \circ d_s^f \\ &= (1-\alpha)v^2 + \alpha(v^2 + v \circ p_v) + \alpha^2\big(\frac{p_v \circ p_v - \tilde{p}_v \circ \tilde{p}_v}{4}\big) \\ &= (1-\alpha)v^2 + \alpha\big(e - (1-\alpha)\frac{p_v \circ p_v}{4} - \alpha\frac{\tilde{p}_v \circ \tilde{p}_v}{4}\big). \end{aligned}$$

It follows that $v_x(\alpha) \circ v_s(\alpha) \in \text{int}\mathcal{K}$ holds if

$$\big\|(1-\alpha)\frac{p_v \circ p_v}{4} + \alpha\frac{\tilde{p}_v \circ \tilde{p}_v}{4}\big\|_F < 1.$$

Using the triangle inequality and (27) we obtain

$$\begin{aligned} \big\|(1-\alpha)\frac{p_v \circ p_v}{4} + \alpha\frac{\tilde{p}_v \circ \tilde{p}_v}{4}\big\|_F &\leq (1-\alpha)\big\|\frac{p_v \circ p_v}{4}\big\|_F + \alpha\big\|\frac{\tilde{p}_v \circ \tilde{p}_v}{4}\big\|_F \\ &\leq (1-\alpha)\frac{\|p_v\|_F^2}{4} + \alpha\frac{\|\tilde{p}_v\|_F^2}{4} = \delta(v)^2 - \alpha\langle d_x^f, d_s^f\rangle \leq \delta(v)^2 + 2\bar{\omega}^2, \end{aligned}$$

where the last inequality follows due to $0 < \alpha \leq 1$ and the following inequality

$$-\langle d_x^f, d_s^f\rangle \leq |\langle d_x^f, d_s^f\rangle| \leq \|d_x^f\|_F \|d_s^f\|_F \leq \frac{1}{2}\big(\|d_x^f\|_F^2 + \|d_s^f\|_F^2\big) = 2\bar{\omega}^2.$$

Therefore, the assumption $\delta(v)^2 + 2\bar{\omega}^2 < 1$ implies that $v_x(\alpha) \circ v_s(\alpha) \in \text{int}\mathcal{K}$ for $0 \leq \alpha \leq 1$. Hence, since $x, s \in \text{int}\mathcal{K}$, Lemma 2.2 implies that $v_x(1) = v + d_x^f \in \text{int}\mathcal{K}$ and $v_s(1) = v + d_s^f \in \text{int}\mathcal{K}$. This completes the proof. □

Let

$$v^f := \frac{P((w^f)^{-\frac{1}{2}})x^f}{\sqrt{\mu^+}}\Big[= \frac{P((w^f)^{\frac{1}{2}})s^f}{\sqrt{\mu^+}}\Big],$$

where w^f is the NT-scaling point of x^f and s^f. In the sequel, we denote $\delta(x^f, s^f; \mu^+)$ shortly by $\delta(v^f)$.

Lemma 3.3. *If $\delta(v)^2 + 2\bar{\omega}^2 < 1$. Then*

$$\delta(v^f) \leq \frac{\delta(v)^2 + 2\bar{\omega}^2 + \theta\sqrt{2N}}{1 - \theta + \sqrt{(1-\theta)(1 - \delta(v)^2 - 2\bar{\omega}^2)}}.$$

Proof. The proof of the lemma is similar to the proof of Lemma 12 in [10]. □

Lemma 3.4. *Let $\delta(v) \leq \tau < 1$. Then $1-\tau \leq \lambda_i(v) \leq 1+\tau, i = 1, \ldots, 2N$.*

Proof. From $\delta(v) = \|e - v\|_F \leq \tau$, we obtain

$$(\lambda_i(v) - 1)^2 \leq \|v - e\|_F^2 \leq \tau^2, i = 1, \ldots, 2N.$$

This implies the desired result. □

Lemma 3.5. *If* SOCLCP *is the Cartesian $P_*(\kappa)$-property, then for any $a, \tilde{b}$ the linear system*

$$-\overline{M}d_x^f + d_s^f = \tilde{b},\ d_x^f + d_s^f = a, \tag{29}$$

has a unique solution (d_x^f, d_s^f) and the following inequality is satisfied:

$$\left\|(d_x^f, d_s^f)\right\|_F \leq \sqrt{1+2\kappa}\|a\|_F + \big(1+\sqrt{2+4\kappa}\big)\eta(\tilde{b}),$$

where

$$\eta(\tilde{b})^2 = \min\left\{\left\|(\tilde{d_x^f}, \tilde{d_s^f})\right\|_F^2 : -\overline{M}\tilde{d_x^f} + \tilde{d_s^f} = \tilde{b}\right\} = \tilde{b}^T(\overline{M}P(w)^{-1}\overline{M}^T + P(w))^{-1}\tilde{b}.$$

Proof. The proof of the lemma is similar to the proof of Lemma 3.3 in [13], and is therefore omitted. □

Comparing system (29) with the system (25) and considering $a = p_v$ and $\tilde{b} = -\frac{\theta\nu}{\sqrt{\mu}}P(w^{\frac{1}{2}})r_q^0$ in the system (29), we have

$$\begin{aligned}\|d_x^f\|_F^2 + \|d_s^f\|_F^2 &\leq \left(\sqrt{1+2\kappa}\|p_v\|_F + \big(1+\sqrt{2+4\kappa}\big)\frac{\theta\nu}{\sqrt{\mu}}\eta\big(-P(w^{\frac{1}{2}})r_q^0\big)\right)^2 \\ &\leq \left(2\sqrt{1+2\kappa}\tau + \big(1+\sqrt{2+4\kappa}\big)\frac{\theta\nu}{\sqrt{\mu}}\eta(-P(w^{\frac{1}{2}})r_q^0)\right)^2. \qquad (30)\end{aligned}$$

Let (x^*, s^*) be the optimal solution of the Cartesian $P_*(\kappa)$-SOCLCP that satisfies (17) and the algorithm starts with $(x^0, s^0) = (\rho_p e, \rho_d e)$. Then,

$$x^* - x^0 \preceq_{\mathcal{K}} \rho_p e,\ s^* - s^0 \preceq_{\mathcal{K}} \rho_d e, \tag{31}$$

$$\begin{aligned}-P(w^{\frac{1}{2}})r_q^0 &= -P(w^{\frac{1}{2}})(s^0 - Mx^0 - q) \\ &= -P(w^{\frac{1}{2}})MP(w^{\frac{1}{2}})P(w^{-\frac{1}{2}})(x^* - x^0) + P(w^{\frac{1}{2}})(s^* - s^0) \\ &= -\overline{M}P(w^{-\frac{1}{2}})(x^* - x^0) + P(w^{\frac{1}{2}})(s^* - s^0). \qquad (32)\end{aligned}$$

Now, by using the definition of $\eta(-P(w^{\frac{1}{2}})r_q^0)$, (31) and (32), we have

$$\begin{aligned}\eta(-P(w^{\frac{1}{2}})r_q^0)^2 &\leq \left\|P(w^{-\frac{1}{2}})(x^* - x^0)\right\|_F^2 + \left\|P(w^{\frac{1}{2}})(s^* - s^0)\right\|_F^2 \\ &\leq \rho_p^2\mathrm{tr}(w^{-2}) + \rho_d^2\mathrm{tr}(w^2) \leq \rho_p^2\frac{\mathrm{tr}(s^2)}{\mu\lambda_{\min}(v)^2} + \rho_d^2\frac{\mathrm{tr}(x^2)}{\mu\lambda_{\min}(v)^2} \\ &\leq \rho_p^2\frac{\mathrm{tr}(s)^2}{\mu(1-\tau)^2} + \rho_d^2\frac{\mathrm{tr}(x)^2}{\mu(1-\tau)^2}. \qquad (33)\end{aligned}$$

The third inequality follows by Lemma 4.5 in [8] and the last inequality follows by (12) and $\mathrm{tr}(z^2) \le \mathrm{tr}(z)^2$ for each $z \in \mathcal{K}$.

Lemma 3.6. *Let (x,s) be feasible for the perturbed problem (19) and let $(x^0,s^0) = (\rho_p e, \rho_d e)$ and (x^*,s^*) be as defined in (17). Then,*

$$\mathrm{tr}(x) \le 2N(1+4\kappa)\rho_p(2+(1+\tau)^2),\ \ \mathrm{tr}(s) \le 2N(1+4\kappa)\rho_d(2+(1+\tau)^2).$$

Proof. It is easily seen that

$$\nu s^0 + (1-\nu)s^* - s = M(\nu x^0 + (1-\nu)x^* - x).$$

From the Cartesian $P_*(\kappa)$ property of M, we get

$$\begin{aligned}
&\langle \nu x^0 + (1-\nu)x^* - x, \nu s^0 + (1-\nu)s^* - s\rangle \\
&\qquad \ge -4\kappa \sum_{j\in I_+} \Big(\langle \nu x_j^0 + (1-\nu)x_j^* - x_j, \nu s_j^0 + (1-\nu)s_j^* - s_j\rangle \Big) \\
&\qquad \ge -4\kappa \sum_{j\in I_+} \Big(\nu^2\langle x_j^0, s_j^0\rangle + \nu(1-\nu)\big(\langle x_j^0, s_j^*\rangle + \langle x_j^*, s_j^0\rangle\big) + \langle x_j, s_j\rangle \Big), \\
&\qquad \ge -4\kappa \sum_{j=1}^{N} \Big(\nu^2\langle x_j^0, s_j^0\rangle + \nu(1-\nu)\big(\langle x_j^0, s_j^*\rangle + \langle x_j^*, s_j^0\rangle\big) + \langle x_j, s_j\rangle \Big), \qquad (34)
\end{aligned}$$

where the second inequality follows by $\langle x^0, s\rangle + \langle x, s^0\rangle \ge 0$, $\langle x^*, s\rangle + \langle x, s^*\rangle \ge 0$ and $\langle x^*, s^*\rangle = 0$. By rearranging the above inequality and using $x^0 = \rho_p e, s^0 = \rho_d e, \|x^*\|_\infty \le \rho_p, \|s^*\|_\infty \le \rho_d$ and $\langle x, s\rangle = \mu\langle v, v\rangle \le 2N\mu(1+\tau)^2$, we obtain

$$\begin{aligned}
\langle x^0, s\rangle + \langle x, s^0\rangle &\le (1+4\kappa)\Big(\nu\langle x^0, s^0\rangle + (1-\nu)\big(\langle x^0, s^*\rangle + \langle x^*, s^0\rangle\big) + \frac{1}{\nu}\langle x, s\rangle\Big) \\
&\le (1+4\kappa)\Big(2N\nu\rho_p\rho_d + 4N(1-\nu)\rho_p\rho_d + 2N\rho_p\rho_d(1+\tau)^2\Big) \\
&\le (1+4\kappa)2N\rho_p\rho_d\big(2+(1+\tau)^2\big).
\end{aligned}$$

Therefore, $\langle x, s^0\rangle \le (1+4\kappa)2N\rho_p\rho_d\big(2+(1+\tau)^2\big)$ which implies the result. □
Using Lemma 3.6, (33), (30) and $\mu = \nu\rho_p\rho_d$, we obtain

$$\|d_x^f\|_F^2 + \|d_s^f\|_F^2 \le \Big(2\sqrt{1+2\kappa}\tau + 2\sqrt{2}N\theta(1+4\kappa)\big(1+\sqrt{2+4\kappa}\big)\frac{2+(1+\tau)^2}{1-\tau}\Big)^2.$$

Therefore, by the definition of $\bar\omega$, we get

$$\bar\omega \le \sqrt{1+2\kappa}\tau + \sqrt{2}N\theta(1+4\kappa)\big(1+\sqrt{2+4\kappa}\big)\frac{2+(1+\tau)^2}{1-\tau}. \qquad (35)$$

In this stage, we choose $\tau = \frac{1}{16(1+4\kappa)}$ and $\theta = \frac{1}{27N(1+4\kappa)^2}$. From (35) it follows that $\bar\omega < \frac{1}{2\sqrt{1+4\kappa}}$. Moreover, $\delta(v)^2 + 2\bar\omega^2 < \frac{1}{256(1+4\kappa)^2} + \frac{1}{2(1+4\kappa)} < 1$, which

implies that the iterate (x^f, s^f) is a strictly feasible solution of (19) with $\nu = \nu^+$. The next lemma gives an upper bound for $\delta(v^f)$.

Lemma 3.7. *Let $\delta(v) \leq \tau$. Then, $\delta(v^f) < \frac{0.3363}{1+4\kappa}$.*

Proof. From Lemma 3.3 we have

$$\begin{aligned}\delta(v^f) &\leq \frac{\delta(v)^2 + 2\bar{\omega}^2 + \theta\sqrt{2N}}{1-\theta+\sqrt{(1-\theta)(1-\delta(v)^2-2\bar{\omega}^2)}} \\ &\leq \frac{\tau^2 + 2\bar{\omega}^2 + \theta\sqrt{2N}}{1-\theta+\sqrt{(1-\theta)(1-\tau^2-2\bar{\omega}^2)}}.\end{aligned}$$

Now, using $\tau = \frac{1}{16(1+4\kappa)}, \theta = \frac{1}{27N(1+4\kappa)^2}$ and $\bar{\omega} < \frac{1}{2\sqrt{1+4\kappa}}$, we get

$$\begin{aligned}\delta(v^f) &< \frac{(\frac{1}{16(1+4\kappa)})^2 + 2(\frac{1}{2\sqrt{1+4\kappa}})^2 + \frac{\sqrt{2N}}{27N(1+4\kappa)^2}}{1-\frac{1}{27N(1+4\kappa)^2}+\sqrt{(1-\frac{1}{27N(1+4\kappa)^2})(1-(\frac{1}{16(1+4\kappa)})^2-2(\frac{1}{2\sqrt{1+4\kappa}})^2)}} \\ &\leq \frac{\frac{1}{1+4\kappa}\big(\frac{1}{16^2}+\frac{1}{2}+\frac{\sqrt{2}}{27}\big)}{\frac{26}{27}+\sqrt{\frac{26}{27}\big(1-\frac{1}{16^2}-\frac{1}{2}\big)}} \leq \frac{0.3363}{1+4\kappa}.\end{aligned}$$

This implies the desired result. □

Lemma 3.8. *Let (x^+, s^+) be the iterates obtained by a main iteration of the algorithm and $\delta(v) \leq \tau$. Then $\delta(v^+) := \delta(x^+, s^+; \mu^+) < \frac{1}{16(1+4\kappa)}$.*

Proof. Since the iterate (x^+, s^+) is obtained by a main iteration of the algorithm, thus $x^+ = x^f + \Delta x,\ s^+ = s^f + \Delta s$. Using Lemma 3.7, we have

$$\delta(v^f) < \frac{0.3363}{1+4\kappa} < \frac{1}{\sqrt{1+4\kappa}},$$

which applying Lemma 2.5 for (19) with $\nu = \nu^+$ implies that x^+ and s^+ are strictly feasible. Now, we use Lemma 2.8 for (19) with $\nu = \nu^+$ and we obtain

$$\delta(v^+) \leq \frac{(1+4\kappa)\delta(v^f)^2}{1+\sqrt{1-(1+4\kappa)\delta(v^f)^2}} < \frac{1}{16(1+4\kappa)}.$$

This completes the proof. □

In each main iteration, both the duality gap and the norm of the residual are reduced by the factor $1-\theta$. Hence, the total number of main iterations is bounded above by

$$\frac{1}{\theta}\log\frac{\max\{(x^0)^T s^0, \|r_q^0\|_F\}}{\epsilon}.$$

Since every main iteration consists of two inner iterations, we may state the main result of the paper.

Theorem 1. *If* (1) *has an optimal solution* (x^*, s^*) *such that* $\|x^*\|_\infty \leq \rho_p$ *and* $\|s^*\|_\infty \leq \rho_d$, *for some* $\rho_p, \rho_d > 0$, *then after at most*

$$54N(1+4\kappa)^2 \log \frac{\max\{(x^0)^T s^0, \|r_q^0\|_F\}}{\epsilon}$$

iterations, the algorithm finds an ϵ*-optimal solution of the Cartesian* $P_*(\kappa)$*-SOCLCP.*

4. Conclusions

We proposed and analyzed a new full Nesterov-Todd step infeasible interior-point method for the Cartesian $P_*(\kappa)$-SOCLCP based on the technique introduced in [5]. We have shown that in each iteration the new algorithm needs a feasibility step and one centering step in order to prove that the algorithm is well defined. We derived the complexity bound for the algorithm which coincides with the currently best-known iteration bound for IIPMs.

References

[1] M. Achache, *A new primal-dual path-following method for convex quadratic programming*, Comp. Appl. Math., 25(1)(2006), pp. 97-110.

[2] K. Ahmadi, F. Hasani, B. Kheirfam, *A full-Newton step infeasible interior-point algorithm based on Darvay directions for linear optimization*, J. Math. Model. Algor. Oper. Res., 13(2)(2014), pp. 191-208.

[3] F. Alizadeh, D. Goldfarb, *Second-order cone optimization*, Math. Program., 95 (2003), pp. 3-51.

[4] Z. Darvay, *New interior point algorithms in linear programming*, Adv. Model. Optim., 5(1) (2003), pp. 51-92.

[5] Z. Darvay, I.M. Papp, P.R. Takács, *An infeasible full-Newton step algorithm for linear optimization with one centering step in major iteration*, Studia Univ. Babes-Bolyai, Informatica, LIX(1) (2014), pp. 28-45.

[6] J. Faraut, A. Korányi, *Analysis on symmetric cones*, Oxford University Press, New York, 1994

[7] L. Faybusovich, *Linear systems in Jordan algebras and primal-dual interior-point algorithms*, J. Comput. Appl. Math., 86(1) (1997), pp. 149-175.

[8] G. Gu, M. Zangiabadi, C. Roos, *Full Nesterov-Todd step infeasible interior-point method for symmetric optimization*, European J. Oper. Res., 214(3) (2011), pp. 473-484.

[9] B. Kheirfam, *A new complexity analysis for full-Newton step infeasible interior-point algorithm for horizontal linear complementarity problems*, J. Optim. Theory Appl., 161(3) (2014), pp. 853-869.

[10] B. Kheirfam, *A new infeasible interior-point method based on Darvay's technique for symmetric optimization*, Ann. Oper. Res., 211(1) (2013), pp. 209-224.

[11] B. Kheirfam, N. Mahdavi-Amiri, *A full Nesterov-Todd step infeasible interior-point algorithm for symmetric cone linear complementarity problem*, Bull. Iranian Math. Soc., 40(3) (2014), pp. 541-564.

[12] B. Kheirfam, N. Mahdavi-Amiri, *An infeasible interior-point algorithm based on modified Nesterov and Todd directions for symmetric linear complementarity problem*, Optimization, 64(7) (2015), pp. 1577-1591.

[13] F.A. Potra, J. Stoer, *On a class of superlinearly convergent polynomial time interior point methods for sufficient LCP*, SIAM J. Optim., 20 (2009), pp. 1333-1363.

[14] M. Kojima, N. Megiddo, T. Noma, A. Yoshise, *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*, Lecture Notes in Comput. Sci. 538, Springer-Verlag, Berlin, 1991

[15] Z.Y. Luo, N.H. Xiu, *Solution existence and boundedness of symmetric cone linear complementarity problems with the Cartesian $P_*(\kappa)$-property*, Preprint, Department of Applied Mathematics, Beijing Jiaotong University, 2007

[16] Y.E. Nesterov, M.J. Todd, *Self-scaled barriers and interior-point methods for convex programming*, Math. Oper. Res., 22(1) (1997), pp. 1-42.

[17] C. Roos, *A full-Newton step $O(n)$ infeasible interior-point algorithm for linear optimization*, SIAM J. Optim., 16(4) (2006), pp. 1110-1136.

[18] C. Roos, T. Terlaky, J.-Ph. Vial, *Theory and Algorithms for Linear Optimization. An Interior-Point Approach*, John Wiley & Sons, Chichester, UK, 1997

[19] S.H. Schmieta, F. Alizadeh, *Extension of primal-dual interior-point algorithm to symmetric cones*, Math. Program., 96(3) (2003), pp. 409-438.

[20] J.F. Sturm, *Similarity and other spectral relations for symmetric cones*, Algebra Appl., 312(1-3) (2000), pp. 135-154.

[21] M.V.C. Vieira, *Jordan algebraic approach to symmetric optimization*, Ph.D thesis, Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands, 2007

[22] G.Q. Wang, Y.Q. Bai, *A primal-dual path-following interior-point algorithm for second-order cone optimization with full Nesterov-Todd step*, Appl. Math. Comput., 215(3) (2009), pp. 1047-1061.

[23] G.Q. Wang, Y.Q. Bai, *A new full Nesterov-Todd step primal-dual path-following interior-point algorithm for symmetric optimization*, J. optim. Theory Appl., 154(3) (2012), pp. 966-985.

[24] G.Q. Wang, *A new polynomial interior-point algorithm for the monotone linear complementarity problem over symmetric cones with full NT-steps*, Asia-Pacific J. Oper. Res., 29(2) (2012), 1250015 (20 pages)

[25] G.Q. Wang, D.T. Zhu, *A class of polynomial interior-point algorithm for the Cartesian $P_*(\kappa)$ second-order cone linear complementarity problem*, Nonlinear Anal., 73 (2010), pp. 3705-3722.

[26] A. Yoshise, *Interior point trajectories and a homogeneous model for nonlinear complementarity problems over symmetric cones*, SIAM J. Optim., 17(4) (2006), pp. 1129-1153.

Department of Applied Mathematics, Azarbaijan Shahid Madani University, Tabriz, I.R. Iran

E-mail address: b.kheirfam@azaruniv.edu

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

ALGORITHMIC APPROACH IN REORIENTATION OF COMPARABILITY GRAPHS

SERGIU CATARANCIUC AND NICOLAE GRIGORIU

Abstract. In this article we present methods and algorithms for arcs reorientation in a transitive orientation of a comparability graph. These methods are based on special classes of subgraphs called B-stable subgraphs. A stable subgraph F of the undirected graph $G = (X; U)$ is called B-stable if F has no common vertices with any other stable sugraph M of G or F is proper subgraph of M. Algorithms of the reorientation of arcs are based on the factorization procedure.

1. Introduction

Sorting still remains a very actual problem in combinatorics and computer science. Partially ordered sets sorting is a research field with many valorous results [1], [2]. It is well known that partially ordered set can be presented as a transitive oriented graph [8].

We use the reorientation of arcs in a transitive orientation of the graph related to the poset in order to get rearrangement of elements in a partially ordered set. Methods described in [3] offer solution based on the orientation of one arc in the graph. In this paper we try to describe a method for reorientation of a set of arcs.

This article is organised as follows. In Section 2 we describe stable subgraphs as basics for manipulation of the transitive orientation of the graph. In this section we provide an algorithm for the construction of a B-stable subgraph. In Section 3 we describe a method for reorientation of arcs in a graph such that the resulting orientation is also transitive. We present two approaches: minimal reorientation of arcs in graph and reorientation of a given set of arcs.

Received by the editors: July 28, 2015.

2010 *Mathematics Subject Classification.* 05C17, 05C20.

1998 *CR Categories and Descriptors.* G.2.2 [**Discrete Mathematics**]: Graph Theory – *Graph algorithms*.

Key words and phrases. graph factor, B-stable subgraph, transitively orientable graph.

2. Stable subgraphs

Transitively orientable graphs have been studied based on implication classes [3] and stable subgraphs [9]. In this paper we use the second approach.

Recall that a subgraph F with the set of vertices X_F is called stable subgraph of the graph $G = (X; U)$ [9], $X_F \subset X$ if $\forall x \in X \setminus X_F$ only one of the following relations holds:

(1) $[x, y] \in U_G$, $\forall y \in X_F$;
(2) $[x, y] \notin U_G$, $\forall y \in X_F$.

Definition 1. [4] *Graph $F = (X_F; U_F)$ is called B-stable subgraph of the undirected graph $G = (X; U)$ if F is stable subgraph of G and for every stable subgraph M of G one of the following conditions is satisfied:*

(1) $X_F \cap X_M = \emptyset$;
(2) $X_F \subseteq X_M$.

Theorem 1. [4] *If F is a B-stable subgraph of the graph $G = (X; U)$ and $x \in X_G \setminus X_F$ is a vertex adjacent to the set X_F, then for every transitive orientation $\overrightarrow{G}$ only one of the following relations holds:*

(1) $[x, y] \in \overrightarrow{U_G}, \forall y \in X_F$;
(2) $[y, x] \in \overrightarrow{U_G}, \forall y \in X_F$.

Remark 1. *If F is a stable subgraph of the undirected graph $G = (X; U)$, then for every vertex $x \in X_G \setminus X_F$ so that $[x, y] \in U_G$, where $y \in X_F$, the following relation holds:*

$$deg(x) \geq deg(y). \tag{1}$$

Remark 2. *If F is a B-stable subgraph, then:*

$$deg(x) > deg(y). \tag{2}$$

Let $G = (X; U)$ be a transitively orientable graph, and F a subgraph of it. Suppose that $\overrightarrow{G} = (X; \overrightarrow{U})$ is a transitive orientation of G. We will denote by $\overrightarrow{F} = (X_F; \overrightarrow{U_F})$ the directed subgraph of $\overrightarrow{G}$, defined by the subgraph F. The following relation holds, $\overrightarrow{U}_{\overrightarrow{G}} = \overrightarrow{U}_{\overrightarrow{G} \setminus \overrightarrow{F}} \cup \overrightarrow{U}_{\overrightarrow{F}}$, where $\overrightarrow{U}_{\overrightarrow{G} \setminus \overrightarrow{F}}$ is the set of all arcs of the graph $\overrightarrow{G}$ except for arcs from $\overrightarrow{F}$.

We will say that F is independent transitively orientable subgraph of G if for every transitive orientation $\overrightarrow{F}^*$ of F, the set of arcs $\overrightarrow{U}_{\overrightarrow{G} \setminus \overrightarrow{F}} \cup \overrightarrow{U}_{\overrightarrow{F}^*}$ defines a transitive orientation $\overrightarrow{G}^*$ of the graph G. The independent transitively orientable subgraph F will be called ITO-subgraph.

From the facts mentioned above it means that in any transitive orientation $\overrightarrow{G}$ of the undirected graph $G = (X; U)$ if we change arcs of the subgraph

$\overrightarrow{F}$, which in G is ITO-subgraph, then the resulting orientation will be also transitive.

Lemma 1. *The subgraph F of the transitively orientable graph $G = (X; U)$ is B-stable, if and only if F is ITO-subgraph.*

Proof. ($\Longrightarrow$) Suppose that F is a B-stable subgraph. Based on Theorem 1, for all edges in U_F we have only one of the following relations:

(1) $[x, y] \in \overrightarrow{U_G}, \forall y \in X_F$;
(2) $[y, x] \in \overrightarrow{U_G}, \forall y \in X_F$.

where $x \in X_G \setminus X_F$. So, the orientation of the edges in U_F is not dependent on the orientation of edges of $U_G \setminus U_F$ in a transitive oriented graph $\overrightarrow{G}$. So, F is ITO-subgraph.

($\Longleftarrow$) Suppose that F is ITO-subgraph. We need to prove that F is a B-stable subgraph.

The fact that F is ITO-subgraph implies that for every transitie orientation of F only one of the following relation holds:

(1) $[x, y] \in \overrightarrow{U_G}, \forall y \in X_F$;
(2) $[y, x] \in \overrightarrow{U_G}, \forall y \in X_F$.

where $x \in X_G \setminus X_F$. This property implies that $[x, y] \in U_G$ where $x \in X_F$ and $y \in X_G \setminus X_F$. So, the subgraph F is stable. Let M be a stable subgraph in G, and $X_F \cap X_M \neq \emptyset$. Because F is ITO-subgraph the transitive orientation of the graph A does not have any influence on the orientation of F. This is possible only if F is subgraph of M. So, the graph F is B-stable. □

We present a solution for finding of B-stable subgraph. The task of construction of B-stable subgraph is split in two recursive algorithms. In the first algorithm we find a stable subgraph. In the second algorithm we check if the given subgraph is B-stable. The first found B-stable subgraph is returned. The input graph is presented as an adjacent list. Construction of a stable subgraph is based on the Depth-First-Search algorithm. For each processed vertex a special class $processed(x)$ with Boolean values $TRUE$ or $FALSE$ is attached. The sets E and P that are presented in the *StableSubgraph* procedure which is defined in the second algorithm.

In the *StableSubgraph* function every vertex of the graph G is processed, and the smallest set of vertexes that satisfy the stable sugraph definition is returned.

We have a vertex x that is adjacent to the stable subgraph and a vertex y that is included in the set of vertices of the stable subgraph as input values. We get a set of vertices that defines a stable subgraph in the output of the algorithm.

Algorithm 1 Stable subgraph

```
1: function StableSubgraph(x, y)
2:     for all z ∉ Γ(x) do
3:         if Γ(z) = Γ(x) then
4:             E ← z
5:         end if
6:     end for
7:     if E ≠ ∅ then
8:         E ← y
9:     end if
10:    for all z ∈ Γ(y) do
11:        if processed(z) = FLASE & Γ(z) ∪ {z} ⊆ Γ(x) ∪ {x} then
12:            processed(z) ← TRUE
13:            P ← z
14:            StableSubgraph(x, z)
15:        end if
16:    end for
17:    if E = ∅ then
18:        return P
19:    end if
20:    return E
21: end function
```

Theorem 2. *Construction of the stable subgraph can be done in $O(\Delta)$ time, where Δ is the maximum degree of a vertex.*

Proof. All cycles in the *StableSubgraph*(x, y) function have $\Gamma(x)$ items. Instructions in these cycles run in constant time. It means that for each cycle we have $O(\Gamma(x))$ time. If we chose the maximal degree in graph then time needed for construction of the potential B-stable subgraph is $O(\Delta)$. □

Further, we present an algorithm for construction of a B-stable subgraph based on the *StableSubgraph*(x, y) function, described above. This algorithm is presented by a recursive function $BSS(G)$. As the previous algorithm this function is based on the Depth-First-Search algorithm. We find a stable subgraph in each level of the graph exploration and check if this subgraph is B-stable. The first found B-stable subgraph is returned. The adjacency list of the graph G is considered as input value of the $BSS(G)$ function. Output of the algorithm is a set of vertices that define a B-stable sugraph. Based on remark 1 and 2, main idea of the algorithm is to subtract a vertex from the graph G, and verify if the resulting subgraph is B-stable.

Algorithm 2 B-stable subgraph

```
1: function BSS(G)
2:     if G is not complete graph then
3:         S ← G
4:         SORT(S)
5:         for all x ∈ X_S do
6:             processed(x) ← TRUE
7:             P ← ∅ & G ← ∅
8:             G ← StableSubgraph(x, x)
9:             if G ≠ S then
10:                BSS(G)
11:            end if
12:        end for
13:    end if
14:    return G
15: end function
```

Theorem 3. *Construction of the B-stable subgraph can be done in $O(n\Delta)$ time, where n and Δ are respectively the number and the maximum degree of vertices of the graph G.*

Proof. The algorithm of the processing of the potential B-stable subgraphs uses the recursive procedure $BSS(G)$. This procedure also explores the graph G using the Depth-First-Search technique.

Graph G is sorted in a descending order based on the degree of the vertices. In the procedure $BSS(G)$ each vertex of the graph is explored. So, for each iteration the $StableSubgraph(x, x)$ function is called. As it is proved in the Theorem 2 the $StableSubraph(x, x)$ procedure can be executed in $O(\Delta)$ time. As a result, we obtain a B-stable subgraph in $O(n\Delta)$ time, if we use the $BSS(G)$ procedure, where n is the number of vertices of the graph G and Δ is the maximum degree of a vertex. □

Remark 3. *Because the StableSubgraph algorithm reduces graph dimension on each iteration, and BSS algorithm stops when graph does not support any changes the BSS function terminates in a finite number of iterations.*

Remark 4. *Based on remark 2 and lemma 1 we can observe that algorithm 2, returns a B-stable subgraph or the whole graph.*

As it was mentioned in the lemma 1 a transitively orientable graph can have more than one orientation. In the next section there will be described a method for reorientation of arcs in a given transitive orientation.

3. Reorientation of a comparability graph

Let F_0 be a B-stable subgraph of the G. We denote by G/F_0 the graph obtained from the graph G by the following rules:

(1) the subgraph F_0 is replaced with the vertex x_{F_0};
(2) all edges $[x, z]$, $\forall x \in X_{F_0}$, $z \in X_G \setminus X_{F_0}$, are replaced with the $[x_{F_0}, z]$.

The graph G/F_0 is called the **graph factor** that corresponds to the B-stable subgraph F_0. The operation of obtaining the graph factor G/F_0 is called **factorization** [5].

If the graph $G^1 = G/F_0$ also contains a B-stable subgraph F_1, then we can get a new graph factor G^1/F_1 from G^1 using the factorization procedure. If this graph also contains a B-stable subgraph then we could repeat the same procedure until we get a graph factor that does not contain any B-stable subgraphs. We can obtain a sequence of undirected graphs:

$$G, G^1 = G/F_0, G^2 = G^1/F_1, \ldots, G^k = G^{k-1}/F_{k-1} \tag{3}$$

with the properties:

(1) F_i is a B-stable subgraph in the graph G^i, where $0 \leq i \leq k-1$, (consider that $G^0 = G$);
(2) the graph $G^k = G^{k-1}/F_{k-1}$ does not contain B-stable subgraphs.

The sequence (3) is called **complete sequence of graph factors** of the graph G.

Reconstruction of the transitive orientation gives the solution for many theoretical and practical problems (see [2], [7]). It is very important to set some specific conditions in order to get the new transitive orientation based on the existing one. We can reorient one arc to achieve a new transitive orientation by using implication classes [6]. In many cases it is necessary to consider reorientation of a set of arcs. Further we will use B-stable subgraphs and factorization procedure in order to accomplish this task.

Definition 2. *If $F = (X_F; U_F)$ is a B-stable subgraph of the transitively orientable graph G, then the set of edges U_F is called the internal factor defined by the subgraph F.*

Let F be a B-stable subgraph of the graph G. Internal factor defined by F is denoted as I_F.

Remark 5. *If transitively orientable graph $G = (X; U)$ does not contain any B-stable subgraphs then the set of edges U_G defines the internal factor of the graph G.*

Let F be a B-stable subgraph of the transitively orientable graph G and I_F is an internal factor defined by the subgraph F, then the next remark holds.

Remark 6. *If $[x, y]$ and $[s, t]$ are two arcs that are contained in the internal factor I_F defined by the B-stable subgraph F, then the transitive orientation defined by the arc $[x, y]$ is the same as the transitive orientation defined by the arc $[s, t]$.*

Definition 3. *Let $x \in X_G \setminus X_F$, where F is a B-stable subgraph of the transitively orientable graph G, is a vertex adjacent to the set X_F. Then, the set of edges $[x, y]$, $\forall y \in X_F$ is called the external factor defined by the subgraph F.*

Let F be a B-stable subgraph of the graph G. External factor defined by F is denoted as E_F.

Remark 7. *If F is a B-stable subgraph of graph G, and E_F is an external factor defined by F, then for every transitive orientation $\overrightarrow{G}$, where $x \in X_{E_F}$ and $y \in X_F$ only one of the following relations is satisfied:*

(1) $[x, y] \in E_F$;
(2) $[y, x] \in E_F$.

It means that all arcs in an external factor have the same direction.

3.1. **Minimal reorientation.**

Lemma 2. *If E_{F_i} is an external factor defined by the B-stable subgraph F_i then there is an internal factor I_{F_j} defined by a B-stable subgraph F_j so that $E_{F_i} \subseteq I_{F_j}$, $1 \leq i \leq k-1$, $i+1 \leq j \leq k$.*

Proof. Let x_{F_i} be a vertex obtained in the factorization operation of the graph G/F_i, and $[x_{F_i}, x_s]$ is an edge of the external factor E_{F_i}. If the vertex x_{F_i} is not contained in another B-stable subgraph, then it can be part of the last graph factor in the complete sequence $G, G^1 = G/F_0, G^2 = G^1/F_1, \ldots, G^k = G^{k-1}/F_{k-1}$. By the Remark 7 the edge $[x_{F_i}, x_s]$ is part of the set U_{G/F_k}. □

Lemma 3. *If F_i is a B-stable subgraph of G and $E_{F_i} \subset I_{F_i}$ then E_{F_i} forces the transitive orientation of the internal factor I_{F_i}.*

Next, we present an algorithm for the reorientation of minimal amount of arcs in a given transitive orientation. We use the $MinimalReorientation$ function. We use the complete series of graph factors $G^1, G^2, \ldots, G^k$ and the current transitive orientation $\overrightarrow{G}$ of the graph G as the input values for this function. New transitive orientation $\overrightarrow{G'}$ of the graph G is the output of the $MinimalReorientation$ function.

Theorem 4. *A new transitive orientation of the graph G using the function $MinimalReorientation$ can be done in $O(k\Delta)$ time, where k is the length of the complete sequence of the graph factors and Δ is the maximal degree of a vertex in the graph.*

Algorithm 3 Minimal Reorientation of arcs in a comparability graph

function MinimalReorientation($G^1, G^2, \ldots, G^k, \overrightarrow{G}$)
  $i \leftarrow 1$
  repeat
    $F_i \leftarrow BSS(G_i)$
    $i \leftarrow i + 1$
  until $U_{F_i} <> \emptyset$
  $\overrightarrow{G'} \longleftarrow \overrightarrow{G^i/F_i}$
  Reorientation of the arcs in the $\overrightarrow{F_i}$
  while $i > 0$ **do**
    $i \leftarrow i - 1$
    $\overrightarrow{G'} \longleftarrow \overrightarrow{G^i/F_i}$
  end while
  return $\overrightarrow{G'}$
end function

Next, we present a method for reorientation of given arcs in a transitive orientation of the graph G.

3.2. **Reorientation forced by a given set of arcs.** Let $G = (X; U)$ be a transitively orientable graph, and $\overrightarrow{G} = (X_G; \overrightarrow{U_G})$ is a transitive orientation of it. The direction of arcs in the subset $\overrightarrow{E_G} \subset \overrightarrow{U_G}$, $2 < |\overrightarrow{E_G}| < |\overrightarrow{U_G}|$, needs to be reversed. A new transitive orientation $\overrightarrow{G'} = (X_G; \overrightarrow{U'_G})$ so that $\overleftarrow{E_G} \subset \overrightarrow{U'_G}$, where $\overleftarrow{E_G} = \{[x, y] | [y, x] \in \overrightarrow{E_G}\}$, should be defined.

We can obtain the set $\overleftarrow{E_G}$ by the reversing of the arcs in $\overrightarrow{E_G}$. The resulting orientation is also transitive. Next, we present the necessary steps for the reconstruction of the orientation that contains all arcs from $\overleftarrow{E_G}$.

We need to apply the factorization procedure on the set $\overleftarrow{E_G}$. Let $[x, y]$ be an arc from $\overleftarrow{E_G}$. If $x \in X_F$, where F is a B-stable subgraph, then we replace the arc $[x, y]$ with the resulting arc $[x', y]$ from the factorization procedure of the subgraph F. If the whole arc is part of the factorized subgraph, then this arc is replaced with the new vertex x'.

We describe an algorithm for the reconstruction of the transitive orientation forced by the given set of arcs $\overleftarrow{E_G}$ based on the idea mentioned above.

This algorithm explores the sequence of the complete graph factors. The exploration of the sequence is done in both directions. In the forward exploration a new set $\overleftarrow{E}_{G^i/F_i}$ is attached for each graph factor. When the sequence of the graph factors is explored backwards, then the new transitive orientation

is created based on the set of arcs that forces the orientation. We use the function *ForcedReorientation*. The complete sequence of the graph factors, set $\overleftarrow{E_G}$ and orientation $\overrightarrow{G}$ are considered as the input values for the function. New orientation of the graph is the output of the *ForcedReorientation* procedure.

Algorithm 4 Reorientation of arcs forced by a set of arcs

```
1: function ForcedReorientation(G^1, G^2, ..., G^k, ←E_G, →G)
2:     i ← 1
3:     while →E_{G^i/F_i} <> G^{i-1} do
4:         →E_{G^i/F_i} ← Factorization of →E_{G^{i-1}/F_{i-1}}
5:         i ← i + 1
6:     end while
7:     while i > 0 do
8:         Construction of the ←E_{G^i/F_i}
9:         →G' ← →G^i/F_i'
10:        i ← i − 1
11:    end while
12:    return →G'
13: end function
```

Theorem 5. *The run time of the algorithm used for the reorientation of the transitive orientation forced by a given set of arcs is $O(k\Delta)$, where k is the number of graph factors in the complete sequence and Δ is the maximal degree of a vertex in the graph.*

4. Conclusions

In this paper we presented algorithms for reorientation of arcs in comparability graphs. These algorithms are based on B-stable subgraphs and the factorization procedure. We can describe comparability graphs by using the factorization procedure of the B-stable subgraphs. Also, B-stable subgraphs can be used in calculation of number of the transitive orientations in a comparability graph. We consider two cases for the accomplishment of the comparability graph reorientation: minimal amount of arcs reorientation and reorientation of a specified set of arcs in a given transitive orientation. We proved that both algorithms run in polynomial time.

References

[1] C. Daskalakis, R. M. Karp, E. Mossel, S. Riesenfeld, E. Verbin. *Sorting and Selection in Posets.* CoRR, abs/0707.1532, (2007)

[2] V.A. Evstigneev. *Applications of the graph theory in programming.* Nauka, Moskow, 352 p. 1985. (in Russian).
[3] M. C. Golumbic. *Graph Theory and Perfect Graphs* (Annals of Discrete Mathematics) 57. North-Holland Publishing Co., ISBN: 978-0-444-51530-8, Amsterdam, (2004).
[4] N. Grigoriu. *B-stable subgraphs in undirected graphs.* The third conference of Mathematical Society of the Republic of Moldova, Chişinău, pp. 354–357, (2014).
[5] N.Grigoriu. *Minimal stable subgraphs in undirected graphs.* International conference, Mathematics & Information Technologies: Research and Education(MITRE-2013), Chişinău, pp. 48–49, (2013).
[6] P. Klavk, J. Kratochvl, T. Krawczyk, B. Walczak. *Extending Partial Representations of Function Graphs and Permutation Graphs.* Algorithms ESA 2012. 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings. pp. 671 – 682. ISBN: 978-3-642-33090-2
[7] B. Sands. *Unsolved problems.* Order 1, p. 311 – 313, (1985).
[8] B. Schroeder. *Ordered Sets: An Introduction.* Birkhuser, ISBN-13: 978-0817641283, (2003)
[9] A. A. Zykov. *Fundamentals of graph theory.* Moskow, ISBN: 5-9502-0057-8, (2004), (in Russian).

Faculty of Mathematics and Computer Science, State University of Moldova, Chisinau
E-mail address: {s.cataranciuc|grigoriunicolae}@gmail.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

OBSTACLE RECOGNITION IN TRAFFIC BY ADAPTING THE HOG DESCRIPTOR AND LEARNING IN LAYERS

ROXANA MOCAN AND LAURA DIOŞAN

Abstract. Despite many years of research, obstacle recognition is still a difficult, but very important task. We present a multi-class approach, that extracts from images the Histogram of Oriented Gradients (HOG) based on aspect ratio of Region of Interest (ROI) and use them in a multi-class classification problem. For the learning phase we propose an original approach based on decision trees. Numerical experiments are performed on a benchmark dataset consisting of animal, pedestrian, car and sign (labeled) images captured in outdoor urban environments and indicate that the proposed model is able to improve the performance of the recognition process.

1. Introduction

Nowadays it is necessary to increase the speed to keep up with traffic and that triggers a higher risk for accidents. This risk is also increased by the high number of road users. Statistics show clearly that the number of accidents and casualties (drivers and pedestrians) has got alarming levels (it is necessary to reduce the number of traffic events). The surveillance of pedestrians, cars, motorcycles and animals in traffic is important for increasing safety.

In this field researchers had made great improvements, classifying each category individually, but using a multiclass classification algorithm for most frequent objects in traffic scene can be useful and faster. In many machine learning methods, a binary classifier is easy to construct, while, in most applications, a multiclass classifier is needed.

A classification task with more than two classes where objects belonging to the same class may vary from each other in views or shapes, has an increased difficulty. Usually this type of problem decomposes trivially into a set of unlinked binary problems.

Received by the editors: June 16, 2015.

2010 *Mathematics Subject Classification.* 68T05, 91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Induction*; I.2.6 [**Artificial Intelligence**]: Learning – *Concept learning*.

Key words and phrases. Multiclass classification, HOG, Decision Trees, Boosting.

The most used algorithm is OAA (One Against All). For this method k classifiers are needed (k is the number of classes). In this method you construct the ith classifier using ith class as positive and the rest of the classes as negative, $i \leq$ number of classes. To predict the class that a test sample belongs to, each classifier has to be verified and to take a vote of confidence. The vote of confidence takes the majority answer from each classifier. Having to train all samples for each classifier involves higher computational time and also means that the number of samples will not be balanced because if x is the number of samples from each class, each classifier will get x positive images and $x * (k-1)$ negative images.

The proposed approach wants to improve the results by equilibrating the positive and negative data and reduce training and testing time by making only $k-1$ classifications and k/2 predictions without taking the vote of confidence.

The outline of the paper is as follows. After briefly reviewing related work in Section 2, we present the background of our system in Section 3. Numerical experiments are presented in Section 4, while the conclusions and further work are highlighted in Section 5.

2. Related work

There are great approaches in this field, researchers used learning for binary classification or for multiclass classification and robust feature extractor. In some papers [3] they found that histogram of oriented Gradients (HOG) can be used for pedestrian detection. The utility of boosting [5] was proved by making the weak classifiers to generalize well and improving prediction results by creating a strong multiclass classifier for urban traffic objects. Creating a hierarchical clustering to create a decision tree by merging classes repeatedly [6] and finding most similar classes using Euclidian distance. Support Vector Machine (SVM) have been successfully employed for a variety of different multiclass classification and regression tasks [2]. Recently in [9], [1] methods that use simple appearance based features which also take advantage of the temporal information, are discussed. Chen et al. introduced the e Differences of Histograms of Oriented Gradients (DHoG) feature based on the changes introduced in the HOG descriptor due to rigid and non rigid motion of vehicles and pedestrians.

3. Proposed approach

The purpose of this work is to correctly classify different type of objects from the road assistance field using a mono-camera. To reach this goal we propose a system which has two components, one for image processing and the other for object recognition. The novelty is from the learning part, since

in the image processing part we used HOG descriptors (the literature proving their effectiveness [3]).

3.1. **Processing the images.** Histogram of Oriented gradients is a robust descriptor that can be described as the distribution of the intensity gradients or edge direction. Each image is divided in regions and each region is divided in four cells. The gradients and orientations from a cell are computed and results a histogram with several bins. Usually the cell has a fixed number of pixels, but the number can vary. Concatenating all four histograms will result a vector. To have the entire HOG vector, only concatenate all regions from image.

In most approaches images are resized, to get a fixed size of HOG vector. For example we can resize all images to 128 x 128 pixels and each region get the fixed number of pixels (16 pixels per region), and get 64 regions from each image. From 64 regions * 36 (region size) = 2304 feature vector. But we can get the same size for feature vector without resizing. For example we have an image with a pedestrian and the image size is 64 x 128 pixels, we only set the cell size at 8 columns and 16 rows. The differences between these 2 methods for HOG computation are showed at Numerical Experiments section.

3.2. **Learning in layers.** The proposed method for learning is using the binary decision tree principles. In a decision tree, each node splits the instance space in two sub-spaces according to a rule. In this case, the rule is if an instance is from positive class or negative class.

The proposed method classifies all data in layers. Each layer is attached to a classifier and the decision rule is in which layer to go if the prediction is positive or negative.

The first layer will classify all k classes, but the second and third layer will classify only $k/2$ classes ($k/2$ classes will be classified in second layer and the rest will be classified in third layer). If the number of samples from each class are equal then the positive and negative numbers of samples are equilibrate. Also using this method we don't need any confidence vote that also takes time.

3.3. **Performance measurements.** The performance measurements that could be taken into account in the case of a classification problem are:

- the true positive rate (TPR): number of positive samples that are predicted well / total number of positive samples,
- the false positive rate (FPR): number of negative samples that are predicted as positive / total number of negative samples
- the precision: the percent of relevant classification among the proposed ones;

- F-measure: the weighted harmonic mean of precision and recall. Grater F-measure (the maximal value being 1) signifies a correct and complete classification;
- the accuracy (Acc): (number of positive samples that are predicted well + number of negative samples that are predicted well) / total number of samples).

4. Numerical experiments

4.1. **Data sets.** For this paper we used four classes: animal, pedestrian, car, signs. The training and testing samples are regions of interest from traffic scene images. The images are gathered from internet: Caltech image data base [4], Inria Person Dataset [3] and LISA dataset [7], [8].

The total number of images is 12037 where 4000 for animal, 4000 for pedestrians, 2547 for cars and 1490 for signs.

The training set has 3500 images with animals, 3500 with pedestrians, 2447 with cars and 1390 with signs. The testing set has 500 images with animals, 500 with pedestrians, 100 with cars and 100 with signs. The samples are regions of interest from cropped images from traffic scene images.

4.2. **Image processing.** Image processing algorithms are implemented using OpenCv libraries.

All images are grey and for feature extraction we used HOG algorithm. From each image we extracted 2304 features (64 blocks * 4 * 9 bins per block). Because the regions of interest from images had different aspect ratio (pedestrian ≈ 0.319, animal ≈ 1.347, signs ≈ 1.1508 and car ≈ 2.5) it was necessary to take different sizes of cells to get the same size of HOG vector. We made that just for simply concatenate class matrixes but it showed that computing HOG in this way made one class samples discriminative from other classes.

We also computed HOG with images that are resized to 128 x 128 pixels and the differences are showed in Tables 1 and 3.

4.3. **Learning and testing.** Machine Learning algorithms are implemented using OpenCv libraries.

For learning we used boosting algorithm with 200 binary decision trees as weak classifiers, with max depth = 2.

As training samples we took for first layer two classes and labeled them as positive, and the other two as negative and trained them using boosting. We choose which class goes to positive and negative by observation the data and the images. This layer uses 10837 images. For second layer we took one class from first layer positive and set as positive for second layer too, and the other class from first layer positive as negative for second layer and trained

them with boosting too. It can be seen that the second layer uses only 5947 images ($\approx 1/2$ from all images).

For third layer we took one class from negative samples from first layer and divided in two (positive and negative), and then train with boosting, the same as first and second layer. For this layer are used 4890 images. For more classes the numbers of layers are increasing and number of samples decreasing.

For testing the first layer we took all the test images from all classes and compute the prediction vector. If for an image the prediction from first layer says that is positive, it goes directly to the second layer and compute the second layer prediction, in case the prediction says that is negative that sample goes to third layer and compute the prediction that says from which class is it.

We trained the same samples using OAA creating four models. For us this means, for training, 3500 samples positive and 7337 negative for animal and pedestrian classes, 2447 samples positive and 8390 negative for cars class and 1390 positive and 9447 negative for signs class. Each model results from training using boosting with the same parameters as learning in layers.

4.4. **Experiment 1.** For the first step we proposed to compare our learning approach to OAA for images that are resized. The results (TPR, FPR, Precision, Recall, F-measure for each class, respectively and Global Accuracy and its confidence interval (for a probability of 95%) for all classes) are represented in Table 1. In addition, in Table 2 the training and testing time involving in this experiment are presented.

		Animal	Pedestrain	Car	Road sign
LiL	TPR	0.66	0.73	0.71	0.99
	FPR	0.18	0.16	0.04	0.03
	Precision	0.72	0.76	0.61	0.75
	Recall	0.66	0.73	0.71	0.99
	F-measure	0.69	0.74	0.66	0.99
	Global Acc		0.776±0.023		
OAA	TPR	0.7	0.75	0.72	0.98
	FPR	0.19	0.21	0.04	0
	Precision	0.72	0.71	0.62	1
	Recall	0.7	0.75	0.72	0.98
	F-measure	0.71	0.73	0.66	0.98
	Global Acc		0.714±0.025		

TABLE 1. Comparison of performances obtained by learning in layers and OAA for resized images

The numerical results from Tables 1 and 2 indicate:

- a better performance (Global accuracy) of the proposed approach in comparison with OAA;
- a better TP rate for car and sign classes, but weaker for animal and pedestrian;
- a better FP rate for animal, pedestrian and sign, but weaker for car;
- the training time for proposed approach is improved;
- the testing time is also better.

	LiL		OAA	
	Training	Testing	Training	Testing
All classes	3069.41	0.012	34544.53	4.535

TABLE 2. Comparison of LiL and OAA running time (seconds) for resized images

4.5. **Experiment 2.** For the second step we proposed also to see if the representation alternative where we changed HOG representation instead of changing image size could imrpove the classification performances. The results (TPR, FPR, Precision, Recall, F-measure for each class, respectively and Global Accuracy and its confidence interval, for all classes) are represented in Table 3. Again, we give the training and testing time (see Table 4).

		Animal	Pedestrain	Car	Road sign
LiL	TPR	0.94	0.99	0.82	1
	FPR	0.025	0.011	0.018	0.001
	Precision	0.96	0.98	0.8	0.99
	Recall	0.94	0.99	0.82	1
	F-measure	0.95	0.98	0.81	0.99
	Global Acc	0.94±0.013			
OAA	TPR	0.95	0.98	0.56	0.94
	FPR	0.074	0.001	0.067	0.006
	Precision	0.9	0.99	0.43	0.93
	Recall	0.95	0.98	0.56	0.94
	F-measure	0.92	0.98	0.48	0.93
	Global Acc	0.861±0.019			

TABLE 3. Comparison of learning in layers and OAA rates for unmodified images

Results from Tables 3 and 4 indicate that:

	LiL		OAA	
	Training	Testing	Training	Testing
All classes	2588.5	0.007	2716	4.34

TABLE 4. Comparison of learning in layers and OAA running time (seconds) for unmodified images

- the proposed approach has better general performance than OAA; TP rate is increased for pedestrian, car and sign classes and weaker for animal class;
- FP rate is higher for animal, car and sign classes, but worst for pedestrians; training time for proposed approach is improved; testing time is also higher.

Differences between the proposed method and One Against All can be summarized as follows.

In the training stage:

- Time for training is shorter because in OAA we use for each classifier 10837 samples and in the proposed method we train only in first layer 10837 samples, in second and third the number of training samples is 5947 and 4890.
- In OAA we need 4 classifiers and in the proposed method we need only three.

In the testing stage:

- in OAA we have compare with four models, and in this method only with two.
- we need to take the confidence vote, in the proposed method we don't.
- the disadvantage for this method is the propagation of the error. If a sample is miscalculated in first layer, can't be recovered in the next layers.

5. Conclusions

Differences between the proposed method and One Against All can be summarized as follows.

An important problem was investigated in this paper: obstacle recognition in images. Each image was represented by using HOG descriptors, whose parameters were adapted to the image size. The extracted features were incorporated finally into a classifier based on decision trees in order to construct a decision model that can be utilized in order to label un-seen images. The learning stage is developed on layers, in order to help the classification model.

We have studied how the process of image resizing vs. adapting the HOG's parameters influence the results of multi-class classification. The results obtained by using the adapted descriptor indicate a better performance of the decision model. Furthermore, the special learning approach improve the classification performances (in comparison to the classical OAA model).

As further work we plan to validate our approach by using more data for this classes or other classes, model validation (cross-validation) and to adapt other image descriptors and also to investigate other types of learning strategies.

References

[1] L. B. Chen, R. S. Feris, Y. Zhai, L. M. Brown, and A. Hampapur. An integrated system for moving object classification in surveillance videos. In *Advanced Video and Signal Based Surveillance, 2008. AVSS '08. IEEE Fifth International Conference on*, pages 52–59, 2008.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273, 1995.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Conference on*, pages I: 886–893, 2005.

[4] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311, 2009.

[5] I. Giosan, A.D. Costea, and S. Nedevschi. Urban traffic dense-stereo obstacle classification using boosting over visual codebook features. In *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*, pages 111–116, 2013.

[6] X. Huaitie, S. Fasheng, and L. Yongsheng. Support Vector Machine algorithm based on kernel hierarchical clustering for multiclass classification. In *Electrical and Control Engineering (ICECE), 2010 International Conference on*, pages 2201–2204, 2010.

[7] Andreas Mogelmose, Mohan M. Trivedi, and Thomas B. Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012.

[8] Sayanan Sivaraman and Mohan M. Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):267–276, 2010.

[9] L. Zhang, S. Li, X. Yuan, and S. Xiang. Real-time object classification in video surveillance based on appearance learning. In *Computer Vision and Pattern Recognition, 2007. CVPR 2007. IEEE Conference on*, pages 1–8, 2007.

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

E-mail address: roxanamocan@ymail.com, lauras@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

SOFTWARE DEFECT DETECTION USING SELF-ORGANIZING MAPS

ZSUZSANNA MARIAN, ISTVÁN GERGELY CZIBULA, GABRIELA CZIBULA AND SERGIU SOTOC

Abstract. This paper addresses the problem of software *defect detection*, an important problem which helps to improve the software systems' maintainability and evolution. In order to detect defective entities within a software system, a self-organizing feature map is proposed. The trained map will be able to identify, using unsupervised learning, if a software module is defective or not. We experimentally evaluate our approach on three open-source case studies, also providing a comparison with similar existing approaches. The obtained results emphasize the effectiveness of using self-organizing maps for software defect detection and confirm the potential of our proposal.

1. Introduction

In order to increase the efficiency of quality assurance, *defect detection* tries to identify those modules of a software where errors are present. In many cases there is no time to thoroughly test each module of the software system, and in these cases defect detection methods can help by suggesting which modules should be focused on during testing.

We are proposing in this paper an unsupervised machine learning method based on self-organizing maps for detecting defective entities within software systems. The self-organizing map architecture was previously applied in the literature for defect detection, but using a kind of hybrid approach, where different threshold values for some software metrics were also used [1]. To our knowledge, there is no approach in the search-based software engineering literature similar to ours. The unsupervised model introduced in this paper proved to outperform most of the similar existing approaches, considering the datasets used for evaluation.

Received by the editors: June 3, 2015.

2010 *Mathematics Subject Classification.* 68T05, 62H30.

1998 *CR Categories and Descriptors.* I.2.6[**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.5.3 [**Computing Methodologies**]: Pattern Recognition – *Clustering*.

The rest of the paper is structured as follows. Section 2 presents the fundamentals of self-organizing maps as well as existing similar approaches for software defect detection. Our proposal for identifying software defects using self-organizing feature maps is introduced in Section 3. Section 4 provides an experimental evaluation of our approach, while an analysis of the obtained results and comparison with existing similar work is given in Section 5. Section 6 contains some conclusions of our paper and indicates directions for further improvement.

2. Background

In this section we aim at presenting the main characteristics of self organizing maps as well as similar approaches for software defect detection.

2.1. **Self-organizing maps.** A *self-organizing map* (SOM) [15] is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (usually two-dimensional) representation of the training samples, called a *map* [5]. Self-organizing maps use a neighborhood function to preserve the topological relationships in the input space and are related to the category of *competitive learning* networks. Self-organizing maps are considered in the neural networks literature as the most innovative form of unsupervised learning.

The SOM provides a topology preserving mapping from the multi-dimensional input space to the map neurons (units). Each neuron from the input layer of a SOM is connected to each neuron from the output layer and each connection has an associated weight. The topology preservation property means that a SOM groups similar input instances on neurons that are close on the SOM [9]. The map is usually trained using the Kohonen algorithm [15].

The trained self-organizing map is able to provide clusters of similar data items [12]. This particular characteristic of SOMs makes them appropriate for data mining tasks that involve classification and clustering of data items [12]. The SOM can be used as an effective tool for clustering as well as a tool for visualizing high-dimensional data.

2.2. **Literature review.** A review on unsupervised learning-based approaches existing in the defect prediction literature will be provided in the following.

Abaei et al. proposed in [1] a fault prediction method by utilizing self-organizing maps and thresholds. Two experiments are conducted in this paper: the first one considers the removal of the modules' labels and re-computing them afterwards by taking threshold values into account for some selected attributes (the ones for which threshold values are known). In the second experiment, a SOM is used for both clustering and evaluating the input data.

Threshold values are used as well for labeling the units from the trained SOM. For this second experiment they report a good Overall Error, and in most cases their proposed solution improves classification of unlabeled program modules in terms of FPR (False Positive Rate) and FNR (False Negative Rate). One drawback to their approach is that they do not obtain good results when the dataset is very small.

Another approach is presented in [2] that predicts software fault using a Quad Tree-based K-Means algorithm. The difference to the original K-Means algorithm is that the cluster centers are found by using Quad Trees. They evaluate their approach on various datasets, and compute the FPR, FNR and Overall Error for them. These values indicate that their approach is slightly better than other cluster center initialization techniques and they achieve slightly better results from fewer number of iterations.

The method presented in [18] uses the K-Means clustering algorithm as well, but it uses Hyper Quad Trees for determining the cluster centers. They present that Hyper Quad Trees are more efficient than simple Quad Trees because they produce more accurate centroids. After the K-Means algorithm is run, a threshold value is used to determine which cluster represents the defective and which represents the non-defective entities. The results for some public datasets confirm obtaining better outcomes in terms of FPR and Overall Error when comparing this approach to simple Quad Tree approach.

Tosun et al. used in [17] network measures to identify defective modules in software systems. Their approach uses the Naive Bayes classifier, together with a Call Graph Based Ranking (CGBR) framework. The experimental evaluation was performed on both small and large datasets and for three cases: complexity metrics only, network metrics only and a combination between them. The results show a great performance of applying network metrics for large datasets, but they do not provide significant improvement for small projects.

A clustering-based approach is presented in [3], where the Xmeans algorithm is used, an algorithm which is similar to K-means, but it can automatically determine the optimal number of clusters. The authors use the implementation of this algorithm from WEKA [7], and when the clusters are created, software metric threshold values are applied to the mean vector of each cluster in order to decide whether it represents the defective or the non-defective entities. They claim that this method proved better results than a simple threshold-based approach, fuzzy c-means and k-means.

Another unsupervised software fault prediction model is given by Park and Hong in [14] where clustering algorithms that determine the number of clusters automatically are used. They have a pre-processing step, where attribute selection is performed, using the CfsSubsetEval method from WEKA.

Results achieved with the Xmeans and EM models from WEKA (which can automatically determined the optimal number of clusters) were compared to other results produced with Xmeans (in [3]) and Quad Tree based K-Means algorithm. They conclude that both Xmeans and EM have good results if attribute selection is not performed, results that are better than the existing ones in most of the considered cases.

3. Methodology

In this section we introduce our unsupervised neural network model for defect identification in software systems.

The main idea of this approach is to represent an entity (class, module, method, function) of a software system as a multidimensional vector, whose elements are the values of different software metrics applied to the given entity. We consider that a software system S is a set of components (called *entities*) $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. We are considering a feature set of software metrics $\mathcal{SM} = \{sm_1, sm_2, ..., sm_k\}$ and thus each entity $s_i \in \mathcal{S}$ from the software system can be represented as a k-dimensional vector, having as components the values of the software metrics from $\mathcal{SM}$, $s_i = (s_{i1}, s_{i2}, \ldots, s_{ik})$ (s_{ij} represents the value of the software metric sm_j applied to the software entity s_i).

For each software entity, the label of the instance (defect or non-defect) is known. We mention that the labels will be used only in the pre-processing step and for evaluating the performance of the model.

3.1. Data pre-processing.

The first step before applying the SOM approach is the *data pre-processing* step. During this step, the input data is scaled to [0,1] using the *Min-Max* normalization method, and then a feature selection step will be applied. Details about the feature selection step will be given in the experimental part of the paper (Section 4). After applying the feature selection step, m software metrics (features) are selected to be further used for building the SOM.

Regarding the normalization method, we have to mention that in our approach the *minimum* and *maximum* values for the software metrics (features) from the training data are used for the *Min-Max* normalization step. We have focused in this paper only on the unsupervised scenario of grouping the existing entities from a software system into *defective* or *non-defective*. In a supervised learning scenario, in which new testing data is used, it is not useful to use for normalization the *minimum* and *maximum* values for the features from the training data. Instead, it would be a good idea to use, for each software metric, the *minimum* and *maximum* values for that software metric. Further extensions of our approach will investigate this situation.

3.2. **The SOM model.** Before designing the SOM model, the input dataset is pre-processed. For the training step of the SOM, a distance function between the input instances is required. We are considering the *distance* between the high-dimensional representation of two software entities s_i and s_j as the *Euclidean Distance* between their corresponding vectors of software metrics values. We have chosen the *Euclidean distance* because it is the most often used distance measure for SOM-based approaches and because, intuitively, considering the m-dimensional instances (preprocessed as mentioned in Section 3.1), the *Euclidean distance* will assign low distances to similar entities that are very likely to have the same output class (defect or not). Nevertheless, in the future we intend to extend our approach to use other distance measures as well.

The set of pre-processed software entities from the dataset $\mathcal{S}$ are grouped into clusters using a SOM. For the self-organizing map, the *torus* topology is used (Figure 1). In geometry, a torus is a surface of revolution generated by revolving a circle in the three dimensional space about an axis coplanar with the circle. It is shown in the literature that this topology provides better neighborhood than the conventional one [11].

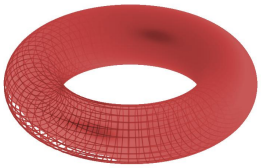


FIGURE 1. A torus

The goal of this step is to obtain two clusters corresponding to the two classes of instances: *defects* and *non-defects*. For grouping the software entities the following steps are performed.

- **Map Construction**. For a given number of *epochs* (training episodes), perform the following. Each m-dimensional training instance (software entity) is fed to the map. For each instance s_i the following steps are performed:
  (1) **Matching**. The neuron having its weight vector closest (considering the *Euclidean distance*) to instance s_i is declared the "winning" neuron. This is a competition phase, in which the output units from the map compete to match the input instance.
  (2) **Updating**. After the "winning neuron" was identified, the connection weights of the winning unit and its neighbors are updated,

such that are moved in the direction of the input instance by a factor determined by a learning rate.

- **Visualization**. After the training phase (the steps described above) was completed, in order to visualize the obtained map, the U-Matrix method [8] is used. The U-Matrix value of a particular node from the map is computed as the average Euclidean distance between the node and its closest 4 or 8 neighbors. These distances can be then be viewed as heights giving a U-Matrix landscape. The U-Matrix may be interpreted as follows [8]: high places on the U-Matrix encode data that are dissimilar while the data falling in the same valleys represent input instances that are similar. Thus, instances within the same valley can be grouped together to represent a cluster. Each cluster visualized on the map identifies a class of instances.

Once the map was built, it may also be used in a supervised learning scenario for classifying a new software entity. First, the "winning neuron" corresponding to this instance is determined (as indicated at the **Matching** step above). The cluster (class) to which the winning neuron belongs will indicate the class membership of the given entity.

3.2.1. *Testing.* For evaluating the performance of the SOM model, we are using several evaluation measures from the supervised classification literature. Since the training instances were labeled, the labels are used to compute the confusion matrix for the two possible outcomes (*non-defect* and *defect*). Considering the *defective* class as the *positive* one and the *non-defective* class as the *negative* one, the confusion matrix [16] for the defect detection task consists of: the number of *true positives* (TP), *false positives* (FP), *true negatives* (TN) and *false negatives* (FN).

Considering the values computed from the confusion matrix, the following evaluation measures will be used in this paper:

(1) False Positive Rate (FPR), computed as $\frac{FP}{FP+TN}$.
(2) False Negative Rate (FNR), computed as $\frac{FN}{FN+TP}$.
(3) Overall Error (OE), computed as $\frac{FN+FP}{FN+FP+TN+TP}$.

We have decided to use these measures, because they are used in papers presenting similar approaches, so a direct comparison of the results is possible. But the datasets used for the experiments are imbalanced, so we have decided to compute the value of a fourth performance measure as well: the *Area Under the ROC Curve (AUC)* [6]. The *ROC* curve is a two-dimensional plot of *sensitivity* vs. *(1-specificity)*, which in our case contains one single point, linked to the $(0, 0)$ and $(1, 1)$ points.

4. Experimental evaluation

In this section we provide an experimental evaluation of the SOM model (described in Section 3) on three case studies which were conducted on open source datasets. We mention that we have used our own implementation for SOM, without using any third party libraries.

4.1. **Datasets.** We have used three openly available datasets for the experimental evaluation of our model, called *Ar3*, *Ar4* and *Ar5*, which can be downloaded from [4]. All three datasets come from a Turkish white-goods manufacturer embedded software implemented in C. They all contain the value of 29 different McCabe and Halstead software metrics, computed for the functions and methods from the software systems, and one class label, denoting whether the entity is defective or not. The *Ar3* dataset contains metric values for 63 entities, out of which 8 are defective. The *Ar4* dataset contains 107 entities, out of which 20 defective, while the *Ar5* dataset has 36 entities, out of which 8 are defective.

For the SOM used in the experiments, the following parameter setting was used: the *number of training epochs* was set to 100000, the *learning coefficient* was set to 0.7, the *radius* was computed as half of the maximum distance between the neurons and the neighborhood function. We have tried out different parameter settings and we have achieved the best results with these values. However, we will perform in the future a thorough study to investigate the effect of different parameter settings.

4.2. **Data pre-processing.** In order to analyze the importance of the features, we are using the *information gain* measure. The *information gain* (IG) of a feature expresses the expected reduction in entropy determined by partitioning the instances according to the considered feature [13]. More exactly, the IG measure indicates the relevance of a feature in the defect classification task. Since the software metrics values (features values) are real numbers, in order to compute the information gain of the attributes we first discretize their values by dividing their interval of variation into ten sub-intervals.

For a better data analysis, we have computed the information gain of the features from the dataset obtained using all three datasets (*Ar3*, *Ar4* and *Ar5*) together. The information gain values for the features are shown in Figure 2.

Starting from the IG values of the software metrics, we have chosen a threshold value τ and considered only the attributes whose IG was higher than this threshold. Out of these attributes, we selected those that measure different characteristics of the software system. For the threshold τ we have selected the value 0.163, because we have achieved the best results with this value. Out of the 18 software metrics whose value was higher than τ, we have selected

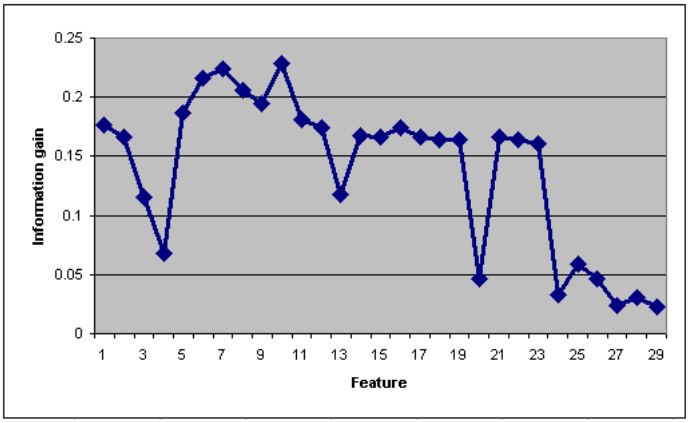


FIGURE 2. Information gain for the features.

the following 9 metrics: *halstead_vocabulary*, *total_operands*, *total_operators*, *executable_loc*, *halstead_length*, *total_loc*, *condition_count*, *branch_count*, *decision_count*. These selected attributes were used in the experimental evaluation on all three considered datasets. We mention that a different, possibly automatic, attribute selection method can also be implemented considering the IG values and will be further investigated.

4.3. **Results.** We are presenting in this section the results we have obtained by applying the SOM model on the *Ar3*, *Ar4* and *Ar5* datasets. For each dataset considered for evaluation, the experiments are conducted as follows. First, the data pre-processing step is applied and then the methodology indicated in Section 3 is used for an unsupervised construction of a torus SOM. The U-Matrix corresponding to the trained SOM will be visualized (the red labels on the U-Matrix represent the defective entities and the yellow labels represent the non-defective ones). Then, the evaluation measures presented in Section 3.2.1 will be computed for evaluating the performance of the obtained results.

4.3.1. *The Ar3 dataset.* A torus SOM, consisting of 150x8 nodes, was trained on the set of software entities from the *Ar3* dataset. Figure 3 depicts the U-Matrix visualization of the trained SOM.

Visualizing the U-Matrix for the resulting map, we have identified the two clusters, representing the defective and non-defective entities. The cluster with the defective entities contains 6 defective entities and 1 non-defective entity, thus we have 1 FP entity and 2 FN ones. All the other entities are placed

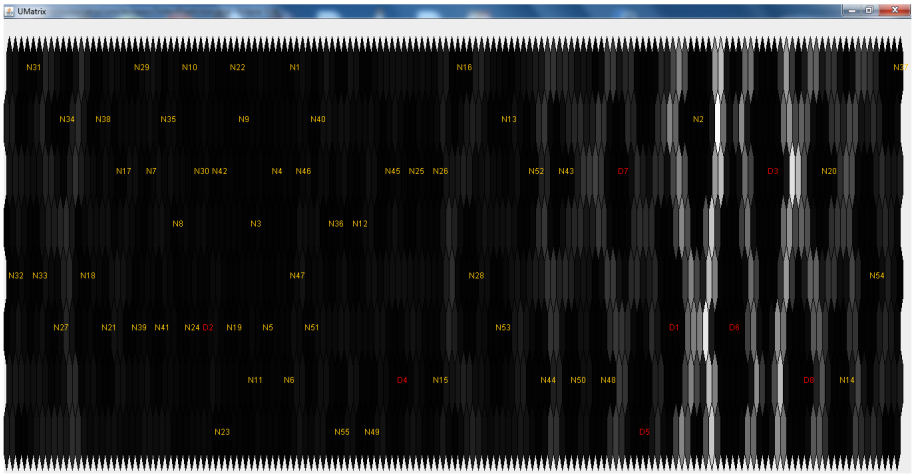


FIGURE 3. U-Matrix for the *Ar3* dataset.

in the correct cluster. The values of the performance measures from Section 3.2.1 are presented in the first three cells of the first row of Table 1.

4.3.2. *The Ar4 dataset.* A torus SOM, consisting of 150x8 nodes, was trained on the set of software entities from the *Ar4* dataset. The U-Matrix visualization of the obtained SOM is illustrated in Figure 4.

Visualizing the U-matrix for the resulting map, we have identified the two clusters which represent the defective and non-defective entities. The cluster with the defective entities contains 10 defective entities and 2 non-defective entities. Consequently we have 10 FN entities and 2 FP ones. The values of the performance measures are presented in the middle three cells of the first row of Table 1.

4.3.3. *The Ar5 dataset.* A torus SOM, consisting of 150x8 notes, was trained on the set of software entities from the *Ar5* dataset. The U-Matrix visualization of the trained SOM is presented in Figure 5.

From Figure 5 we can observe that the obtained SOM indicates a good topological mapping of the input instances, and also identifies subclasses within the defective and non-defective classes. Most of the defective entities are grouped together (there is only one FN), but there is also one non-defective entity in this cluster, so we have 1 FP as well. The values of the performance measures for this dataset are presented in the last three cells of the first row of Table 1.

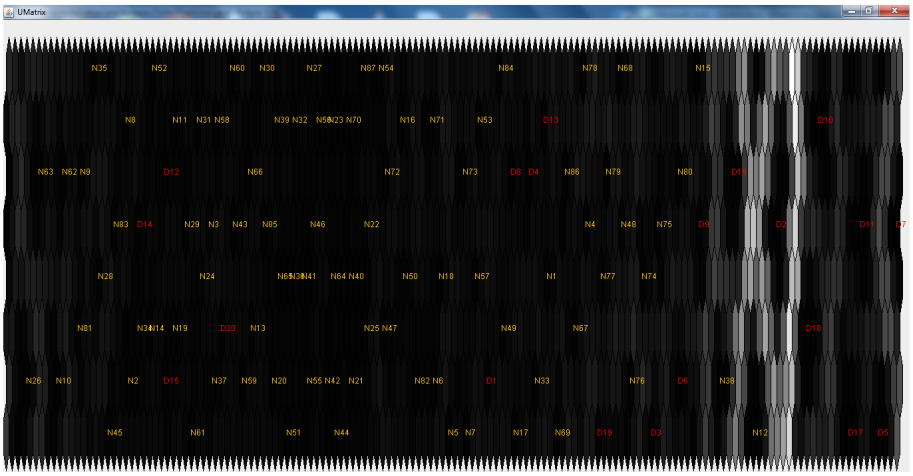


FIGURE 4. U-Matrix for the *Ar4* dataset.

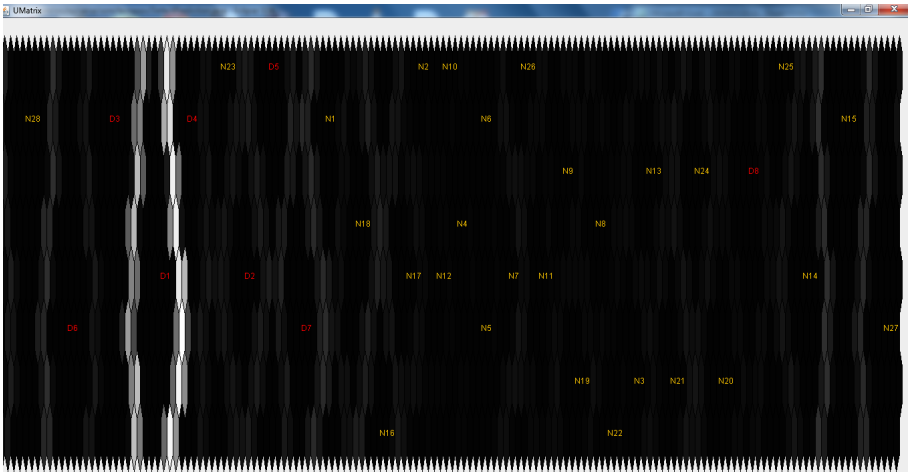


FIGURE 5. U-Matrix for the *Ar5* dataset.

5. Discussion and comparison to related work

An analysis of the approach we have introduced in Section 3 for detecting the defective entities from software systems will be provided in the following. A discussion on the obtained experimental results, as well as a comparison of them with similar approaches from the literature is conducted.

As presented in the previous section, our approach was capable of separating the defective and non-defective entities in two clusters, obtaining a good topological mapping of the input instances. Even if the separation was not perfect, for all three datasets we had both false positive and false negative entities. A major advantage of the self-organizing map is that it does not require supervision and no assumption about the distribution of the input data is made. Thus, it may find unexpected hidden structures from the data being investigated. Moreover, as seen from our experiments (Section 4), it is interesting that the SOM is able to detect, within the defective/non-defective class, subclasses of instances. This would be very useful, from a data mining perspective, since it may provide useful knowledge for the software developers.

As the accuracy of the trained SOMs depends on the choice of some parameters (number of training epochs, learning coefficient, neighborhood function), we have to measure it. One method for evaluating the quality of the resulting map is to calculate the *average quantization error* over the input samples, defined as the Euclidean norm of the difference between the input vector and the best-matching model [10]. Figures 6, 7 and 8 give a graphical representation of the average quantization error during the training steps, for each case study considered for evaluation. It can be easily seen that while the error fluctuates at the beginning of the training phase, it decreases during it, and after the training is completed, a very small average quantization error of the trained maps is obtained (1.6×10^{-6} for *Ar3*, 1.7×10^{-4} for *Ar4* and 6.7×10^{-13} for *Ar5*), which shows the accuracy of the trained maps.

Considering the subclasses identified within the clusters for the defective and non-defective entities as further work we propose to analyze these subclasses to identify the characteristics of the software entities placed in them.

Table 1 presents the values of the FPR, FNR and OE performance measures computed for the results of our approach, but it also contains values reported in the literature for some existing approaches, presented in Section 2.2. The Hyper Quad Tree-based approach presented in [18] does not report FNR values, this is marked with "NR" in the table.

From Table 1 we can see, that even if our approach does not provide the best results in each case, it has better results than most of the approaches. Out of 51 cases in total, our algorithm has a better or equal value for a performance measure in 43 cases, which represents **84.3%** of the cases.

The first line of Table 2 presents the value of the AUC measure computed for our approach. As presented in Section 3.2.1, for computing the value of the AUC measure we need to compute the *sensitivity* and *specificity* of the classification. Since *sensitivity* is equal to $1 - FNR$ and $(1 - specificity)$ is equal to *FPR*, we computed the value of the AUC measure for those approaches from the literature which report both of these values. They are presented in

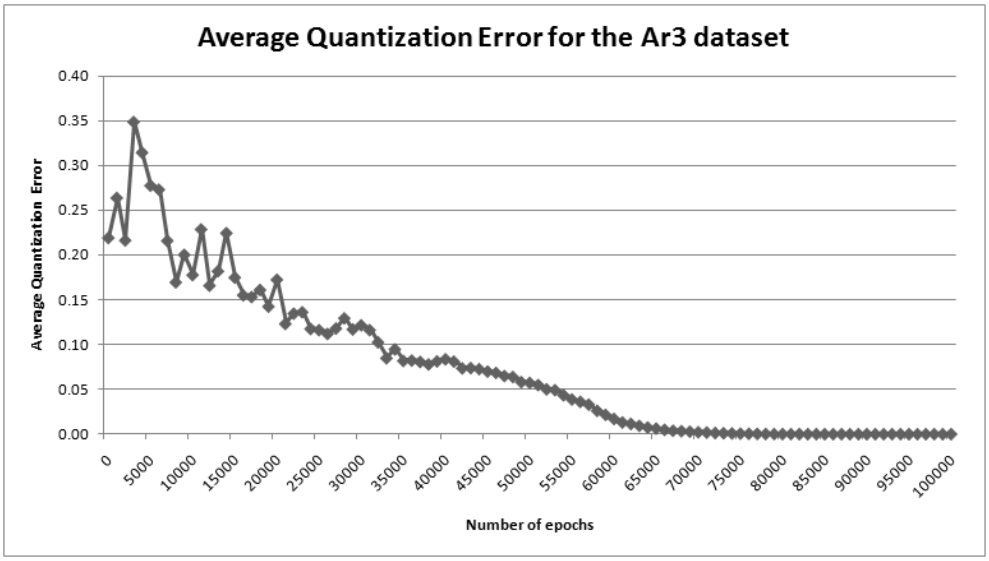


FIGURE 6. Average Quantization Error for the *Ar3* dataset.

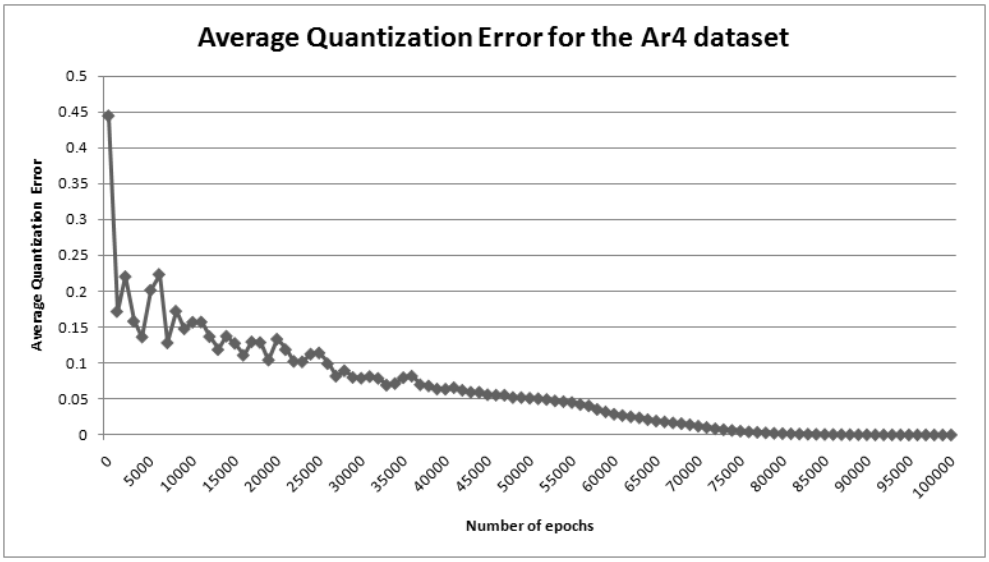


FIGURE 7. Average Quantization Error for the *Ar4* dataset.

Table 2 as well, and for each dataset the best value is marked with bold. Our approach has the highest AUC value for the *Ar5* dataset, the second highest

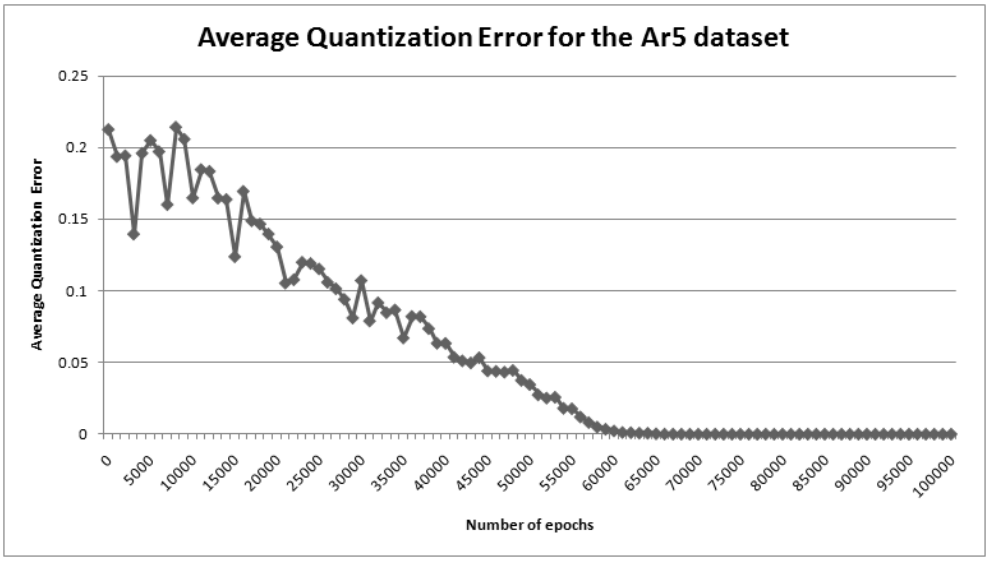


FIGURE 8. Average Quantization Error for the *Ar5* dataset.

	Ar3			Ar4			Ar5		
Approach	**FPR**	**FNR**	**OE**	**FPR**	**FNR**	**OE**	**FPR**	**FNR**	**OE**
Our SOM	0.0182	0.25	0.0476	0.0230	0.5	0.1121	0.0357	0.125	0.0556
SOM and threshold [1]	0	0.25	0.0556	0.1034	0	0.0938	0.0714	0.25	0.1111
K-means - QT [2]	0.3454	0.25	0.3333	0.0459	0.45	0.1214	0.1428	0.125	0.1388
K-means - Hyper QT [18]	0.0263	NR	0.0263	0.1875	NR	0.1846	0.0246	NR	0.0246
XMeans [3]	0.3455	0.25	0.3333	0.4483	0.05	0.3738	0.1429	0.125	0.1389
XMeans [14]	0.0727	0.25	0.0952	0.023	0.6	0.1308	0.149	0.125	0.1389
EM [14]	0.1091	0.25	0.127	0.023	0.6	0.1308	0.149	0.25	0.1667

TABLE 1. Comparison of the performance of our method to existing approaches.

value for the *Ar3* dataset and the fourth highest value for the *Ar4* dataset. Interestingly, for the *Ar3* and *Ar4* datasets the best value is achieved by the other SOM-based approach presented in the literature, suggesting that SOMs are indeed suitable for this problem.

6. Conclusions and future work

We have introduced in this paper a self-organizing feature map which may be used for an unsupervised detection of software defects. The experimental results obtained on three open-source datasets reveal a good performance of

Approach	Ar3	Ar4	Ar5
Our SOM	0.866	0.739	**0.92**
SOM and threshold [1]	**0.875**	0.**948**	0.839
K-means - QT [2]	0.702	0.752	0.866
XMeans [3]	0.702	0.751	0.866
XMeans [14]	0.839	0.689	0.863
EM [14]	0.820	0.689	0.801

TABLE 2. Comparison of AUC values.

the proposed approach, it provides better results than many of the existing approaches report in the literature.

Future work will be done in order to extend the evaluation of the proposed machine learning based model on other open source case studies and real software systems. We will also investigate the applicability of fuzzy [19] self-organizing maps for software defect detection, as well as to further consider techniques for data pre-processing and feature selection.

ACKNOWLEDGMENTS

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS–UEFISCDI, project number PN-II-RU-TE-2014-4-0082.

REFERENCES

[1] G. Abaei, Z. Rezaei, and A. Selamat. Fault prediction by utilizing self-organizing map and threshold. In *2013 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pages 465–470, Nov 2013.

[2] P.S. Bishnu and V. Bhattacherjee. Software fault prediction using quad tree-based k-means clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1146–1150, June 2012.

[3] C. Catal, U. Sevim, and B. Diri. Software fault prediction of unlabeled program modules. In *Proceedings of the World Congress on Engineering (WCE)*, pages 212–217, Dec 2009.

[4] Tera-promise repository. http://openscience.us/repo/.

[5] N. Elfelly, J.-Y. Dieulot, and P. Borne. A neural approach of multimodel representation of complex processes. *International Journal of Computers, Communications & Control*, III(2):149–160, 2008.

[6] T. Fawcett. An introduction to ROC analysis *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[8] S. Kaski and T. Kohonen. Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In *Neural Networks in Financial Engineering.*

Proceedings of the Third International Conference on Neural Networks in the Capital Markets, pages 498–507. World Scientific, 1996.

[9] Andreas Khler, Matthias Ohrnberger, and Frank Scherbaum. Unsupervised feature selection and general pattern discovery using self-organizing maps for gaining insights into the nature of seismic wavefields. *Computers & Geosciences*, 35(9):1757 – 1767, 2009.

[10] Teuvo Kohonen, Ilari T. Nieminen, and Timo Honkela. On the Quantization Error in SOM vs. VQ: A Critical and Systematic Study. In *Advances in Self-Organizing Maps, 7th International Workshop, WSOM*, pages 133–144. St. Augustine, FL, USA, 2009.

[11] Peter K. Kihato, Heizo Tokutaka, Masaaki Ohkita, Kikuo Fujimura, Kazuhiko Kotani, Yoichi Kurozawa, and Yoshio Maniwa. Spherical and torus som approaches to metabolic syndrome evaluation. In Masumi Ishikawa, Kenji Doya, Hiroyuki Miyamoto, and Takeshi Yamakawa, editors, *ICONIP (2)*, volume 4985 of *Lecture Notes in Computer Science*, pages 274–284. Springer, 2007.

[12] J. Lampinen and E. Oja. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2(3):261–272, 1992.

[13] Thomas M. Mitchell. *Machine learning*. McGraw-Hill, Inc. New York, USA, 1997.

[14] Mikyeong Park and Euyseok Hong. Software fault prediction model using clustering algorithms determining the number of clusters automatically. *International Journal of Software Engineering and Its Applications*, 8(7):199–205, 2014.

[15] Panu Somervuo and Teuvo Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10:151–159, 1999.

[16] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77 – 89, 1997.

[17] Ayse Tosun, Burak Turhan, and Ayse Basar Bener. Validation of network measures as indicators of defective modules in software systems. In *Proceedings of the 5th International Workshop on Predictive Models in Software Engineering, PROMISE 2009, Vancouver, BC, Canada, May 18-19, 2009*, pages 5–14, 2009.

[18] Swati Varade and Madhav Ingle. Hyper-quad-tree based k-means clustering algorithm for fault prediction. *International Journal of Computer Applications*, 76(5):6–10, August 2013.

[19] Lotfi A. Zadeh. A summary and update of "fuzzy logic". In *2010 IEEE International Conference on Granular Computing, GrC 2010, San Jose, California, USA, 14-16 August 2010*, pages 42–44, 2010.

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

E-mail address: {marianzsu, istvanc, gabis}@cs.ubbcluj.ro
E-mail address: ssic0977@scs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

APPLYING SUPPORT VECTOR REGRESSION METHODS FOR HEIGHT ESTIMATION IN ARCHAEOLOGY

VLAD-SEBASTIAN IONESCU

Abstract. In this paper we apply Support Vector Machines to the problem of predicting the height of human skeletons given bone measurements. There exist archaeological methods for estimating height, but our purpose is to investigate the potential of Support Vector Regression for this task. Since skeletal stature clearly depends on individual bone lengths, building SVM models for this task has the potential of giving an accurate machine learning automation for this task, which can be useful for archaeologists. We investigate multiple kernels and performance evaluation methodologies and compare our results to existing literature results on the topic. Our experiments show that SVM regression models are very good for the problem at hand, outperforming existing approaches.

1. Introduction

A very important problem in archaeology and forensic science is the problem of height estimation. Existing approaches that deal with this problem involve simple regression formulas based on statistical methods. Our goal is to investigate the potential of more complex methods, such as Support Vector Machines (SVMs). We believe that the good performance of Support Vector Machines on other problems can make them ideal for solving the problem of height estimation as well, due to their good performance at inferring complex relationships between data: in our case between the bone measurements and the associated height of the skeletons.

This is a problem that is difficult even for humans, with no clear relationship between the features and their labels, which suggests the use of machine learning models.

Received by the editors: June 3, 2015.

2010 *Mathematics Subject Classification.* 68T05, 68T01.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Concept learning*; I.2.7 [**Artificial Intelligence**]: Learning – *Induction*.

Key words and phrases. height estimation, regression, support vector machines, archaeology, forensic science.

The problem is important for researchers in archaeology and related fields because it allows them to discover important facts about a certain population, relating to issues such as their health, gender differences and body sizes at different times in history.

We propose using SVM regression models because of their known good performance in general on regression tasks, their efficient implementations in various libraries and their ability to easily adapt to multiple problem settings through the use of kernels. To the best of our knowledge, our approach is novel, since SVM models have not been used in the literature for height estimation until now. Our obtained experimental results are significantly better than the existing ones in the literature and prove the ability of machine learning models and SVMs in particular to solve this problem.

Our experiments are performed on open source skeletal data which was previously used for this task. Because of this, we have a relevant baseline to compare our results against.

The paper has the following structure. Section 2 presents the problem of height estimation and its importance for archaeologists, as well as existing approaches for solving it. Section 3 provides an overview of Support Vector Regression. Section 4 presents our experimental setup and methodology. Section 5 presents our data sets and the results obtained on each one. Section 6 presents a comparison of our results to related work and Section 7 contains our conclusions and future research directions.

2. The height estimation problem

According to [17, 3], height estimation is a central part of anthropological analysis and is generally used in order to determine social stucture in extinct populations. More complex theories can also be inferred from stature, relating to health, body size trends and adaptability to environmental changes.

The first anatomical method for height estimation was introduced by Thomas Dwight in 1894, a method that caused significant errors. Karl Pearson then introduced regression formulas, but there were studies that identified certain shortcomings regarding their applicability to different populations [10, 8, 9].

A milestone approach is introduced in [2], along with the open source data sets we use in this article. The approach consists of multiple formulas based on measurements of important bones in the human body.

A variety of approaches exist for this problem, which proves its importance. The existing methods are either anatomical or mathematical in nature. A study comparing the two classes of methods can be found in [18], which concludes that anatomical methods are superior only if the skeletal remains

are sufficiently complete and that otherwise the mathematical methods are to be preferred.

A comprehensive literature review on this topic can be consulted in [15]. We have previously introduced two novel machine learning models based on artificial neural networks and genetic algorithms for the problem approached in this paper [7].

3. Support Vector Regression

Support Vector Machines were introduced for binary classification by Vapnik as far back as 1963 and further developed by Cortes and Vapnik [16]. The idea behind SVMs is to find a maximum margin separating hyperplane. They are very robust to different problems due to the kernel trick, which allows them to accurately do non-linear classification as well, and due to the fact that, unlikely neural networks, they do not have to concern themselves with local optimums.

Support Vector Regression is an extension of SVMs to regression problems. A widely used method is called ε-Support Vector Regression, in which we want to find a function that approximates each training example with at most ε error, if possible, or allow for some degree of error (specified by a hyperparameter $C > 0$) if not. This is similar to the width of the margin in Support Vector Classification.

The mathematical formulation in which only an error of ε is allowed is given in Formula (1) [16, 1].

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to} \quad \begin{cases} y_i - (w \cdot x_i + b) \leq \varepsilon \\ (w \cdot x_i + b) - y_i \leq \varepsilon \end{cases} \end{aligned} \tag{1}$$

The more robust formulation that allows for some mistakes and is used in practice is given in Formula (2) [16, 1].

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i^- + \xi_i^+) \\ &\text{subject to} \quad \begin{cases} y_i - (w \cdot x_i + b) \leq \varepsilon + \xi_i^- \\ (w \cdot x_i + b) - y_i \leq \varepsilon + \xi_i^+ \\ \xi_i^-, \xi_i^+ \geq 0 \end{cases} \end{aligned} \tag{2}$$

In both equations, standard notation is used: w is a weight vector that we use to multiply the input with and make predictions, m is the number

of training instances, x_i is a training sample, y_i is its target and b is a bias term. The resulting optimization problems are then solved using numerical algorithms. Various kernel functions can also be used in order to obtain non-linear classification or regression [16].

Intuitively, Support Vector classification and regression are similar in that classification attempts to find a separating hyperplane that goes as much through the "middle" of the space that exists between the data classes and regression attempts to find a hyperplane that goes as much through the "middle" of the data as possible. The hyperparameter C is what controls how much "slack" we give the algorithm, or how much we tolerate mistakes.

For reference, Formula (3) presents some of the most often used kernels for Support Vector Machines.

$$
\begin{aligned}
K_{linear}(x, x') &= x \cdot x' \\
K_{polynomial}(x, x') &= (\gamma(x \cdot x') + r)^d \\
K_{rbf}(x, x') &= e^{-\gamma\|x-x'\|^2} \\
K_{sigmoid}(x, x') &= tanh(\gamma(x \cdot x') + r)
\end{aligned} \tag{3}
$$

4. Experimental setup and methodology

In this section, we describe the software libraries and the experimental and testing methods used for running our experiments.

4.1. **Software libraries and experimental methods.** All of our experiments are performed using the **scikit-learn** machine learning library [14]. For Support Vector Machines, this in turn uses the **libsvm** library [5], which is known to be a very powerful library for SVMs. Using well-known open source and well documented libraries ensures bug free experiments and guarantees that our experiments can easily be reproduced by other researchers.

We run experiments using the linear, RBF (Radial Basis Function), polynomial and sigmoid kernels. We use a randomized grid search to tune the hyperparameters for our SVM model (such as the C and ε values). It has recently been shown that using a random search is better than using a standard grid search [13]. We use 10 fold cross validation [6] as the model evaluation method within the random search. The parameter configuration that gives the best results, according to the Mean Absolute Error (MAE), is returned after 1000 random parameter samplings from a uniform distribution over $[0, 1)$ for each model parameter, except the d parameter for the polynomial kernel, which was sampled from the set $\{1, 2, 3, 4, 5\}$, and the C parameter, which was sampled from $[0, 10000)$ when optimizing the RBF kernel, from $[0, 10)$

when optimizing the sigmoid kernel and from $[0, 100)$ when optimizing the linear kernel. For the polynomial kernel, the grid searches found $d = 1$, which basically considers a linear kernel, so we do not include it in the presentation.

We also normalized our data by mean subtraction and division by the standard deviation.

For finding the optimal hyperparameters, we have added the normalization step as the first step of a pipeline, with the second and final step being the Support Vector Regressor. Our normalizer only scales features. The resulting pipeline is then used as the final regressor given to the randomized search for optimization. This entails that mean and standard deviation are computed on the training folds and the same values are used to normalize the test fold during testing. After normalization, the test fold is fed to the actual regressor part of the pipeline.

The way in which we optimize hyperparameters using randomized grid searching is fixed (we will refer to it as the **M1** method). However, we consider the following methods as well, which we will optimize using a standard grid search over a feasible set of hyperparameters.

(1) Method **M2** involves normalizing our entire data prior to doing 10 fold cross validation and also normalizing our targets (the correct statures). Performance scores are reported considering the values returned by the model unscaled. This approach can potentially overestimate the performance of a model, due to the fact that it does not really mimic real world scenarios by assuming we can also normalize the test data together with the training data;
(2) Method **M3** uses the pipeline approach of **M1**, but it also normalizes targets. We believe this to be a realistic application scenario that can help improve performance.

4.2. **Testing method.** For each methodology and kernel, we use a single run of 10 fold cross validation per iteration, storing and using the hyperparameters that define the model which minimizes the Mean Absolute Error score (MAE) over all iterations. That model is then used to report the MAE and Standard Error of the Estimate (SEE) according to Formula (4). In this formula, m is the number of considered instances, y_i is the known target value, and $\hat{y}_i$ is the value given by our trained model.

A 95% confidence interval for the mean on the 10 test folds is also reported, as described in [4].

We present our results for two case studies representing different populations as well as for their concatenation.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i| \qquad SEE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{4}$$

5. Data sets and experimental results

In this section we present the experimental evaluation of the SVM model on three case studies, following the process described in Section 4.

5.1. **Data sets.** Our data set is open source and taken from [2]. It consists of two case studies, both in the same format and containing seven features related to skeletal bone measurements in centimeters and the skeleton gender: the length of the humerus, the radius and ulna lengths, the femur, tibia and fibula lengths, the length of the whole leg (femur+tibia) and the gender. For each of the bones, the measurement represents the length of the longest of the two bones. Each of the two case studies contains 92 instances: the first one contains measurements of caucasians and the second one measurements of afro-americans (47 males and 45 females in each).

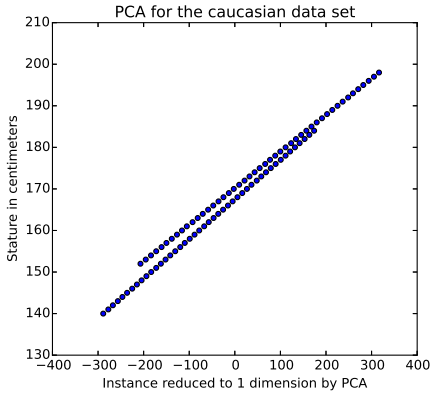
5.2. **Experimental results.** Figure 1 presents the two case studies reduced to a single dimension using Principal Component Analysis (PCA) [11]. The x axis of this graph represents the value of the single feature computed by PCA and the y axis represents heights. It can be seen that even under this setting, it is trivial to find a linear function that approximates the data, which makes us expect very good results from the linear SVM kernel, since it will be able to make use of more features, when even a single one shows good potential, at least if obtained by PCA.

5.2.1. *First case study - Caucasians.* Table 1 presents results for the first case study under all three evaluation methodologies.

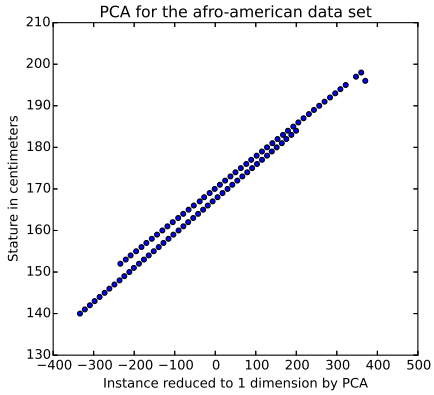
For the Caucasian case study, the linear kernel proved to be the best under the **M1** testing methodology, while the RBF kernel proved to be the best under **M2** and **M3**.

For the **M2** and **M3** testing methodologies, we obtained the best results with identical model hyperparameters. This is understandable, since the methods are very similar and the scaled targets cannot differ too much in our data set. For larger datasets, the differences could potentially be bigger, thus requiring different hyperparameters. Therefore, one might want to employ a randomized grid search for the other methodologies as well. For our purposes however, we wanted to showcase both kinds of searches.

We note that all three kernels give very good results under all three methodologies, considering that the errors are in centimeters and the data



(A) Caucasian case study



(B) Afro-american case study

FIGURE 1. Data set reduced to a single feature using Principal Component Analysis.

Kernel	MAE (cm)	SEE (cm)	Hyperparameters	M
linear	0.046±0.010	0.056±0.011	$C = 91.92, \varepsilon = 0.07$	M1
RBF	0.088±0.061	0.101±0.068	$C = 9625.77, \varepsilon = 0.07, \gamma = 0.004$	
sigmoid	0.881±0.504	0.913±0.504	$C = 4.064, \quad \varepsilon = 0.114,$ $\gamma = 0.0162, \quad r = 0.164$	
linear	0.049±0.009	0.057±0.009	$C = 5, \varepsilon = 0.001$	M2
RBF	0.042±0.014	0.050±0.015	$C = 910, \varepsilon = 0.0001, \gamma = 0.001$	
sigmoid	0.490±0.350	0.515±0.355	$C = 5, \quad \varepsilon = 0.0001,$ $\gamma = 0.01, \quad r = 0.0001$	
linear	0.049±0.010	0.057±0.010	$C = 5, \varepsilon = 0.001$	M3
RBF	0.040±0.012	0.048±0.014	$C = 910, \varepsilon = 0.0001, \gamma = 0.001$	
sigmoid	0.735±0.515	0.756±0.520	$C = 5, \quad \varepsilon = 0.0001,$ $\gamma = 0.01, \quad r = 0.0001$	

TABLE 1. Results obtained on the Caucasian case study. 95% confidence intervals are used for the results.

set contains almost 100 instances, which means that even in the worst case (0.881 MAE for the sigmoid kernel in the **M1** methodology), our average mistake is under one centimeter.

Overall, the best results are obtained under the **M3** methodology.

Kernel	MAE (cm)	SEE (cm)	Hyperparameters	M
linear	0.031±0.008	0.041±0.017	$C = 28.41, \varepsilon = 0.017$	M1
RBF	0.107±0.097	0.141±0.131	$C = 7304.752, \varepsilon = 0.037, \gamma = 0.02$	
sigmoid	0.268±0.099	0.339±0.171	$C = 9.844, \quad \varepsilon = 0.013,$ $\gamma = 0.0062, \quad r = 0.479$	
linear	0.056±0.052	0.114±0.154	$C = 0.1, \varepsilon = 0.001$	M2
RBF	0.051±0.030	0.081±0.072	$C = 910, \varepsilon = 0.001, \gamma = 0.001$	
sigmoid	0.146±0.090	0.187±0.141	$C = 0.9, \quad \varepsilon = 0.0001,$ $\gamma = 0.01, \quad r = 0.0001$	
linear	0.057±0.053	0.116±0.157	$C = 0.1, \varepsilon = 0.001$	M3
RBF	0.055±0.037	0.090±0.091	$C = 910, \varepsilon = 0.001, \gamma = 0.001$	
sigmoid	0.195±0.147	0.220±0.167	$C = 0.9, \quad \varepsilon = 0.0001,$ $\gamma = 0.01, \quad r = 0.0001$	

TABLE 2. Results obtained on the African-american case study. 95% confidence intervals are used for the results.

5.2.2. *Second case study - Afro-americans.* Table 2 presents results for the second case study under all three evaluation methodologies.

Once more, the linear kernel is the best under the **M1** methodology. Compared to the first case study's **M1** results, scores are better with the linear and sigmoid kernels and worse with the RBF kernel, although the differences are very small for any practical concerns.

The RBF kernel took the top spot under the **M2** and **M3** methods once again. Compared with the first case study, the **M2** and **M3** results are worse for the Afro-american case study, only the sigmoid kernel in the **M2** setting managing to surpass its direct competitor.

Overall, results are worse on the second case study than on the first, but not in a significant fashion.

This time, the best results are obtained under the **M1** methodology.

5.2.3. *Mixed case study.* The mixed case study consists of the concatenation of the data sets corresponding to the previous two case studies, resulting in a bigger data set that contains both populations.

We have not added any new feature to distinguish the two populations.

Since the two populations look similar in the PCA plots (Figure 1), we expect their concatenation to yield good results as well.

Table 3 presents results for the mixed study. This time, the radial basis function kernel clearly outperforms the other kernels taken into consideration.

Kernel	MAE (cm)	SEE (cm)	Hyperparameters	M
linear	0.340±0.103	0.417±0.129	$C = 98.508, \varepsilon = 0.394$	M1
RBF	0.243±0.152	0.347±0.223	$C = 9563.38, \varepsilon = 0.164, \gamma = 0.009$	
sigmoid	2.604±1.177	2.855±1.208	$C = 5.824, \quad \varepsilon = 0.0972,$ $\gamma = 0.01, \quad r = 0.7156$	
linear	0.458±0.225	0.551±0.254	$C = 0.29, \varepsilon = 0.001$	M2
RBF	0.261±0.137	0.367±0.215	$C = 10000, \varepsilon = 0.0001, \gamma = 0.001$	
sigmoid	1.246±0.391	1.305±0.389	$C = 6, \quad \varepsilon = 0.1,$ $\gamma = 0.01, \quad r = 0.0001$	
linear	0.469±0.223	0.577±0.270	$C = 0.29, \varepsilon = 0.001$	M3
RBF	0.252±0.130	0.376±0.268	$C = 10000, \varepsilon = 0.0001, \gamma = 0.001$	
sigmoid	1.469±0.498	1.585±0.527	$C = 6, \quad \varepsilon = 0.1,$ $\gamma = 0.01, \quad r = 0.0001$	

TABLE 3. Results obtained on the mixed case study. 95% confidence intervals are used for the results.

This implies that the RBF kernel is the most robust, being able to deal with more data instances even if they are from different populations. This suggests that the RBF kernel should perform the best in practice.

For the **M2** and **M3** methodologies, we again did not obtain significant differences in results with different model hyperparameters.

We obtained the best results under the **M1** methodology.

Another testing scenario that we plan to research in the future involves the concatenation of the two data sets, but with a new feature added that specifies which population each instance belongs to.

Figure 2 shows the learning curves for the RBF kernel on the mixed case study, unde the **M1** testing methodology. It can be seen the model generalizes very well and there is no overfitting. Because of the good generalization, it is unlikely for more data to produce better results. Because the training score error increases very slowly, it is likely that more data will not lead to worse results. It can also be seen that the model achieves its optimal performance with few training instances.

6. Comparison to related work

All of our results outperform existing literature results. As far as we are aware of, only [2] presents results on these data sets. Their obtained SEE

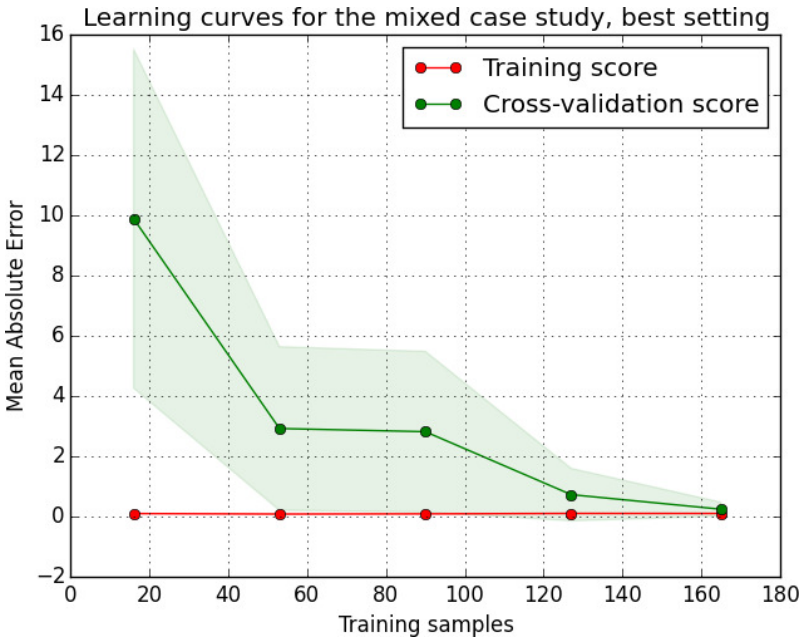


FIGURE 2. Learning curves for the mixed case study, with the RBF kernel under the **M1** methodology and considering MAE scores.

errors are between 3.05 and 3.66 cm. Our worst result is 2.855 SEE on the mixed case study using the sigmoid kernel under the **M1** testing methodology.

On the individual case studies, all of our results are well below 1 SEE, with most of them being under 0.5 and the best of them under 0.1.

Taking into consideration work done on other data sets but regarding the same issue, such as [17, 3, 12, 19] and others, we note that no previously existing method manages to achieve errors less than a centimeter, under any scoring metric. This can be attributed to the fact that previous methods only seem to consider a few features, to which they apply rather basic statistical procedures. It is impossible for us to test SVM applications on those data sets, since they are not public. Therefore, a direct comparison between our results and existing results on the data sets in [17, 3, 12, 19] would be meaningless. However, it stands to reason that, given the results we have presented in Section 5, SVM-based models can potentially outperform existing methods on other data sets as well.

Table 4 presents a quick comparison between our methods and other literature results on the data set we have worked with.

Method	Best SEE	Worst SEE	Short description
SVM	0.048	0.913	First case study, RBF kernel under the **M3** methodology for the best SEE and sigmoid kernel under the **M1** methodology for the worst case.
SVM	0.041	0.339	Second case study, linear and sigmoid kernels, both under the **M1** methodology.
SVM	0.347	2.855	Mixed case study, RBF kernel under the **M1** methodology and sigmoid kernel also under the **M1** methodology.
[2]	3.05	3.66	Regression formulas based on basic statistical methods.

TABLE 4. Overview of literature results on the data set we have used.

Therefore, our Support Vector Regression approach is a very reliable solution to the problem of skeletal height estimation given the lengths of certain bones, leading to much better results than previous approaches and having the potential to be easily adapted to new data sets from the field.

7. Conclusions and future work

We have presented in this paper how SVM regression methods can be applied for estimating the height of archaeological remains. We have run extensive experiments on two archaeological data sets that are freely available, obtaining very good results that surpass previous results from the literature.

Our results are also superior considering other data sets. While this comparison does require further investigation, it is a valid conjecture due to the fact that SVMs are a family of machine learning algorithms, which means they can learn from any type of data. If they could learn well on one data sets, it stands to reason that they are able to do the same on another data set containing similar kind of data.

We have applied three testing methodologies, which we believe to mimic certain real world scenarios well. We also used a randomized grid search for one of them, which recent research has shown to be the best was of optimizing hyperparameters.

Therefore, we consider the SVM-based methods that we have applied to offer significant contributions to the fields of archaeology and forensic analysis and believe that they will generalize well to other problems of a similar nature.

As a future research direction, we plan to experiment with more machine learning libraries and on more data sets. We also plan to find different kernels to test. Another possible direction is refining our hyperparameter searches, by reducing the intervals we search in and by running the randomized search for more iterations, increasing the probability of finding better hyperparameters.

If we can obtain more data, we believe that researching online learning options would also be a useful undertaking.

Since the PCA reduction did not seem to generate useless data, it is also worth investigating the results that can be obtained using less features, since fewer measurements should always be helpful in practice.

References

[1] A. Smola., B. Schlkopf, *A tutorial on support vector regression*, Statistics and Computing, 14 (2004), no. 3, pp. 199-222.

[2] M. Trotter, G. Gleser, *Estimation of Stature from Long Bones of American Whites and Negroes*, American Journal of Physical Anthropology, **10**(1952), no. 4, pp. 463-514.

[3] D.C. Pal, A.K. Datta, *Estimation of stature from radius length in living adult Bengali males*, Indian Journal of Basic and Applied Medical Research, 3 (2014), pp. 380-389.

[4] L. Brown, T. Cat, A. DasGupta, *Interval estimation for a proportion*, Statistical Science, 16 (2001), pp. 101-133.

[5] C.-C Chang, C.-J., Lin, *LIBSVM: A library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 2 (2011), no. 3, pp. 27:1-27:27.

[6] T.M. Mitchell, *Machine Learning*, McGraw-Hill, Inc. New York, USA, 1997.

[7] G. Czibula, V. Ionescu, D. Miholca, I. Mircea, *Novel supervised learning based approaches for stature prediction of archaeological skeletal remains from bones*, Journal of archaeological science, under review.

[8] P.H. Stevenson, *On racial differences in stature long bone regression formulae, with special reference to stature reconstruction formulae for the chinese*, Biometrika Trust, 21 (1929), pp. 303-318.

[9] A. Telkka, *On the Prediction of Human Stature from the Long Bones*, Acta Anatomica, 9 (1950), no. 1-2, pp. 103-117.

[10] C.W. Depertuis, J.A. Hadden, *On the reconstruction of stature from long bones*, American Journal of Physical Anthropology, 9 (2005), no. 1, pp. 15-54.

[11] M.E. Tipping, C.M. Bishop *Probabilistic principal component analysis*, Journal of the Royal Statistical Society, Series B, 61 (1999), pp. 611-622.

[12] M.R. Feldesman, *Race Specificity and the Femur/Stature Ratio*, American Journal of Physical Anthropology, 100 (1996), no. 2, pp. 207-224.

[13] J. Bergstra, Y. Bengio, *Random search for hyper-parameter optimization*, Journal of Machine Learning Research, 13 (2012), pp. 281-305.

[14] F. Pedregosa et al., *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825-2830.

[15] N. Navsa, *Skeletal morphology of the human hand as applied in forensic anthropology*, Ph.D. thesis, University of Pretoria, 2010.

[16] C. Cortes, V. Vapnik, *Support-vector networks*, Machine Learning, 20 (1995), no. 3, pp. 273-297.

[17] D. McCarthy, *The long and short of it: The reliability and inter-populational applicability of stature regression equations*, Master's thesis, Oregon State University, 2001.
[18] J.K. Lundy, *The mathematical versus anatomical methods of stature estimate from long bones*, The American Journal of Forensic Medicine and Pathology, 6 (1985), pp. 73-75.
[19] G. Fully, *Une Nouvelle Methode de Determination De la Taille*, Annales de Medecine et de Criminologie, 36 (1956), pp. 266-273.

"Babeş-Bolyai" University,, Faculty of Mathematics and Computer Sciences, 1, Kogălniceanu Street,, 400084 Cluj-Napoca,, Romania
E-mail address: ivlad@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

METRICS-BASED REFACTORING STRATEGY AND IMPACT ON SOFTWARE QUALITY

SIMONA MOTOGNA, CAMELIA ŞERBAN, AND ANDREEA VESCAN

Abstract. Nowadays, software systems become very large and complex applications. They are the result of a process evolution, undergoing frequent modifications that affect their design quality. Therefore, repeated assessment of the systems design should be made throughout the development lifecycle. This activity is time-consuming, because of design complexity and therefore, methods and techniques being needed to evaluate the system design in an automatic manner. Software metrics are an alternative solution, a means for quantifying those aspects considered important for the assessment.

In this article we propose a case-study for object oriented design (OOD) assessment using software metrics. Our goal is to identify those design entities affected by "God Class" design flaw, and to identify possible refactorings that could improve the system design and to evaluate the impact of the applied refactoring on software quality.

1. Introduction

Over an extended period of time software systems are often subject to a process of evolution becoming very large and complex applications. They undergo repeated modifications in order to satisfy any requirement regarding a business change. The result is that the code deviates from its original design and the system becomes unmanageable. A minor change in one of its parts may have unpredictable effects in completely other parts [1]. To avoid such risk a high quality design should be preserved throughout the system life cycle. This can be achieved by repeatedly assessing the system design, aiming to identify in due course those design entities that do not comply with the rules, principles and practices of a good design, and suggesting possible refactorings or improvements to be performed.

Received by the editors: September 27, 2015.

2010 *Mathematics Subject Classification.* 68N30, 68T37.

1998 *CR Categories and Descriptors.* D.2.8. [**Software Engineering**]: Metric – *Product Metrics*; D.2.8. [**Software Engineering**]: Refactoring – *Extract Method*; D.1.5. [**Pattern recognition**]: Clustering – *Fuzzy Clustering*.

Key words and phrases. Software metrics, refactoring, fuzzy clustering.

Due to the complexity of OOD, its assessment becomes a time-consuming activity. Consequently, methods and techniques are needed in order to evaluate the system design in an automatic manner. Software metrics are an alternative solution, being a means for quantifying those aspects considered important for the assessment.

Our previous work [2] was focused on developing a methodology for quantitative evaluation of OOD. The proposed methodology is based on static analysis of the source code and is described by a framework of four abstraction layers. To complete the interpretation of the obtained measurement results, a new metric, named Design Flaw Entropy (DFE), which measures the distribution of a specified design flaw among the analyzed design entities, was also proposed [7].

Starting from the above mentioned our previous proposed methodology for OOD, the current article adds a new step in the process of OOD assessment. More precisely, after the interpretation of the obtained measurements results which has the goal of identifying "suspect" design entities, we suggest possible refactorings and we study their impact regarding the design improvement. In other words, at this step we apply some refactorings and we repeat the evaluation for the obtained design, comparatively studying the two sets of metrics values.

The paper is organized as follows. Section 2 briefly describes the process of OOD assessment, whereas in Section 3 we present a case-study in order to validate our approach for design assessment. Section 4 summarizes the contributions of this work and outlines directions for further research.

2. Object oriented design assessment process

The steps needed to be perform in order to apply our previous proposed methodology for OOD assessment are described in what follows.

2.1. **Setting the assessment objectives.**

The first step of OOD assessment process is to establish the assessment objectives. In this study we aim to identify those design entities affected by "God Class" [6] design flaw. Consequently, the assessed entities, are the set of classes from the analyzed design system.

An instance of God Class does most of the operation tasks, leaving only minor details to a series of trivial classes; it also uses the data from other classes. Briefly, *God Class* design flaw refers to those classes "which tend to centralize the intelligence of the system" [1]. As a consequence, the principle of manageable complexity is violated, as god classes tend to capture more than one abstraction. Another shortcoming of these pathological classes is their

tendency towards non-cohesion. If we consider the quality attributes, god-classes also have a negative impact on the reusability and understandability of that part of the system they belong to. To detect a God Class, Salehie et al. [5] have related this design flaw to a set of three heuristics [4]:

- distribute system intelligence horizontally as uniformly as possible;
- beware of classes with much non-communicative behavior;
- beware of classes that access directly data from other classes.

Therefore, each class design entity, will be evaluated in respect with the above mentioned heuristics.

2.2. **Metrics identification.** Having identified some heuristics which are correlated with "God Class" design flaw, our goal is to find for each heuristic relevant metrics. Looking at the above heuristics the conclusion drawn may be:

- the first rule suggests that it should be a uniform distribution of intelligence among classes, so it refers to high class complexity;
- the second rule refers to the level of intra-class communication; i.e. to weak cohesion of classes;
- the third rule refers to classes that use a lot of data from other classes.

Thus, the selected heuristics are then related with the following metrics:

- Cyclomatic complexity [11] is a measure of a module control flow complexity based on graph theory. A control flow graph describes the logic structure a software module. Each flow graph consists of nodes and edges. The nodes represent computational statements or expressions, and the edges represent the transfer of control between nodes [12]. Cyclomatic complexity is defined for each module to be e n + 2, where e are the number of edges and n are the number of nodes in the control flow graph.

  Impact of CC metric value on software quality. A high value for the cyclomatic complexity metric indicates a low quality code which might involve difficulties in testing and maintaining.

  The CC metric is defined on methods. Adapted to the object oriented world, this metric [10] is also defined for classes and structures as the sum of its methods CC.
- Lack of Cohesion Of Methods (LCOM) [8]. LCOM is defined by the difference between the number of method pairs using common instance variables and the number of method pairs that do not use any common variables.

  Impact of CC metric value on software quality. The single responsibility principle states that a class should not have more than one

reason to change. Such a class is said to be cohesive. A high LCOM value generally pinpoints a poorly cohesive class. Looking into LCOM metric, Henderson-Sellers [9] points out the large number of dissimilar situations with 0 value of LCOM.

- Efferent Coupling at type level (Ce). The Efferent Coupling for a particular type is the number of types it directly depends on. Notice that types declared in third-party assemblies are taken into account.

  Impact of CC metric value on software quality. Types where Ce ¿ 50 are types that depends on too many other types. They are complex and have more than one responsibility. They are good candidate for refactoring.

Thus, at this step of the assessment process, each class can be viewed as a vector with the corresponding values of the LCOM, CC, Ce metrics.

Analyzing the definitions of these metrics and taking into account the above mentioned principles and heuristics related with "God Class", we can conclude that a possible God Class suspect will have high values for the LCOM, Ce si CC metrics.

2.3. **Fuzzy based God Class Detection and DFE metric.** Having computing the metrics values, for each class design entity, the next step is to identify the list of "suspect", i.e those classes affected by "God Class" design flaw. In order to obtain these entities we use fuzzy clustering analysis. This method overcome the limitations of the existing approaches which use thresholds values for metrics, thus an entity (class) will be place in more than one group (cluster), having different membership degree. Each cluster of the obtained partition is analyzed in detail, in order to decide if it contains "suspect" classes.

A metric proposed in our previous work [7], Design Flaw Entropy, provides an in-depth analysis regarding the distribution of the analyzed design flaw (the degree of its spread) into the system.

2.4. **Proposed Refactorings and their impact.** At the step of the assessment process, we aim to identify those refactorings which could be applied in order to improve the system's design accordingly with the studied design flaw. Taking into account that classes with high values of metrics CC and Ce are hard to understand and maintain, we then identify a set of possible refactoring strategies to be applied on the list of suspect entities. In this paper, our study refers to "Extract Method" refactoring method. After the refactoring is applied, we obtaind a new design of the system, design that will be again evaluated. The metrics values obtained before and after applying this refactoring are comparatively analyzed.

TABLE 1. Project 01 - The list of classes' clusters with the corresponding metrics

Cluster	Class Id	CC	EC	LCOM
1.1	1, 3, 4, 5, 7, 10, 11,15, 16, 17, 19, 21, 23, 24, 25, 26, 27, 28, 30, 31	medium	medium	high
1.2	2, 6, 9, 12, 20, 29	small	small	small
2	8, 13, 14, 32, 18, 22	high	high	high

3. Case Study

A case study was used to validate our metric based approach for identifying God Class design flaw. Two medium size projects, having 32 classes respectively 52 classes, were analyzed. The selected metrics are those described in Section 2.2, following the heuristics that characterize the "God Class" design flaw, i.e. Cyclomatic Complexity (CC), Lack of Cohesion Of Methods (LCOM) and Efferent Coupling at type level (Ce). The metrics were computed using NDepend [10] software.

Therefore, the first two steps (setting the assessment objectives and metrics identification) that define the proposed methodology for OOD assessment (Section 2) have already been performed. We have also mention that the assessment objectives and the selected metrics are the same for both projects.

In what follows we apply Fuzzy Divisive Hierarchic Clustering (FDHC) algorithm [3], computing also the DFE metric value, and we suggest some refactorings to improve the design. After the refactorings were applied, the newly resulted design was also assessed and some comparasions were made.

3.1. **Project 01 - Case Study.**

After applying the FDHC algorithm [3] we have obtained the clusters briefly described in Table 1.

The analysis of the metric's values from each cluster reveals the cluster with "problems" regarding the "God Class" design flaw: in the first cluster, i.e. 1.1 the CC metric and Ce metric have "good" values, but the LCOM value indicate a poorly cohesive class, for the 1.2 cluster all the metric values are "good", indicating that the classes from this cluster are not candidates for the "God Class' design flaw, but the elements from the cluster 2 have the value for the Cyclomatic Complexity metric greater than 30 (being extremely complex), Ce metric is high and the LCOM very closed to 1, indicating poorly cohesive class and have more than one responsibility. Thus, the classes from the 2 cluster are candidates for "God Class" design flaw, being candidates for refactoring.

TABLE 2. Project 01 - Metrics values before and after applying refactoring for the analyzed classes.

	Before			After		
Class	CC	Ce	LCOM	CC	Ce	LCOM
DataServer	37	28	0.86	35	28	0.64
IM	59	43	0.96	44	43	0.91
InvForm	52	67	0.94	51	67	0.94

TABLE 3. Project 02 - The list of classes' clusters with the corresponding metrics

Cluster	Class Id	CC	Ce	LCOM
1.1	3, 24, 25, 26, 29, 44, 45, 47	medium	medium	0
1.2	1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 22, 23, 27, 28, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 46, 48, 49, 50, 51, 52	small	small	0
2.1	20,21,35,40	high	high	high
2.2	10, 18, 19	medium	medium	high/0

The value of the DFE metric is 1.33, the minimum value being 0.4 and the maximum value 1.58, the obtained value suggesting that the "God class" design flaw is largely studded into the software system.

The suggested refactoring are Extract Method and this should be applied for the classes in the 2 cluster.

The Extract Method was applied for the following 3 classes: DataServer, IM and InvForm, respectively 7 methods, and the complexity of some methods/classes was reduced. See Table 2for more details.

3.2. **Project 02 - Case Study.** Regarding the second project we analyzed, after applying the FDHC Algorithm [3] we have obtained the clusters specified in Table 3.

The analysis of the metric's values from each cluster reveals the cluster with "problems" regarding the "God Class" design flaw: in the first (1.1.) and the second (1.2) cluster the values of the metrics don't indicate "God Class" design flaw entities, the third cluster 2.2 has "bad" values for each metric (high values for CC and Ce, and value near 1 for the LCOM metric), cluster 2.2 hhas "goog' value for the Ce metric but "bad" values for the other two

TABLE 4. Project 02 - Metrics values before and after applying refactoring for the analyzed classes.

	Before			After		
Class	Ce	CC	LCOM	Ce	CC	LCOM
LinkMemoDbDataSet	35	59	0.82	37	59	0.83
TableAdapterManager	63	43	0.78	66	43	0.81
tbl_UrlTableAdapter	55	48	0.9	43	48	0.92

metrics, and the cluster with the isolation points, i.e. IP contains classes with "good" values for all the metrics, except for the CC metric (but not very high).

The value of the DFE metric is 1.07, having minimum value 0.33 and the maximum value 1.37, the obtained value suggesting that the "God class" design flaw is largely studded into the software system.

Extract Method refactoring was applied for three classes from the 2.1 cluster: LinkMemoDbDataSet, tbl_UrlTableAdapter, TableAdapterManager. See Table 4for more details.

Analyzing the results obtained after applying the proposed refactoring for the second project, we can conclude that:

- there is only one improvement, the class *tbl_UrlTableAdapter* decreases its complexity
- the value of Ce metric remains unchanged for the refactored classes
- the values of metrics CC and LCOM metrics have higher values (except for *tbl_UrlTableAdapter* class) which does not improve the design.

So, if for the first project we obtained an improvement after applying refactoring, for the second one we have to consider a new analysis to suggest other refactoring.

The decision of choosing a refactoring and the analysis of the impact of applying it, is compromise decision, known in literature as "technical debt" cite Suryanarayana14. Technical debt is a recent metaphor referring to the eventual consequences of any system design, software architecture or software development within a codebase. It is possible that choosing a particular refactoring, to improve some quality attributes, while others remain the same or even worse.

4. CONCLUSION

Software metrics are considered of great importance in software quality assurance but there is a gap between the things measured and the ones really important in terms of quality characteristics. The current paper proposes a

new metric based approach to evaluate the impact of applied refactoring on software desing.

For future work, we intend to focus our research in the following directions:

- to apply the proposed evaluation framework on more case studies;
- to automate the task of establishing the list of suspect entities and the list of refactorings that could be applied.

References

[1] R. Marinescu: *Measurement and quality in object-oriented design*, Ph.D. thesis in the Faculty of Automatics and Computer Science of the Politehnica University of Timisoara, 2003.

[2] Camelia Serban, *A Conceptual Framework for Object-oriented Design Assessment*, Computer Modeling and Simulation, UKSim Fourth European Modelling Symposium on Computer Modelling and Simulation, 90–95, 2010.

[3] Dumitrescu, D., *Hierarchical pattern classification*, Fuzzy Sets and Systems 28, 145–162, 1988.

[4] A.J. Riel. *Object-Oriented Design Heuristics*, Addison-Wesley, 1996.

[5] M. Salehie and Li Shimin and L. Tahvildari *A Metric-Based Heuristic Framework to Detect Object-Oriented Design Flaws*, Program Comprehension, 2006.

[6] M. Fowler and K. Beck and J. Brant and W. Opdyke and and D. Roberts: *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999.

[7] C. Serban and A. Vescan and H.F. Pop *Design Flaw Entropy metric for software quality assessment*, Computer Science - Research and Development (submitted)

[8] S.R. Chidamber and C.F. Kemerer. *A Metric Suite for Object- Oriented Design.* IEEE Transactions on Software Engineering, 20(6):476493, 1994.

[9] B. Henderson-Sellers. *Object-Oriented Metrics-Measures of Complexity.* Prentice Hall, Sydney, 1996.

[10] NDepend, *NDepend, http://www.ndepend.com/*, 2015.

[11] T.J. McCabe. *A Complexity Measure.* IEEE Transactions on Software Engineering, 2(4), pages 308320, 1976.

[12] Arthur H. Watson and Thomas J. McCabe. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric.* In National Institute of Standards and Technology NIST Special Publication, pages 500235, 1996.

[13] Suryanarayana G. *Refactoring for Software Design Smells (1st ed.).* Morgan Kaufmann. p. 258. ISBN 978-0128013977. 2014

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

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

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

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

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

SEX IDENTIFICATION IN ARCHAEOLOGICAL REMAINS USING DECISION TREE LEARNING

IOAN-GABRIEL MIRCEA, GABRIELA CZIBULA AND MARA-RENATA PETRUŞEL

Abstract. We are approaching in this paper, from a machine learning perspective, the problem of detecting the gender of human skeletal remains from bone measurements. The problem of sex identification of human remains is of major importance for bioarchaeologists, since it provides information regarding the characteristics of past societies. For predicting the gender of human skeletons, an inductive learning method based on decision trees will be used. Computational experiments are performed on publicly available archaeological data sets. The obtained results emphasize the effectiveness of the proposed approach with respect to the similar approaches existing in the literature.

1. Introduction

Detecting the gender of human skeletal remains is very important for studying the gender differences in past populations [5]. This contributes to a better understanding of the social position and attributions of each gender in society. The sex identification task is a very delicate one and is highly influenced by the historical period and the geographic origin of the skeleton.

In this paper we are focusing on the problem of gender detection of human skeletal remains from bone measurements. Most of the approaches existing in the literature regarding the gender detection of human skeletons are using statistical methods or are based on bone measurements and DNA or gene analysis. Few computational intelligence techniques have been investigated for detecting the sex of human skeletons [2, 13]. The previous approach from [13] applies CHAID (*CHi-squared Automatic Interaction Detection*), a tree based technique which uses the Chi-square test [7] to determine the best next split at each node in the tree.

Received by the editors: June 3, 2015.

2010 *Mathematics Subject Classification.* 68T05,68T10.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Induction*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *Medicine and science* .

Key words and phrases. bioarchaeology, machine learning, decision tree learning.

We propose in this paper an approach based on an optimized ID3 decision tree learning algorithm for solving the sex detection problem. The main contribution of the paper consists in using a Chi-square pre-pruning technique for reducing the overfitting in the learning process. Three case studies will be used for evaluating the performance of our models and a study towards identifying the best feature set for learning is also conducted. The obtained results emphasize that our approach overperforms an existing approach from the literature based on discriminant function analysis. As far as we know, a decision tree based approach for gender detection similar to ours has not been reported in the literature.

In the case of gender classification we are dealing with real-valued features (bone measurements) and a lot of machine learning algorithms could be used, without requiring the transformation of the initial continuous input space into a discrete one. Still, our main motivation behind choosing the *decision tree* learning is that decision trees provide human-readable rules. A decision tree can be easily converted into set of rules and thus, the obtained results can be easily understood by humans and this way, a feed-back from the bioarchaeologists would be simply obtained. We consider this as a main advantage of our tree based approach for gender detection. Certainly, there are limitations of such decision tree based approaches, such as their sensitivity to errors in the training data (e.g outliers, noise, etc) and their tendency to overfitttting. The occurrence of errors in the training data may be attenuated when the decision tree enables fuzzy logic in the decision process. The overfitting tendency is reduced in the present approach by use of the Chi-squared pre-pruning technique.

The remainder of the paper is organized as follows. Section 2 outlines the fundamentals of decision tree learning. Section 3 introduces our approach for the detection of gender in human skeletons. Experimental evaluations are given in Section 4 and a comparison to similar work from the literature is presented in Section 5. Section 6 presents the conclusions of the paper and mentions future research directions.

2. Background

In this section we are providing a brief background on decision trees. One of the most commonly employed methods for approximating discrete- valued functions is decision tree learning. In this case, the learned function is defined by a decision tree. Furthermore, the decision tree learning method is able to learn disjunctive statements and is robust to noisy data.

When using decision trees, in order to classify instances, these are sorted starting from the root of the tree, until a leaf node, which will specify the

classification of the instance. All the nodes in the tree point out a test of an attribute of the instance, while every branch downward that node correlates to one of the possible values of this attribute [10]. A well-known algorithm for decision tree learning is the ID3 algorithm, being the basis for other known algorithms for building decision trees such as CART (*Classification and Regression Trees* [4]) and C4.5 [11]. The way the ID3 algorithm learns decision trees is by constructing them top-down, starting with the best attribute at the root of the tree, which is selected after each instance attribute is tested using a statistical test. The measure that is commonly used for deciding the best attribute at a given node in the tree is the *information gain* [12]. After that, a descendent of the root node is built for every available value of this attribute. Also, the training examples will be sorted to the suitable descendant node. The whole process is then repeated and this results in a greedy search, that never returns to reconsider choices already made [10].

It has to be mentioned that a decision tree may be viewed as a set of rules, thus the reasoning process (the way the classification of an instance was decided) is available to the user. This makes the *decision tree* learning reliable.

3. Our approach

This section presents our approach on using *decision trees* for sex identification in human remains from the length of long bones of the arm and leg.

Let us consider a data set consisting of human skeletons. Each skeleton from the data set is labeled as **male** or **female**. Each skeleton is characterized by m numerical features representing different measurements that were performed on it. Usually, the measurements correspond to several significant bones in the body. Therefore, an instance (skeleton) may be viewed as an m-dimensional vector.

The first step in building our inductive learning models is the *data preprocessing* step. Since we are dealing with real-valued features (i.e the bone measurements are real values), the data is discretized. The discretization idea is the one indicated in [10] and presumes that a discrete-valued feature will be defined dynamically to divide the continuous attribute value into a discrete set of intervals. For discretizing a particular feature (bone measurement) we are searching for a threshold t which will divide the continuous attribute space into a discrete one. In our approach we selected two numerical values for discretizing each feature. More exactly, the attribute values that are less than t will be replaced with 1 and the other values are replaced with 2. The most convenient value for the threshold t is the one that will produce the greatest information gain of the considered feature.

3.1. **Building the model.** After the data set is pre-processed as indicated above, the inductive learning model will be built during the training step. The classification process takes place in two phases that indicate the ideas of an inductive learning algorithm: *training* and *testing*. In the training phase the inductive model will be built and further applied for classifying an unseen skeleton as part of the testing phase.

For building the *decision tree* (DT), an optimized variant of the ID3 algorithm [11] is used. When building the tree, an heuristic is used in order to stop splitting a node in the tree if this spilt is considered to be unimportant. More precisely, let us assume that a certain node n is built in the tree (using the ID3 algorithm) and S_n is the set of instances corresponding to it (sorted to node n). If the percentage of instances (skeletons) from the set S_n belonging to one of the two classes (male or female) is less than a given threshold τ, then the node n is considered to be a leaf node and it is labeled with the most common classification of instances from S_n.

Besides the heuristic described above, in order to avoid overfitting, a χ^2 *pruning* [7] is used with the scope of reducing the tree while it is built. This is a form of pre-pruning, in which a statistical test is applied to the data at a particular node in the tree, in order to determine if the distribution of classes in the data is or not statistically significant.

The main idea of performing pruning at a particular node n in the tree, i.e to stop growing the tree below n, is to apply the χ^2 test to verify if the feature X corresponding to the node n is uncorrelated with the decision (of splitting the node). If uncorrelated, we expect that the real number of male and female instances at this node to be close to the expected number of male and female instances at the node. For this, we need a measure of "deviation", defined as in Formula (1). The intuition is the following: for each possible value for the feature X (i.e 1 and 2), we compute the subset S_n^1 (the subset of instances from S_n having the value 1 for the feature X) and S_n^2 (the subset of instances from S_n having the value 2 for the feature X).

$$C = \sum_{v=1,2} \left(\frac{(Real\ count_{S_n^v}^{female} - Expected\ count_{S_n^v}^{female})^2}{Expected\ count_{S_n^v}^{female}} \right) .+$$

$$+ \sum_{v=1,2} \left(\frac{(Real\ count_{S_n^v}^{male} - Expected\ count_{S_n^v}^{male})^2}{Expected\ count_{S_n^v}^{male}} \right). \tag{1}$$

In Formula (1), for a value v of the feature X (1 or 2) and for a given class c (male or female), by $Real\ count_{S_n^v}^c$ we denote the number of instances classified with c within the set S_n^v and by $Expected\ count_{S_n^v}^c$ we denote the expected number of instances classified with c within the set S_n^v.

Intuitively, the smaller the value of C is, more likely, the feature X at node n is uncorrelated with the splitting decision. Thus, if C is less than a threshold ϵ, a pruning is performed at the node n (we decide to stop build the tree below the node n).

3.2. **Testing.** After the inductive learning model was built as described above, a testing step is performed to evaluate its performance. When a new instance has to be classified, it is sorted down the DT starting from the root node until a leaf node which gives the classification.

First, the confusion matrix for the two possible outcomes (female and male) is computed. Considering that the "female" class is the positive one and the "male" class is the negative one, the following values are computed: TP - the number of *true positives* (the number of actual positive instances predicted as positive), FP - the number of *false positives* (the number of actual negative instances predicted as positive), TN - the number of *true negatives* (the number of actual negative instances predicted as negative) and FN - the number of *false negatives* (the number of actual positive instances predicted as negative).

Using the values computed from the confusion matrix, two evaluation measures will be further used to test the performance of the DT model. The *accuracy* (denoted by Acc) indicates the percentage of correctly classified instances, i.e $Acc = \frac{TP+TN}{TP+TN+FP+FN}$. The *Area under the ROC curve* measure (denoted by AUC) which is considered one of the best evaluation measures used to compare classifiers [9, 6]. The AUC measure represents the area under the ROC (Receiver Operating Characteristics) curve.

ROC curves can be constructed for classifiers which, instead of directly providing the class of an instance, return a score which may be transformed into a class label using a threshold. A threshold is applied on the continuous output of the classifier and the ROC curve is actually obtained by varying the decision threshold (over a given range). In such cases, for each threshold different (1-$specificity$, $recall$) pairs are obtained, which are represented on the ROC curve. The *recall* of the classifier is computed as the proportion of actual positive instances which are predicted positive, i.e. $recall = \frac{TP}{TP+FN}$. The *specificity* of the classifier represents the proportion of actual negative instances which are predicted negative, i.e. $specificity = \frac{TN}{TN+FP}$.

In case of classifiers which return directly the class, as our decision tree based approach is, the ROC space has a single point. As presented in [6], the ROC curve is obtained by linking the (1-$specificity$, $recall$) point to the points at (0,0) and (1,1). For the constructed curve, the AUC measure can be computed.

Good classifiers have high *accuracy* and *AUC* values. Thus, these measures need to be maximized in order to obtain better classifiers.

For evaluating the performance of the DT model, a *leave-one out* cross-validation was used. *Cross-validation* is a well-known technique used for estimating the generalization error of a classifier [15]. In the *leave-one out* cross-validation on a data set with k instances, k-1 instances are used for training and then the obtained model is tested on the instance which was left out. This is repeated k times and the *accuracy* and the AUC measures are computed as described above.

4. Experimental evaluation

This section contains the experimental evaluation of the DT model (described in Section 3) considering three case studies which were performed on two data sets obtained from the literature [1]. The data set from [1] consists of 200 male and 200 female skeletons from the Pretoria Bone and Raymond A. Dart collections. Ten anthropometric measurements were taken from the radius bone and nine measurements from the ulna bone. The skeletal remains represent black South Africans from the 19^{th} and 20^{th} centuries, born between 1863 and 1996. A statistical analysis of the considered data set has proven that , for each anthropometric measurement, the underlying data follows a normal distribution.

In each data set considered for evaluation, the instances (skeletons) within the data sets are labeled as being **male** or **female**.

The experiments are conducted as follows.

In order to study the influence of the features regarding the performance of the classification tasks, experiments were conducted considering different feature subsets. We used correlation based feature selection. For each feature, the Pearson correlation coefficient [14] between the feature and the target classification output (i.e the gender) is computed. We mention that the correlations have been computed before data discretization.

The features were then sorted in the increasing order of their correlation to the output. The performance of the proposed approach was assessed first on the entire set of ordered features and then subsequently on the set obtained by removing the first feature from the previous set.The assessment process ends when the feature set contains only the two most correlated features.

When building the decision tree, two impurity functions are used to measure the heterogeneity of a set of labeled samples. The first impurity function is the *entropy* and is commonly used in building decision trees. The second impurity function we are using is the *misclassification* function. For a set S of instances (consisting of a males and b females), the *entropy* of S is computed

as $-\frac{a}{a+b} \cdot log_2 \frac{a}{a+b} - \frac{b}{a+b} \cdot log_2 \frac{b}{a+b}$ and the *misclassification* of S is computed as $misclassification(S) = \begin{cases} \frac{b}{a+b} & if \quad a > b \\ \frac{a}{a+b} & otherwise \end{cases}$

The decision tree will be built as described in Section 3.1, considering a value of 0.90 for the threshold τ and a value of 0.80 for the threshold ϵ used for the χ^2 *pruning* step.

4.1. **First case study.** The first case study we are considering for evaluation consists of human remains identified by ten radial measurements. Thus, there are 10 features characterizing the instances within the data set. The features represent the following radial measurements [1]: maximum length of the radius (F1), distal breadth (F2), circumference at the midshaft (F3), sagittal diameter at midshaft (minimum diameter) (F4), transverse diameter at midshaft (maximum diameter) (F5), vertical radial head height (F6), minimum head diameter (F7), maximum head diameter (F8), circumference of the radial (F9) and circumference at the tuberosity (F10). The correlations between the features and the target gender are given in Figure 1. We observe that the features are well enough correlated with the output.

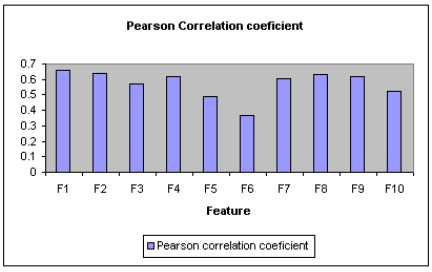


FIGURE 1. Correlations for the features from the first case study

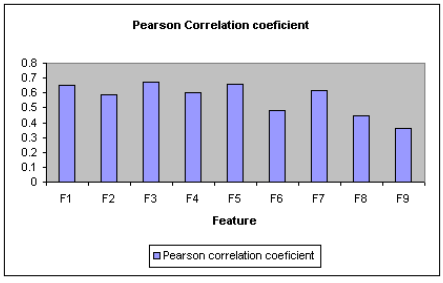


FIGURE 2. Correlations for the features from the second case study

Table 1 presents the results obtained after the experiments performed on the first case study using a *decision tree* constructed with χ^2 pruning. For each experiment we depict the set of features used for classification, the impurity function used for building the tree (*Entropy* and *Misclassification*) and the values obtained for the *Acc* and *AUC* evaluation measures using a *leave-one-out* cross-validation. The best result obtained is marked with bold.

Experiment	Feature set	Entropy		Misclassification	
		Acc	AUC	Acc	AUC
1	$\{6, 5, 10, 3, 7, 9, 4, 8, 2, 1\}$	0.843	0.843	0.835	0.836
2	$\{5, 10, 3, 7, 9, 4, 8, 2, 1\}$	0.843	0.843	0.840	0.841
3	$\{10, 3, 7, 9, 4, 8, 2, 1\}$	0.850	0.850	0.858	0.858
4	$\{3, 7, 9, 4, 8, 2, 1\}$	0.853	0.853	0.853	0.853
5	$\{7, 9, 4, 8, 2, 1\}$	0.853	0.853	0.853	0.853
6	$\{9, 4, 8, 2, 1\}$	0.853	0.853	0.840	0.840
7	$\{4, 8, 2, 1\}$	**0.860**	**0.860**	0.855	0.855
8	$\{8, 2, 1\}$	0.818	0.833	0.803	0.822
9	$\{2, 1\}$	0.825	0.831	0.825	0.832

TABLE 1. Results obtained on the first case study.

4.2. **Second case study.** The second case study consists of nine measurements of the ulna bone from the human skeletons. There are 9 features describing the instances within the data set: maximum length of the ulna (F1), maximum length of the ulna measured using the plumbline geniometer method (F2), anterior-posterior diameter (minimum diameter) (F3), medial-lateral diameter (maximum diameter) (F4), circumference at midshaft (F5), minimum circumference of the ulna (F6), olecranon breadth (F7), minimum olecranon breadth (F8) and height of the olecranon (F9). The correlations between the features and the target gender are given in Figure 2 and show a good correlation with the gender. The results obtained on the second case study are outlined in Table 2 and the best result obtained is highlighted.

Experiment	Feature set	Entropy		Misclassification	
		Acc	AUC	Acc	AUC
1	$\{9, 8, 6, 2, 4, 7, 1, 5, 3\}$	0.868	0.872	0.845	0.847
2	$\{8, 6, 2, 4, 7, 1, 5, 3\}$	0.868	0.872	0.845	0.847
3	$\{6, 2, 4, 7, 1, 5, 3\}$	0.868	0.872	0.843	0.844
4	$\{2, 4, 7, 1, 5, 3\}$	0.875	0.881	0.845	0.847
5	$\{4, 7, 1, 5, 3\}$	**0.878**	**0.885**	0.848	0.849
6	$\{7, 1, 5, 3\}$	0.858	0.858	0.863	0.863
7	$\{1, 5, 3\}$	0.850	0.868	0.855	0.855
8	$\{5, 3\}$	0.838	0.839	0.838	0.839

TABLE 2. Results obtained on the second case study.

4.3. **Third case study.** We considered in this paper, as the third case study, the data set which contains both the radial and ulnar measurements (considered in the first and second case studies). Consequently, in this data set, each

skeleton (instance) will be represented by 19 measurements (features): the first ten are the radial measurements (as in the first case study) and the next nine features represent the ulnar measurements (as in the second case study).

The results obtained on the third case study are given in Table 3. The last line in Table 3 contains the results we have obtained when considering as feature set the union of features which have provided the best results for the first two case studies. One can observe that this set of features provided the best accuracy, using the entropy misclassification function.

Experiment	**Feature set**	**Entropy**		**Misclassification**	
		Acc	**AUC**	**Acc**	**AUC**
1	$\{19, 6, 18, 16, 5, 10, 3, 12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.860	0.860
2	$\{6, 18, 16, 5, 10, 3, 12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.860	0.860
3	$\{18, 16, 5, 10, 3, 12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.873	0.873
4	$\{16, 5, 10, 3, 12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.875	0.875
5	$\{5, 10, 3, 12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.883	0.883
6	$\{10, 3, 12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.883	0.883
7	$\{3, 12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.873	0.873
8	$\{12, 14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.880	0.880	0.875	0.875
9	$\{14, 7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	**0.885**	**0.886**	0.883	0.883
10	$\{7, 17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.870	0.871	0.880	0.880
11	$\{17, 4, 9, 8, 2, 11, 1, 15, 13\}$	0.870	0.871	0.875	0.875
12	$\{4, 9, 8, 2, 11, 1, 15, 13\}$	0.873	0.873	0.873	0.873
13	$\{9, 8, 2, 11, 1, 15, 13\}$	0.863	0.866	0.845	0.845
14	$\{8, 2, 11, 1, 15, 13\}$	0.863	0.866	0.858	0.859
15	$\{2, 11, 1, 15, 13\}$	0.870	0.874	0.865	0.866
16	$\{11, 1, 15, 13\}$	0.863	0.863	0.865	0.871
17	$\{1, 15, 13\}$	0.865	0.871	0.865	0.871
18	$\{15, 13\}$	0.838	0.839	0.838	0.839
19	$\{4, 8, 2, 1, 14, 17, 11, 15, 13\}$	**0.885**	**0.886**	0.873	0.873

TABLE 3. Results obtained on the third case study.

The fact that there are insignificant fluctuations between the best values obtained in the third case study (both when using entropy and misclassification impurity functions) and the ones obtained when considering as feature set the

best values obtained in the first and second case studies, demonstrates once more the precision of our approach on the sex determination problem using decision tree learning method.

5. Discussion and comparison to related work

Table 4 summarizes the *Acc* and *AUC* values (minimum, maximum, mean and standard deviation) obtained using decision tree learning on the case studies considered for evaluation in Section 4. For each case study, the best values for the *Acc* and *AUC* are highlighted. One can see that the best values, for each case study, are obtained using the *entropy* impurity function.

Case study	Impurity function	Evaluation measure	Mean	Min	Max	Stdev
First	Entropy	Accuracy	**0.844**	**0.818**	**0.860**	0.014
First	Entropy	AUC	**0.847**	**0.831**	**0.860**	0.010
First	Misclassification	Accuracy	0.840	0.803	0.858	0.018
First	Misclassification	AUC	0.843	0.822	0.858	0.012
Second	Entropy	Accuracy	**0.863**	**0.838**	**0.878**	0.013
Second	Entropy	AUC	**0.868**	**0.839**	**0.885**	0.014
Second	Misclassification	Accuracy	0.848	0.838	0.863	0.008
Second	Misclassification	AUC	0.849	0.839	0.863	0.007
Third	Entropy	Accuracy	**0.872**	**0.838**	**0.885**	0.011
Third	Entropy	AUC	**0.873**	**0.839**	**0.886**	0.011
Third	Misclassification	Accuracy	0.868	0.838	0.883	0.013
Third	Misclassification	AUC	0.869	0.839	0.883	0.012

Table 4. Obtained results on the considered case studies.

We also note that the best *Acc* and *AUC* values are obtained in the third case study using the *entropy* impurity function. The variations of the *Acc* and *AUC* values obtained for this experiment are depicted in Figure 3, respectively in Figure 4. The small values obtained for the standard deviation of the *Acc* and *AUC* values (0.011) indicate a good precision of the decision tree model.

Due to the fact that the removal of attributes in the evaluation process does not provoke important fluctuations of *Acc* and *AUC* measurements (Figure 3 and Figure 4) in the results obtained, we can conclude that our tree is well constructed from the beginning. The case in which we test only two attributes is an expected exception of the rule above, because a set consisting of only two attributes is definitely too small to obtain good results.

Most of the approaches existing in the literature for determining the sex of skeletal remains are based on bone measurements, statistical methods or DNA and gene analysis. There is only one approach in the literature that uses

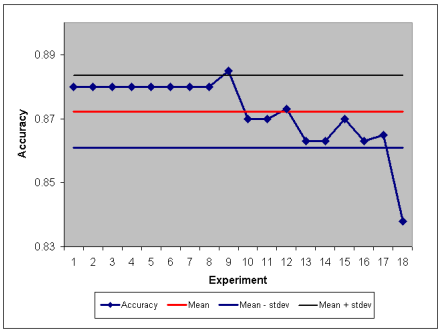


FIGURE 3. *Acc* values obtained on the third case study using the *entropy* impurity function

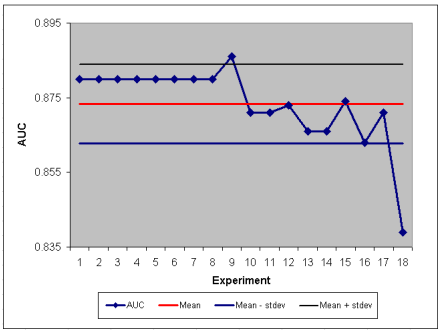


FIGURE 4. *AUC* values obtained for the third case study using the *entropy* impurity function

the same data sets as in our paper, a discriminant analysis method which was introduced in [1]. Five discriminant functions were used in this paper for the data set we have considered in our first case study and four functions were used for the data set considered in our second case study. For estimating the performance of the gender prediction task, only the accuracy is reported in [1], thus we will also use for comparison this evaluation measure.

Table 5 comparatively presents the best *Acc* values (minimum, maximum, mean and standard deviation) reported by our approach and the discriminant analysis method from [1]. The best obtained values are marked with bold. We note that [1] uses the same evaluation method as in our paper, i.e "leave-one-out" cross-validation.

Case study	Classifier	Mean	Min	Max	Stdev
First	Our DT approach	0.844	0.818	0.86	0.014
First	Discriminant functions [1]	0.838	0.81	0.865	0.026
Second	Our DT approach	0.863	0.838	0.878	0.013
Second	Discriminant functions [1]	0.843	0.795	0.875	0.034
Third	**Our DT approach**	**0.872**	**0.838**	**0.885**	**0.011**
Third	Discriminant functions [1]	-	-	-	-

TABLE 5. Comparative results on the considered case studies.

From Table 5 we observe that our DT model is more performant that the discriminant analysis method from [1]. One can easily observe the difference between the maximum value of *accuracy* (0.885) obtained using decision

tree learning method, in comparison with the best result obtained until now in literature using *discriminant functions* [1] (0.875). We mention that our approach achieves best accuracy on the data set containing both radius and ulna measurements using the feature set $\{F14, F7, F17, F4, F9, F8, F2, F11, F1, F15, F13\}$, while the method from [1] uses the entire feature set without any form of feature selection.

Stevenson et al. have approached in [13] the sex prediction problem using CHAID, a type of tree based technique based on the Chi-square test [7] that determines the next best split at each node in the tree. The approach from [13] differs from ours, since the tree is built differently than in our approach. Experiments were performed on 304 remains of Americans, European and African ancestry who died between 1915 and 1955 and accuracies between 85% and 85.5% were obtained. The data set used in [13] is not publicly available, that is why a fair comparison with our approach can not be made. Still, if we look at the obtained accuracies, we observe that our best accuracy exceeds the maximum accuracy reported in [13].

A comparison of our approach to other existing approaches from the literature is hard to be made, since in the existing approaches the experiments are performed on data sets which differ from the one considered in this paper. That is why a comparison that is based only on the obtained accuracies is not relevant, since the data sets used in the experiments are not the same. The good performances of our DT model on the case studies considered in this paper makes us believe that it will perform well when applied on other data sets.

Based on the experimental results we have obtained, we can conclude that decision trees are machine learning models which seem to offer accurate predictions for the gender detection problem. Moreover, when compared to other machine learning models, we consider that decision trees are better alternatives for bioarchaeologists. Decision trees are able to provide a set of rules indicating the way the prediction was made and this would be of great interest for bioarchaeologists.

6. Conclusions and further work

We have proposed in this paper an inductive learning methods for detecting the sex of human remains from bone measurements, which is based on decision trees. The experimental results obtained on three open-source data sets reveal that our approach outperforms similar approaches from the literature.

Further work will be done in order to extend the experimental evaluation of the proposed machine learning based model on real data sets [8] to better investigate their performance. We also plan to investigate the use of *random*

forests [3] and *fuzzy* [16] tree based models, as well as to further consider techniques for feature selection and for data discretization.

Acknowledgments

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS–UEFISCDI, project number PN-II-RU-TE-2014-4-0082.

References

[1] I. L. O. Barrier. Sex determination from the bones of the forearm in a modern South African sample. PhD thesis, University of Pretoria, 2007.
[2] S. Bell and R. Jantz. Neural network classification of skeletal remains. In G. Burenhult and J. Arvidsson, editors, *Archaeological Informatics: Pushing The Envelope. CAA2001*, pages 205–212. Archaeopress, Oxford, 2001.
[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
[4] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Chapman and Hall, 1984.
[5] M. Faerman, D. Filon, G. Kahila, C. L. Greenblatt, P. Smith, and A. Oppenheim. Sex identification of archaeological human remains based on amplification of the X and Y amelogenin alleles. *Gene*, 167(1-2):327–32, 1995.
[6] T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.
[7] P. E. Greenwood and M. S. Nikulin. *A Guide to Chi-Squared Testing*. Wiley Series in Probabilities and Statistics. Wiley, 1996.
[8] Institute of Interdisciplinary Research in Bio-Nano-Sciences. http://bionanosci.institute.ubbcluj.ro/.
[9] N. Lavrac, B. Kavsek, P. A. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.
[10] T. M. Mitchell. *Machine learning*. McGraw-Hill, Inc. New York, USA, 1997.
[11] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
[12] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
[13] J.C. Stevenson, E.R Mahoney, P.L. Walker, and P.M. Everson. Technical note: prediction of sex based on five skull traits using decision analysis (CHAID). *Am J Phys Anthropol*, 139(3):434 – 441, 2009.
[14] S. Tuffary. *Data Mining and Statistics for Decision Making*. John Wiley and Sons, 2011.
[15] G. Wahba, Y. Lin, and H. Zhang. GACV for support vector machines, or, another way to look at margin-like quantities. *Advances in large margin classifiers*, 298–309, 1998.
[16] L. A. Zadeh. A summary and update of "fuzzy logic". In *2010 IEEE International Conference on Granular Computing, GrC 2010, San Jose, California, USA, 14-16 August 2010*, pages 42–44, 2010.

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

E-mail address: {mircea, gabis}@cs.ubbcluj.ro, pmir1335@scs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

A STARUML PLUGIN FOR INCLUDING ASPECTS IN A UML CLASS DIAGRAM

BRISTENA VRÂNCIANU AND GRIGORETA S. COJOCAR

Abstract. Aspect oriented programming (AOP) is a programming paradigm that complements the existing programming paradigms in order to be able to clearly separate all the concerns from a software system in analysis, design and implementation phases. One of the main difficulties when using the aspect oriented paradigm is that the control flow of the system is difficult to follow and understand just inspecting the source code 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. In this paper we propose a set of notations for including aspects in an UML class diagram and we present a StarUML plugin that allows the use of these notations. The aspects, their relationships with other aspects, classes or interfaces, and the visualization of the classes that will be modified dynamically or statically by includind the aspects into the final system can be represented with the plugin. This may ease the understanding of the overall static structure of a software system and may highlight the consequences of adding aspects to a software system.

1. Introduction

Separation of concerns [15] is 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. Most concerns can be clearly separated using just one programming paradigm, however there still are some concerns whose design and implementation are entangled with other concerns. Other programming paradigms, extending the existing ones, have been developed in order to allow better separation of concerns that are still entangled. Aspect oriented paradigm (AOP) is one of these paradigms and it usually extends the object-oriented paradigm [13].

Received by the editors: .June 26, 2015.

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.

Even though there are already a number of aspect oriented languages such as AspectJ [3] and AspectC++ [2] that provide new language constructs for implementing the crosscutting concerns, there is no generally accepted design notation that supports the design of aspect oriented systems.

Having a graphical design notation would make the understanding of an aspect oriented system much easier and it would also serve as a basis for assessing the impact of crosscutting concerns on their base classes (core classes).

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 [5, 17, 22]. 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 [21]. 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 the addition of new modeling elements, modify the specification of the existing ones or change their semantics [5].

The main contributions of this paper are the proposal of a new notation for representing introductions in an UML class diagram and the development of a StarUML plugin that allows developers to include aspects in UML class diagrams. Aspects relationships with other elements from a UML class diagram can also be displayed using the proposed plugin.

The paper is structured as follows. In Section 2 we present the new concepts introduced by the aspect oriented paradigm. The existing proposed notations for including aspects in an UML class diagram are presented in Section 3. The proposed notations are given in Section 4 and the StarUML plugin is described in Section 5. A small example of using the proposed notations is presented in Section 6. Some conclusions and future research directions are given in Section 7.

2. AOP Concepts

In order to design and implement crosscutting concerns, the aspect oriented paradigm introduces new concepts: *join point*, *pointcut*, *advice*, *aspect* and *introduction*, and *weaving* for building the final software system. An important characteristic of aspect oriented programming is that it can only be used 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. In the following the concepts introduced by AOP are briefly presented.

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.

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.

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):

- *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).

- *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, or it may cause multiple executions of the selected join point.

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).

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. In some aspect oriented languages (i.e., AspectJ [3], AspectC++[2]) aspects 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.

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 system, 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. 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.

3. Existing UML-based Notations and Plugins for Aspects

Since the appearance of aspect oriented programming, many attempts to identify an appropriate notation of aspect oriented design have been made. Most approaches (Suzuki and Yamamoto [20], Aldawud et al. [1], Kande et al. [12, 11], Zakaria et al. [23], Pawlak et al. [16], Stein et al. [19], Jacobson and Ng [10], Basch and Sanchez [4], and Zhang [24]) 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. [9]) focused on a particular crosscutting concern and tried to introduce special notations for the elements needed to design that crosscutting concern.

All the proposals have considered a way of representing the aspect concept into the class diagram. Most of them also considered representing the advice [12, 16, 19, 20, 23, 24] and the pointcut [10, 12, 16, 19, 24]. Very few proposals considered representing introductions [10, 19, 20] and even fewer considered representing join points [4, 19]. There is very little consensus related to the appropriateness of a chosen representation for the concepts introduced by AOP:

- *Aspect* - The aspect is usually represented starting from the *Class* classifier enhanced with the *aspect* stereotype [1, 16, 19, 23] or starting directly from the *Classifier* [10, 12, 20]. Only a few proposals considered using a non classifier as a starting element, namely the package with two compartments for pointcuts and advices [4, 24].
- *Join point* - There are only two proposals for representing them, consisting in links [19] and a notation in the form of a circle with a cross inside [4].
- *Pointcut* - There is very little overlapping between the proposals for representing pointcuts. In [12] and [19] pointcuts are represented using the *pointcut* stereotype, while Pawlak et al. [16] propose a representation based on an association from an aspect class towards a classifier stereotype with *pointcut*. Zakaria et al. [23] 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 [24]. Jacobson and Ng [10] proposed representing pointcuts as operations in their own compartment of the aspect stereotype.
- *Advice* - Stein et al.[19] and Zakaria et al.[23] model an advice as an UML operation of a stereotyped named *advice*; while Suzuki and Yamamoto [20] provide a suggestion of using a constraint for the corresponding weave and Kande et al. [12] suggest to represent it as a compartment in the new *aspect* classifier.
- *Introductions* - There is only one approach, proposed by Stein et al. [19], in explicitly modeling introductions that uses parameterized templates. The class extensions compartment introduced by Jacobson et Ng [10] 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).

Only a few proposals (Aldawud et al. [1], Suzuki and Yamamoto [20], Zakaria et al. [23], Kande et al. [12] and Stein et al. [19]) have taken into consideration the relationships that the aspects can have with the other elements from the class diagram (classes, interfaces). Aldawud et al. [1] proposed to use the *control* relationship to represent which other classes the aspect code controls. Suzuki and Yamamoto [20] proposed the usage of the already existing *realize* relationship to represent an aspect and the classes that the aspect affects. Zakaria et al. [23] proposed to use one of their 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. Kande et al. [12] introduced the *binding* relationship to specify what class of objects an instance of the aspect can be bound to. Stein et al. [19] 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.

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 [10] 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. [12] 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 can also be considered as relationships that show the affected parts of the software system.

Plugins. Even though there were many proposals for including aspects into an UML class diagram, only a couple plugins were developed (Suzuki and Yamamoto [20] and Herrero et al. [9]) in order to actually use the proposed notations. These plugins were developed for Rational Rose CASE tool.

4. UML Aspect Notations Proposed for Designing

We consider that 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 information to the developers. Adding aspects to

the class diagram may ease the understanding of the overall static structure of a software system and may highlight the consequences of adding aspects to a software system. The information that should, in our opinion, be represented in a class diagram are:

- 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, 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.

4.1. **Concepts represented.** In the following we describe the notations that we propose to be used to represent the AOP concepts in a class diagram:

- **Aspect**. An aspect is a stereotype of the UML *Class* model element, like in [1]. We use the stereotype <<aspect>> to distinguish between the aspect and base class (see Figure 1).
- **Pointcut**. A pointcut is a stereotype of the UML *Operation* model element, like in [19]. A pointcut is displayed as an operation with the $<< pointcut >>$ stereotype (Figure 1).
- **Advice**. An advice is a stereotype of the UML *Operation* model element. We define a new stereotype for each type of advice: $<< before >>$, $<< after >>$ and $<< around >>$ (Figure 1).
- **Introduction**. Usually, there are two types of introductions that can be performed using AOP: addition of members (attributes or methods) to a class, and the modification of class hierarchy (inheriting from another class or implementing one or more interfaces). We propose the usage of our newly defined $<< introduces >>$ relationship in order

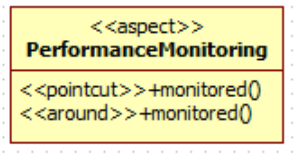


FIGURE 1. UML notation for aspect, pointcut and advice.

to specify that the introduction is done through the aspect. For each situation, the << *introduces* >> relationship is linked with different model elements:

– *Slice.* A *slice* is a stereotype of *Class* model element, and it contains the attributes and the methods added to a class through an aspect, like in AspectC++ [2]. The slice is linked with the class that will contain its members, and with the aspect, as shown in Figure 2. The end of the relationship corresponding to the class is a plus sign to represent that the slice members will be added to it.

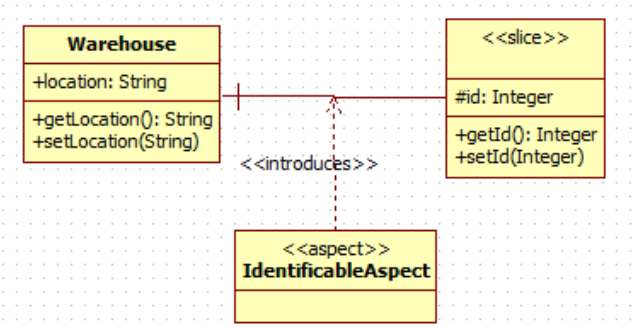


FIGURE 2. Members introduction with slice.

– *Generalization.* The << *introduces* >> relationship is linked with a generalization relationship between two classes to represent that the inheritance is introduced through the aspect at weave time (see Figure 3). If the aspect is removed from the system, the generalization relationship between the two classes will not exist anymore.
– *Interface realization.* The << *introduces* >> relationship is linked with an interface realization relationship between a class and an interface to represent that the realization is introduced through the aspect at weave time (see Figure 4). If the aspect

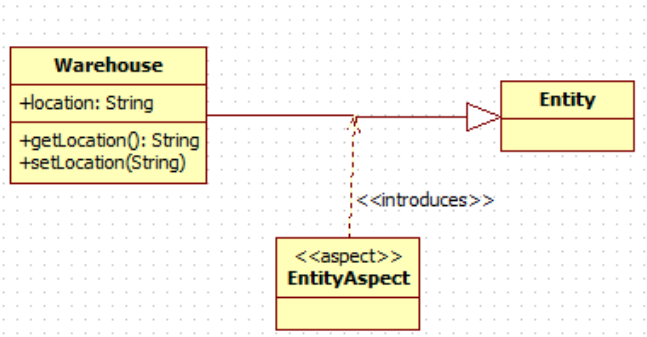


FIGURE 3. Class inheritance introduction.

is removed from the system, the interface realization relationship between the class and the interface will not be defined anymore.

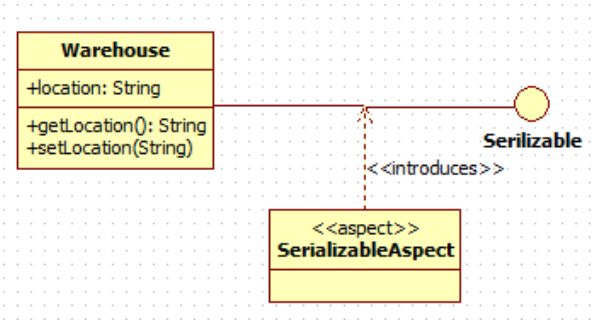


FIGURE 4. Interface implementation introduction.

- **Crosscutting**. A << *crosscut* >> relationship between an aspect and a class is defined, as in [1], to signify that the aspect will modify the behaviour of one or more source code parts from the class (usually methods) through an advice (before, after, or around) (see Figure 5).

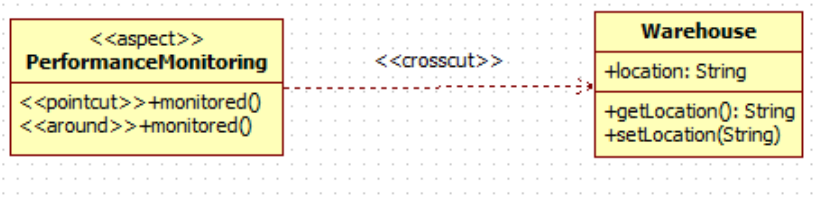


FIGURE 5. Crosscutting relationship

- **Aspect generalization**. An aspect may inherit from another abstract aspect (it may contain abstract pointcuts). A generalization relationship is defined between two aspects (as in Figure 6), with the constraint that the base aspect should be abstract.

FIGURE 6. Aspect generalization

5. StarUML Plugin

StarUML [18] is a software modeling platform that supports UML (Unified Modeling Language). It is an open source software and it provides extensibility, and flexibility. It accepts UML 2.0 notation and it offers many types of diagrams, but you can also create your own type of diagram. StarUML has two important versions:

- StarUML1 which appeared around 1996, and since then suffered many changes until 2005. It was implemented in Delphi, C/C++, JavaScript, C# and has 11 types of diagrams, including: class diagram, use-case diagram, sequence diagram, etc. The 2005 version can still be used today.
- StarUML2, appeared at the beginning of 2015. There were no changes made on the first version from 2005 until 2015. Because this version of StarUML appeared only in 2015, its documentation is not complete, especially the part about extending its features.

We have developed a plugin for StarUML version 1 because it has a developers guide that allows us to make the extensions.

This plugin can be used for representing the set of notations described in Section 4.1: aspects, pointcuts and advice and the relationships between aspects, classes or interfaces. StarUML offers a special concept for creating user defined elements: Notation Extension Technology (NXT) that is a dialect of the Lisp programming language. The profile is implemented in XML.

Code generation. StarUML also offers the possibility of generating an XMI file, that is an XML-based type of file, associated to a diagram. Starting from this file, we can automatically generate the AspectJ (or other aspect oriented language) source code corresponding to the aspects from the diagram and their relationships (*introduces*, *aspect generalization*, etc.). We cannot generate code for all the relationships that can be defined in the diagram. For example, the *crosscut* relationship is too ambiguous to automatically generate code for it. The current version of our plugin can automatically generate source code only for the aspects included in the diagram. The code corresponding to other elements such as classes and interfaces is not generated.

In Listing 1 and Listing 2 are shown the AspectJ automatically generated code corresponding to the aspects from the diagram presented in Figure 6.

```
public abstract aspect AspectTracing{
  public abstract pointcut trace();
  before(): trace(){
  }
  after():trace(){
  }
}
```

Listing 1. AspectTracing generated code.

```
public aspect ApplicationTracing extends AspectTracing{
   public pointcut trace();
}
```

Listing 2. ApplicationTracing generated code.

6. Example

Hannemann and Kiczales [8] have studied the effects of using aspect oriented techniques in the structure of the design patterns introduced by Gamma et al. in [7], and implemented them in Java and AspectJ. Their results have shown that aspect oriented techniques improve the implementation of many patterns. In some cases the improvement is reflected in a new solution structure with fewer or different participants, in other cases, the structure remains the same, only the implementation changes. Patterns assign roles to their participants, for example *Subject* and *Observer* for the *Observer* pattern. These roles define the functionality of the participants in the pattern context. They found that patterns with crosscutting structure between roles and participant classes gain the most improvement. The improvement comes primarily from modularizing the implementation of the pattern. This is directly reflected in the implementation being textually localized. An integral part of achieving

this is to remove code-level dependencies from the participant classes to the implementation of the pattern.

In order to show the usage of the proposed notations and plugin, we use the Weather Monitoring application designed and implemented as an example for the *Observer* design pattern in *Head First Design Pattern* [6]. The Weather Monitoring application tracks from a weather station the current weather conditions (temperature, humidity, and barometric pressure), and provides three display options: current conditions, weather statistics and a simple forecast which are all updated in real time as the station acquires the last measurements. The design of the application without using AOP is shown in Figure 7. The weather conditions are kept in a `WeatherData` object and there are three different displays for each option: `CurrentConditions`, `StatisticsDisplay` and `ForecastDisplay`. The `WeatherData` is the subject and the three display options are the observers from the *Observer* pattern.

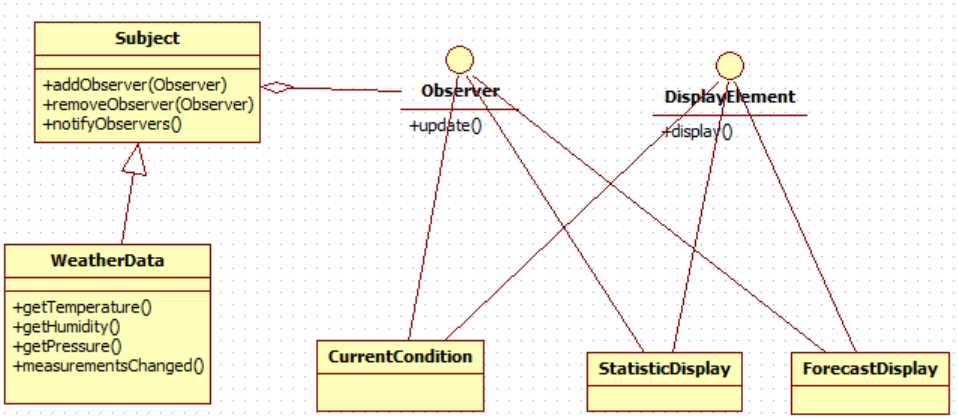


FIGURE 7. Weather Monitoring architecture with Observer pattern.

The AOP implementation of the *Observer* pattern is similar with the one described in [14] and removes the dependency between the concrete subject class and the abstract `Subject` class, and the dependency between the concrete observers and the `Observer` interface from the source code. The dependencies are made using introductions in a separate aspect. The new architecture of the application using AOP is shown in Figure 8 (the methods from the classes are not shown).

The diagram shown that the dependencies between `CurrentConditions`, `StatisticsDisplay` and `ForecastDisplay` and the `Observer` are introduced by the `ObserverAspect`. Without the aspect, these dependencies do not exist

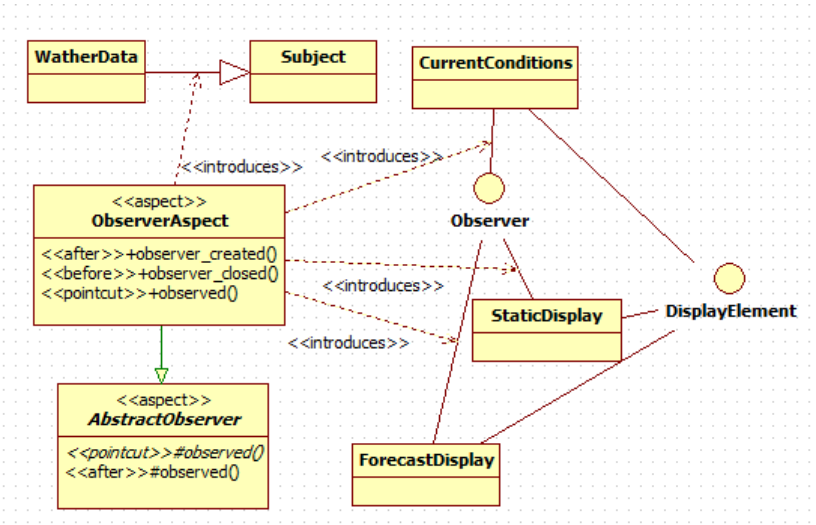


FIGURE 8. Weather Monitoring architecture with AOP based Observer.

anymore. The `WeatherData` generalization of the `Subject` class is also introduced by the aspect. The `ObserverAspect` also crosscuts the `WeatherData` class, as each time the `measurementsChanged` method is executed, an advice is executed that will automatically notify the observers.

All these changes to the static structure of the Weather Monitoring application can be vizualized from the AOP-enhaced class diagram.

7. Conclusions and Further Work

We have proposed in this paper a set of notations to be used in order to model aspects in a UML class diagram. Some notations were previously introduced by other authors, and a few of them are newly defined (the *slice*, the *introduces* relationship, *aspect generalization*, etc.). These notations can ease the understanding of an AOP-based software system by graphically representing the parts that will be affected by introducing the aspects into the final software system. The notations were included in a StarUML plugin that we have developed in order to ease their usage and acceptance. We have also presented an example of how these notations can be used for modeling the AOP-based architecture of a small application.

Further work should be done in the following directions:

- To consider other AOP elements that should be represented in the diagram, like the precedence relationship between two aspects.

- To propose a more general version of the *crosscut* relationship, as for large software systems it may be difficult the vizualize all the classes that will be modified by the aspect. For *Logging* crosscutting concern which affects a big number of classes from the software system, the usage of the *crosscut* relationship may burden the understanding of the design.

References

[1] 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.

[2] AspectC++ Homepage. http://www.aspectc.org/.

[3] AspectJ Project. http://eclipse.org/aspectj/.

[4] Mark Basch and Arturo Sanchez. Incorporating aspects into the uml. In *Proceedings of the Aspect Oriented Modeling Workshop at AOSD*, 2003.

[5] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide (2nd Edition)*. Addison-Wesley Professional, 2005.

[6] Elisabeth Freeman, Eric Freeman, Bert Bates, and Kathy Sierra. *Head First Design Patterns*. O' Reilly & Associates, Inc., 2004.

[7] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley, March 1995.

[8] Jan Hannemann and Gregor Kiczales. Design Pattern Implementation in Java and AspectJ. In *OOPSLA '02: Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 161–173, New York, NY, USA, 2002. ACM Press.

[9] 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.

[10] Ivar Jacobson and Pan-Wei Ng. *Aspect-Oriented Software Development with Use Cases*. Addison Wesley, 2004.

[11] 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.

[12] 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.

[13] 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.

[14] Russell Miles. *AspectJ Cookbook*. O'Reilly, March 2004.

[15] David L. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15(12):1053–1058, December 1972.

[16] 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.

[17] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[18] StarUML - The Open Source UML/MDA Platform. http://staruml.sourceforge.net/v1/about.php.

[19] Dominik Stein, Stefan Hanenberg, and Rainer Unland. An UML-based Aspect-Oriented Design Notation for AspectJ. In *AOSD 2002*, pages 1–7, 2002.

[20] 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.

[21] UML 2.4.1 Superstructure. http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/.

[22] Unified Modeling Language(UML). http://www.uml.org/.

[23] 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.

[24] 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. Kogalniceanu Street, Cluj-Napoca, Romania

E-mail address: vrancianu_bristena@yahoo.com

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

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

DOMAIN MOBILE AMBIENTS FOR NETWORK ROUTING SCHEME

GABRIEL CIOBANU AND DAN COJOCAR

Abstract. Ambient calculus is a calculus for mobile computing introduced to describe the movement of processes or devices. Until now the domain capability of an ambient was defined like a static resource. In this paper we add the composition and choice operations for domain attribute in order to express different addressing and routing schemes. The new formalism can be used to easily describe routing schemes like anycast and multicast.

1. Introduction

Ambient calculus was introduced by Cardelli and Gordon in 1999 as a need to express the movement of processes or devices, even between different administrative domains [1]. The ambient calculus differs from other formalisms such as π-calculus [2] in such that the computational model is based on *movement* instead of *communication*. An ambient represents a unit of movement. Ambient mobility is controlled by the capabilities like: in, out, and open. Mobile ambient capabilities are similar to prefixes in CCS [3] and π-calculus. Several variants of the ambient calculus have been proposed by adding and/or removing features of the original calculus [4, 5, 6].

In [1] the definition of mobile ambients is related to network communication. Aman and Ciobanu proposed a new formalism named Timed Mobile Ambients (*tMA*) that adds timers to capabilities and ambients in order to express timeouts in network communication [7]. In [8], Ciobanu extended the formalism for mobility with timers by adding *capacity*, *weight* and *domain* as ambient attributes.

In this paper we extend the use of composition and choice operations, that were defined in [1] for process operations, to be used on *domain* attribute

Received by the editors: August 29, 2015.

2010 *Mathematics Subject Classification.* 14D15,68M14.

1998 *CR Categories and Descriptors.* D.2.4 [**SOFTWARE ENGINEERING**]: Software/Program Verification – *Formal methods*; C.2.4 [**COMPUTER-COMMUNICATION NETWORKS**]: Distributed Systems – *Distributed applications*.

Key words and phrases. formal methods, mobile ambients, DNS lookup.

also. Using these two operations we are able to express routing and addressing scheme like: unicast, anycast and multicast.

The rest of this paper is structured as follows. Section 2 introduces the mobile ambients and Coordination of mobile agents formalism. In Section 3 we present the formalism changes and the reductions rules that address the domain attribute, and how to apply the changes using a DNS anycast sample. Conclusions and further works are presented in Section 4.

2. Mobile Ambients

Cardelli and Gordon introduced the mobile ambients with the main purpose to model the mobility of agents that are involved in distributed computing [1]. An ambient was defined as a boundary place where computation will happen (laptop, web-page, network routing equipments, etc.). Moving operations, like *go*, *in* and *out* were defined in order to express the movement between ambients. An ambient has a name, a collection of local capabilities which control the ambient, and a collection of subambients.

2.1. **Formal Syntax.** The following table describes the syntax of coordinated mobile ambients.

Table 1. Coordinated Mobile Ambient Syntax

n,m,p		ambient names	P,Q	::=	processes
C	::=	capabilities		0	inactivity
	in n	can enter n		C.(P,Q)	movement
	out n	can exit n		$(n^{\Delta t}_{(l,h,d)}[P], Q)$	ambient
	open n	can open n		P|Q	composition
	go y	migration to y		P+Q	choice
				$M^{\Delta t}.(P,Q)$	movement
				(vn)P	restriction
				*P	replication

A migration *go y* changes the domain d to a domain y. A capability $open^{\Delta t}n.(P,Q)$ evolves to P whenever in Δt the process becomes sibling to an ambient n ; otherwise evolves to Q. An output action $! < m >^{\Delta t} .(P,Q)$ evolves to P whenever in Δt the process becomes sibling to a process which is intending to capture the name m; otherwise evolves to Q. More details about time-stepping function and semantics of coordinated mobile ambients can be found in [8].

3. Domain Mobile Ambients Behavior

In this section we present our proposal to extend the model presented in [8]. By overloading the domain attribute, in order to enhance the local knowledge of an ambient, we can encapsulate the domain choice of an action. We will be able to express or formalize all sorts of actions and mechanism, like: the choices that a student has when leaving the university location. We can imagine that the student has the choice to visit the following locations: the campus, a library or a local restaurant. Also we could easily formalize TCP/IP unicast, anycast, multicast and broadcast routing schemes [9].

By overloading the domain attribute we can rewrite existing migration rules where the domain is the choice. This way we will be able to write more concise rules.

In the student case, the choice is limited to a single option, the student is not able to visit multiple locations at the same moment. But we can imagine other situations, where a choice can result in multiple parallel action. When the same subject has the choice and the ability to perform the same action on different locations at the same time. Such an example could be a beam of light hitting the surface of a lake, a part of the beam will be refracted and the rest will be reflected.

With different LAN topologies, interconnected by network equipments and all sort of link types, the Internet is a giant graph [10]. In this way all routing problems can be reduced to graph search algorithms. Each vertex (router) in these graphs are maintaining tree structures with informations about other vertexes [11], in order to quickly adapt to network failures.

To be able to apply different classic graph search algorithms, like Dijkstra's algorithm [12] and others, to anycast and multicast schemes, we need a way to easily represent different types of endpoints.

Since the differences between the routing schemes are primary on the endpoint levels, we propose to add composition and choice operations to the domain attribute to be able to formalize all the defined routing schemes.

3.1. **Extending Domain Attribute of an Mobile Ambient.**

To represent an endpoint group, we use the notation d_m^n, where:

- m - represents the total number of nodes in the domain.
- n - represents the number of nodes that are used.

Using the above notations for each routing scheme we can define the following endpoint groups:

- *unicast group* $= d_1^1 \equiv d_1$ - the client request will be routed to the single node that represents the unicast group.

- *anycast group* $= d_m^1 \equiv d^1$ - the client request will be routed to only one node that is part of the anycast group. Here the anycast group has m nodes, but in this case we will contact only a single node.
- *multicast group* $= d_m^n \equiv d^n$ - the request will be routed to n destinations from the total of m nodes of a network. n represents the total number of nodes in this multicast group, n being less then m.
- *broadcast group* $= d_m^m \equiv d_m$ - the request is routed to all the available nodes, n from the multicast group is equal to m.

For each routing scheme the domains are defined using composition or choice operations:

- *unicast group* - will use the following domain:

$$d_1 = d_i$$

  where i is between 1 and m, meaning that the domain d contains only a single subdomain.
- *anycast group* - will use the following domain:

$$d^1 = d_1 + d_2 + ... + d_m$$

  meaning that the domain d^1 contains the d_m subdomains and a packet send to this domain will go to only one of the defined subdomains.
- *multicast group* - will use the composition operation instead of choice and it will have the following form:

$$d^n = d_1|d_2|...|d_n.$$

  A packet send to a d^n domain will go to all n defined subdomains. Here d_1, d_2, ..., d_n are part of d_1, d_2, ...,d_m network.
- *broadcast* - similar to multicast will have the following domain:

$$d_m = d_1|d_2|...|d_m.$$

Considering the above domain definitions we can define an ambient with an unicast domain in the following way:

$$(n^{\Delta t}_{(l,h,d_1)}\,[P]\,, Q) \equiv (n^{\Delta t}_{(l,h,d_i)}\,[P]\,, Q)$$

where i is one of the subdomains.

Also an ambient with anycast domain has the following form:

$$(n^{\Delta t}_{(l,h,d^1)}\,[P]\,, Q) \equiv (n^{\Delta t}_{(l,h,d_1+d_2+...+d_m)}\,[P]\,, Q)$$

and it will use choice operation on domain field.

Likewise an ambient with multicast or broadcast domain will be represented as follows:

$$(n^{\Delta t}_{(l,h,d^n)}\,[P]\,, Q) \equiv (n^{\Delta t}_{(l,h,d_1|d_2|...|d_n)}\,[P]\,, Q)$$

and it will use composition on domain field.

The reduction rules for migration *go y* operations on ambients with unicast, anycast, multicast or broadcast groups, are presented next:

- Unicast domain:

$$\frac{t^{'}=0}{(n^{\Delta t}_{(l,h,d)}\left[go^{\Delta t'}d_1.P\right],Q)\rightarrow(n^{\Delta t}_{(l,h,d_1)}\left[P\right],Q)}\equiv$$

$$\frac{t^{'}=0}{(n^{\Delta t}_{(l,h,d)}\left[go^{\Delta t'}d_1.P\right],Q)\rightarrow(n^{\Delta t}_{(l,h,d_1)}\left[P\right],Q)}$$

- Anycast domain:

$$\frac{t^{'}=0}{(n^{\Delta t}_{(l,h,d)}\left[go^{\Delta t'}d^1.P\right],Q)\rightarrow(n^{\Delta t}_{(l,h,d^1)}\left[P\right],Q)}\equiv$$

$$\frac{t^{'}=0}{\substack{(n^{\Delta t}_{(l,h,d)}\left[go^{\Delta t'}d^1.P\right],Q)\rightarrow\\(n^{\Delta t}_{(l,h,d_1)}[P],Q)+(n^{\Delta t}_{(l,h,d_2)}[P],Q)+...+(n^{\Delta t}_{(l,h,d_m)}[P],Q)}}$$

- Multicast/Broadcast domain:

$$\frac{t^{'}=0}{(n^{\Delta t}_{(l,h,d)}\left[go^{\Delta t'}d^n.P\right],Q)\rightarrow(n^{\Delta t}_{(l,h,d^n)}\left[P\right],Q)}\equiv$$

$$\frac{t^{'}=0}{\substack{(n^{\Delta t}_{(l,h,d)}\left[go^{\Delta t'}d^n.P\right],Q)\rightarrow\\(n^{\Delta t}_{(l,h,d_1)}[*P],Q)|(n^{\Delta t}_{(l,h,d_2)}[*P],Q)|...|(n^{\Delta t}_{(l,h,d_n)}[*P],Q)}}$$

3.2. **Anycast DNS Lookup.** Sarat and Terzis in [13] have measured that up to 55% of the DNS queries are answered by the topologically closest server when using anycast addresses. Using the above proposed domain attribute changes we can express the workflow of such DNS queries.

A scenario where a client that performs DNS queries, using two public DNS servers that are scattered between three locations, is presented in Figure 1. Because we are using anycast addresses the queries will be handled by the closest server [13]. For example when using 8.8.8.8 address we are only two hops away from server A, compared to three hops to the next nearest server. If we use 8.8.4.4 address the closest server will be server D.

Using mobile ambients, with the proposed domain attribute changes, we are able to express in a concise manner how our client is performing such a DNS lookup query. On the client configuration we have two nameservers 8.8.8.8 and 8.8.4.4. When performing a DNS lookup query the following steps are performed:

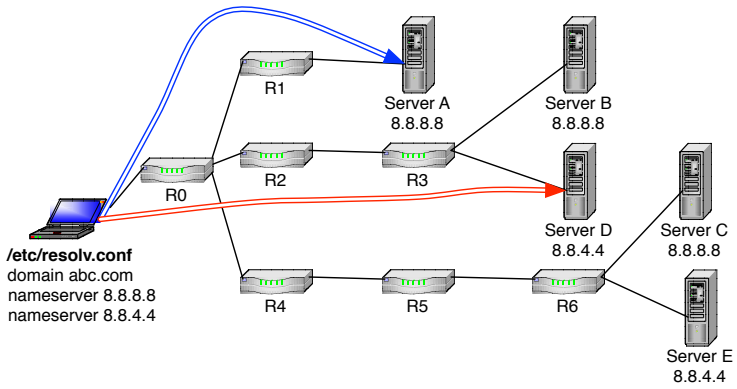


FIGURE 1. DNS lookup using anycast address.

- *Step 1.* - the client is using the 8.8.8.8 address because this is the first configured nameserver, for his DNS lookup query.
- *Step 2.* - the request is processed by $R0$ router. $R0$ knows that server A, B and C all have the 8.8.8.8 anycast address configured, and because the server A is the closest server will forward the DNS lookup to router $R1$.
- *Step 3.* - the router $R1$ is simply forwarding the client request to server A.
- *Step 4.* - server A is processing the client request and responds with a corresponding message.

If one of the above steps the connection is lost, a timeout is reached or the server A is responding with an error, the client restarts the process and performs the same steps using 8.8.4.4 address. Using the second address the $R0$ router is forwarding the request to server D network. If the failed scenarios is repeated in this case too the client will end up deciding that he is not able to resolve the DNS lookup and an error message will be presented to the user [14].

Considering the DNS lookup with mobile ambients we can express the above logic like this:

$$dns\_lookup := (dns\_request^{\Delta t_1}_{(l,h,d^1)}\,[send\_request]\,, process\_error)$$
$$send\_request = send^{\Delta t_2}\,\left[out^{\Delta t_3} client.go^{\Delta t_4}.in^{\Delta t_5} server\right]$$
$$process\_error = (dns\_request^{\Delta t_6}_{(l,h,d^2)}\,[send\_request]\,, display\_error)$$
$$display\_error =! < error\_message >^{\Delta t_7}$$

Where d^1 has the 8.8.8.8 address associated, the first configured nameserver, and d^2 is using 8.8.4.4 address, the failover nameserver, used if an error is received from the first *dns_request* query. If after the second attempt we received an error, a failure message is displayed to the client.

When receiving a request, a router R is performing the following actions:

$$\begin{aligned} process :&= (router^{\Delta t}_{(l,h,d)} [forward\_request], discard\_request) \\ forward\_request &= send^{\Delta t_2} \left[out^{\Delta t_3} router.go^{\Delta t_4}.in^{\Delta t_5} next\_hop\right] \\ discard\_request &= send^{\Delta t_6} \left[out^{\Delta t_7} router.go^{\Delta t_8}.in^{\Delta t_9} client\right] \\ next\_hop &= send^{\Delta t_{10}} [out^{\Delta t_{11}} router.go^{\Delta t_{12}}.in^{\Delta t_{13}} forward\_request \\ &\quad + server] \end{aligned}$$

The next hop from a router could be the destination server or another router. For each router we are doing the same action: check to see if we can forward the request to one of our servers or to a next router. If the Δt expires, in our case the TTL value id decremented and reaches zero, the router will discard the request and is sending an error code to the client [15].

When everything is working properly and the request is received by the proper server, the service is resolving the request and sends a response back to the client.

$$\begin{aligned} server &= open^{\Delta t_{14}} \left[request^{\Delta t_{15}} | response^{\Delta t_{16}}\right] \\ response &= send^{\Delta t_{17}} \left[out^{\Delta t_{18}} server.go^{\Delta t_{19}}.in^{\Delta t_{20}} client\right] \end{aligned}$$

Because an anycast address is used, on router $R0$, our *process* action is defined in the following way:

$$process := (R0^{\Delta t}_{(l,h,R1+R2+R3)} [forward\_request], discard\_request)$$

Based on the knowledge that the $R0$ has, it is able to decide that the $R1$ is the "nearest" router that he needs to forward the client request [13]. $R0$ maintains a list of anycast servers, using the routing informations from $R1$, $R2$ and $R3$. In this case the $forward\_request$ action is defined like this:

$$forward\_request = send^{\Delta t_2} \left[out^{\Delta t_3} R0.go^{\Delta t_4}.in^{\Delta t_5} R1\right]$$

4. Conclusion and Further Works

Mobile ambients are a powerful formalism that can be used to express network protocols and all communications between different devices. In this paper we added the composition and choice operations for domain attribute in order to express different addressing and routing schemes. It offers a way

to easily formalize all the routing schemes defined in IPv4, IPv6 and describe routing schemes like anycast, multicast and broadcast. The evolution of ambients is illustrated with an example from real life, DNS routing scheme. In this example we can see how easily is to express the complex operations that are involved when making a DNS request and also how the fallback logic is triggered, when multiple servers are defined. By using this formalism we were able to focus on the big picture and also capture the domain choices.

Using the proposed changes we were also able to write more concise and compact expressions when describing complex network algorithms where the need of multiple domain choices are involved. Having a concise way to express the movement is helping in validating and comparing classic routing algorithms on all type of networks, independent of the used addressing scheme. The formalism can be used to express all sorts of interactions not only the network related ones. Anywhere we have one-to-one, one-to-(one-of-many) or one-to-many relations, between entities that are interacting somehow, we can formalize the conversation.

As further work, we would like to express the interaction between the processes that are happening in a distributed network, like load balancing or map reduce, where multiple queries are sent to different nodes in order to establish the answer, or a partial answer, as quickly as possible. Also we are further considering to tackle and formalize the movement of particles when two object are colliding.

References

[1] L. Cardelli, A. D. Gordon, and G. Ghelli, "Mobility types for mobile ambients," in *ICAL '99: Proceedings of the 26th International Colloquium on Automata, Languages and Programming.* London, UK: Springer-Verlag, 1999, pp. 230–239.

[2] R. Milner, *Communicating and mobile systems: the π-calculus.* New York, NY, USA: Cambridge University Press, 1999.

[3] ——, *Communication and concurrency.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[4] M. Bugliesi, G. Castagna, and S. Crafa, "Boxed ambients," in *In Proc. TACS 2001, LNCS 2215.* Springer-Verlag, 2001, pp. 38–63.

[5] F. Levi and D. Sangiorgi, "Controlling interference in ambients," in *POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages.* New York, NY, USA: ACM, 2000, pp. 352–364.

[6] D. Teller, P. Zimmer, and D. Hirschkoff, "Using ambients to control resources," *Int. J. Inf. Secur.*, vol. 2, no. 3, pp. 126–144, 2004.

[7] B. Aman and G. Ciobanu, "Timed mobile ambients for network protocols," in *FORTE '08: Proceedings of the 28th IFIP WG 6.1 international conference on Formal Techniques for Networked and Distributed Systems.* Berlin, Heidelberg: Springer-Verlag, 2008, pp. 234–250.

[8] G. Ciobanu, "Coordinated mobile agents," *Proceedings of the Romanian Academy Series A-Mathematics Physics Technical Sciences Information Science*, vol. 10, no. 1, pp. 73–79, 2009.
[9] J. Postel *et al.*, "Rfc 791: Internet protocol," 1981.
[10] Z. G. Daniel, D. R. Figueiredo, S. Jaiswal, and L. Gao, "On the hierarchical structure of the logical internet graph," in *in Proc. SPIE ITCOM*, 2001, pp. 208–222.
[11] M. Thorup and U. Zwick, "Compact routing schemes," in *SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures.* New York, NY, USA: ACM, 2001, pp. 1–10.
[12] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
[13] S. Sarat, V. Pappas, and A. Terzis, "On the use of anycast in dns," in *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems.* New York, NY, USA: ACM, 2005, pp. 394–395.
[14] P. Mockapetris, "Rfc 1034: Domain names: concepts and facilities (november 1987)," *Status: Standard*, 2003.
[15] M. Allman, V. Paxson, and W. Stevens, "Rfc 2581: Tcp congestion control," 1999.

Romanian Academy, Institute of Computer Science, Blvd. King Carol I, Iaşi 700505, Romania
E-mail address: gabriel@info.uaic.ro

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

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 2, 2015

ENVIRONMENT MODEL-BASED TESTING OF REACTIVE SYSTEMS: A CASE STUDY ON A SCADE MODEL

ANNAMÁRIA SZENKOVITS

Abstract. Model-based testing can facilitate automatic test generation, thus, it can significantly decrease testing costs. This paper presents a case study where model-based testing was performed on a SCADE (Safety Critical Application Development Environment) system. Test inputs were generated automatically based on a non-deterministic, realistic environment model expressed in Lutin. The goal of the case study was to investigate whether such a realistic test environment can increase model and oracle coverage. The main contribution of the paper consists in filling in a gap in the existing literature, since there are no other works available discussing both the model and oracle coverage obtained with Lutin on a SCADE system.

1. Introduction

Model-based testing is a widely used technique in the field of verification of reactive systems. The technique consists of five main steps [16]:

(1) The system under test (SUT) and/or its environment is modelled.
(2) Abstract test cases are generated from the model — usually automatically or semi-automatically.
(3) The abstract tests are concretized in order to make them executable.
(4) The tests are executed on the SUT and verdicts are assigned (fail/pass) by the oracle.
(5) The results are analysed.

In our case study, we performed environment-model based testing — according to the above mentioned five steps — on a reactive system. The main characteristics of reactive systems is that they are in continuous interaction with their environment. During their lifecycle, they continuously read the sensor data coming from the environment, update their internal state, then write

Received by the editors: July 14, 2015.

2010 *Mathematics Subject Classification.* 68N30, 68T35.

1998 *CR Categories and Descriptors.* D.2.5 **[Software Engineering]**: Testing and Debugging – *Testing tools (e.g., data generators, coverage testing)*;

Key words and phrases. Model-based testing, Reactive systems, SCADE.

the outputs to their actuators, by which they act upon their environment. The behaviour of reactive systems can be best described using synchronous languages [7].

The model of the SUT upon which our case study was based, was designed and developed in SCADE [5]. SCADE is a synchronous, widely used industrial toolset, especially for designing avionic and railway systems. For modelling the environment of the SUT, the Lutin language [15] was chosen. Similar to SCADE, Lutin is also a synchronous language. In addition to SCADE, Lutin's syntax contains elements that enable us to describe non-deterministic behaviour, thus, to express the logic of the test environment in a more realistic way.

In order to automatically assign verdicts (fail/pass) to the execution of a test case, but also to calculate the oracle coverage rate, we used the Lustre language, another synchronous language, based on the data-flow notation [3, 7]. Lustre is also the kernel of SCADE and Lutin. To analyse the coverage of the SUT achieved during the testing, we used the SCADE Suite MTC (Model Test Coverage)[1] tool.

The paper is structured as follows. Section 2 presents the components of the testing framework. Part 2.1 describes some of the fundamental aspects of the language Lutin, focusing on how different properties of the language will be exploited in our case study. Parts 2.2 and 2.3 explain the MTC and oracle coverage criteria, respectively. Section 3 summarizes the experimental results, while section 5 presents the conclusions and future work. Finally, section 4 reviews some of the work relevant for this topic.

2. Environment-model based testing of reactive systems

The testing framework used in the current case study consisted of the following components, as illustrated by figure 1: the SUT, environment model, oracle and MTC analyser.

The SUT on which the testing was performed was the the SCADE model of an airplane's roll control system. More precisely, we used the C code generated from the SCADE model by the SCADE Suite KCG code generator[2] and executed the test cases on it.

To model the environment, but also to simulate the SUT and to generate test inputs, we used the Lutin language. This way, we were able to generate realistic test inputs and achieve a high model and oracle coverage.

[1] http://www.esterel-technologies.com/wp-content/uploads/2013/02/SCADE-Suite-MTC.pdf, downloaded on May 26, 2015

[2] http://www.esterel-technologies.com/products/scade-suite/automatic-code-generation/scade-suite-kcg-ada-code-generator/, downloaded on May 26, 2015

The test decision was performed automatically be the test oracle. Its role was to decide whether the SUT generated the right outputs for the given inputs. In addition, the oracle also contained coverage criteria based on which we analysed the oracle coverage rate. The language used for formalizing the oracle and oracle coverage criteria was Lustre. Finally, we used the SCADE Suite MTC tool to analyse how thoroughly the C code generated from the SCADE model was executed.

As illustrated in figure 1, the SUT and environment were in continuous interaction. The SUT read the inputs from the Lutin environment, updated its internal state, and generated some outputs. This cyclic behaviour continued for a specified number of iterations. Meanwhile, the oracle observed the behaviour of the SUT and decided whether the outputs generated by the SUT matched the expected outputs for the given inputs. The SUT, environment and oracles were connected by the Lurette tool[3], an automatic test generator for reactive systems. Lurette automated the test decision and stimulation of the SUT, by executing the Lutin code and feeding in the inputs generated to the SUT. In addition, Lurette also computed the oracle coverage [9].

Further on in this section, we present more in detail the components of the testing framework.

2.1. **Realistic test cases with Lutin.** By using Lutin for modelling the test environment, we were able to express non-deterministic behaviour, and to perform guided random exploration of the environment state space. Because in industrial, real-life problems the environment is often underspecified, the non-deterministic nature of Lutin makes it suitable to model realistic test scenarios.

The Lutin language is based on the use of constraints, which represent descriptions of the environment and the expected properties of the SUT. The constraints can be both boolean and numerical [14]. The constraint solver of Lutin solves the constraints and randomly selects some of the solutions [10]. This way, a random exploration of the environment is performed.

Furthermore, Lutin enables us to influence the random exploration, and thereby, to perform guided random exploration. There are two main elements in Lutin's syntax that enable us to realize this. On one hand, we have the non-deterministic *choice operator* |, as illustrated in the code example from Figure 2. Weights can be assigned to the branches of this choice operator, this way we can influence how the environment reacts. On the other hand, non-determinism can be expressed in Lutin with *random loops*, which are defined in terms of expected number of iterations. Based on Raymond et al. [15], there are two possibilities to express the expected number of iterations:

[3]http://www-verimag.imag.fr/Lurette,107.html?lang=, downloaded on May 26, 2015

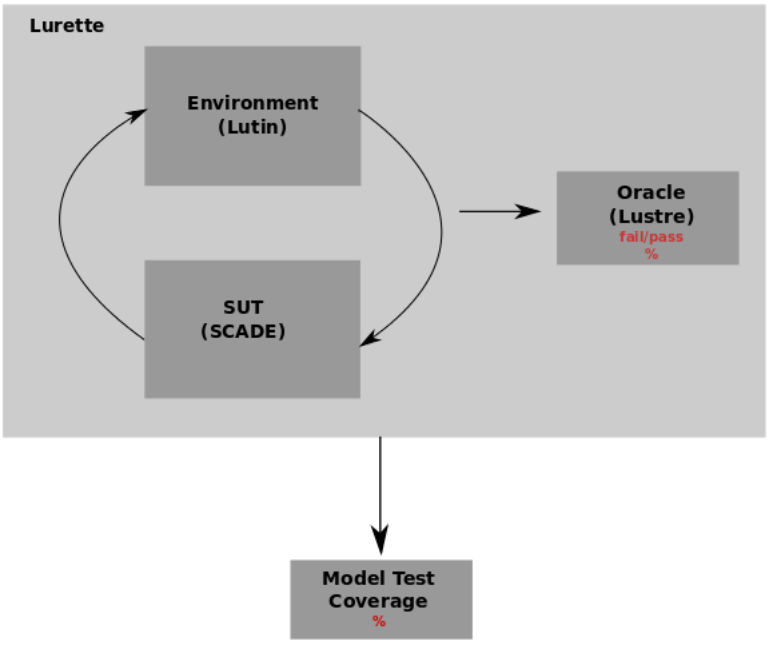


FIGURE 1. Components of the test framework: The **SUT** as a C code generated from a SCADE model with SCADE Suite KCG, the **environment** model expressed in Lutin, the **oracle** formulated in Lustre, and the **MTC analyser**. The SUT and environment are in continuous interaction with each other and have a cyclic behaviour. The execution of the SUT and environment, as well as the automation of the test decision and computation of the oracle coverage is done by the Lurette tool.

(1) `loop[min,max]`: the number of iterations should be between the constants min and max.
(2) `loop~av:sd`: the average number of iteration should be `av`, with a standard deviation `sd`.

The above mentioned control structures were used to create an environment model for the SCADE SUT. The environment model is presented in detail in section 3.

2.2. **Model test coverage.** In order to evaluate the performance of the automated test method, the SCADE Suite MTC tool was used. The MTC tool retrieves coverage data of the SUT model based on the MC/DC (modified condition/decision coverage) criterion [1]. The MC/DC is a widely-used coverage

```
node choice () returns( x :int) =
  loop {
    | 3 : x = 42
    | 1 : x = 1
}
```

FIGURE 2. Lutin code, featuring a choice operator and the weights in boldfaced font, associated with the different choice possibilities.

criterion in the model-based testing of SCADE models and of safety-critical systems in general.

One of the goals of this case study was to decide whether a higher model coverage can be obtained with a realistic environment model than with a random one. In order to investigate this question, the coverage rate obtained by the realistic environment was compared to the one obtained by some test inputs generated at random. A detailed analysis of our results can be found in section 3.

2.3. **Oracle and oracle coverage.** The SUT expected properties were formalized as Lustre predicates. We used the functional requirements specification of the SUT in order to extract the oracle information.

In addition to automating the test decision, *oracle coverage* can also be measured with Lurette. It is defined as a set of Boolean conditions, while the *oracle coverage rate* is the rate of coverage conditions that have been true at least once during the run of the simulation [9].

In our case study, we used the oracle coverage rate as feedback in order to manually refine the environment and create even more realistic test scenarios. Furthermore, we compared the oracle coverage rate with the MTC coverage rate obtained with the Lutin environment. The results of the comparison are presented in section 3.

3. EXPERIMENTAL RESULTS

3.1. **SUT model.** Our case study focused on the roll control SCADE model. The model, together with the system's specification, was provided by Ansys [4], as part of the SCADE installation package. As shown in figure 3, the system's main functionality is to calculate a plane's roll rate, based on the joystick command and the left and right adverse yaw rates. The model has three real inputs: the joystick command (generated by the pilot), and the left and right adverse yaw of the plane's wings. Based on these inputs, the SUT calculates

[4]http://www.ansys.com/, downloaded on May 26, 2015

the plane's roll rate. In addition, if this rate falls outside of a given interval, warning alarms are generated by the model, represented as boolean outputs.

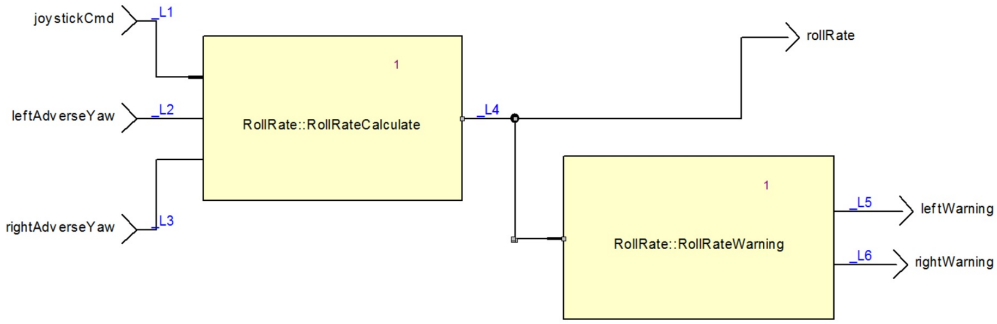


FIGURE 3. SCADE model of the roll control system. The system calculates the the plane roll rate (output **rollRate**) according to a joystick command (input **joystickCmd**) and the effect from the two plane wings (inputs **leftAdverseYaw** and **rightAdverseYaw**). In addition, the system computes the left and right warning alarms (outputs **leftWarning** and **rightWarning**) which are activated, respectively, if the plane roll rate is strictly less than -15.0° per second or strictly greater than 15.0° per second.

3.2. **Environment model.** For designing a realistic environment model in Lutin, the code from the Lutin manual[5] was used and adapted to the roll control SCADE model. The Lutin code simulates the change of the joystick command and left and right adverse yaws, respectively. Starting from an initial value chosen at random, the values start to increase or decrease by a random number, as shown in code snippets from figures 4 and 5, until they reach a certain minimum or maximum. Then, their values start to change in the opposite direction. The direction of the change is also chosen at random, using Lutin's choice operator. The number of iterations is influenced with Lutin's choice operator, as illustrated in the code snippet from figure 6. The parameters of the main node — shown in figure 7 were chosen empirically based on the oracle coverage. Figure 8 illustrates the inputs generated with Lutin after 50 steps.

3.3. **MTC analysis.** In order to analyse the effect of a realistic environment model on MTC, we generated two test suites. In the first case, we used the Lutin environment to generate input sequences of length 5, 10, 20, 50 and

[5]http://www-verimag.imag.fr/DIST-TOOLS/SYNCHRONE/lurette/doc/lutin-man.pdf, downloaded on April 24, 2015

```
let between(x, min, max : real) : bool =
        ((min < x) and (x < max))
```

FIGURE 4. Lutin combinator choosing a random number between **min** and **max**. Combinators are a kind of well-typed macros, which were introduced in the language to allow code reuse.

```
node up(init, delta:real) returns( x : real) =
   x = init fby loop
                { between(x, pre x, pre x + delta) }

node down(init, delta:real) returns( x : real) =
   x = init fby loop
                { between(x, pre x - delta, pre x) }
```

FIGURE 5. Lutin nodes increasing/decreasing the value of **x** with with a random number between 0 and **delta**. The **pre** operator accesses the value of **x** from the previous iteration.

```
node up_and_down(min, max, delta : real) returns
        (x : real) = between(x, min, max)
 fby
   loop {
        | run x := up(pre x, delta)
                        in loop { x < max }
        | run x := down(pre x, delta)
                        in loop { x > min }
       }
```

FIGURE 6. Lutin node with the choice operator |, choosing at random between the execution of the two nodes: up and down. Since there are no weights assigned to the two branches of the operator, the nodes are both chosen with probability 0.5. Once a branch is chosen, the execution of the node up/down is repeated until **x** reaches **max**/**min**.

100. In the second one, we randomly generated test sequences with the same length. The length of the input sequences was chosen empirically. The coverage obtained with the two methods was analysed with the MTC tool. The results are summarized in table 1.

A significant improvement in the coverage rate can be observed in the case of the Lutin environment compared to the random one, where the length of the input sequences was 5, 10 and 20. However, no improvements can be observed

```
node main(a:real; b:bool; c:bool) returns
        ( x:real; y:real; z:real; t:bool) =
  run x:= up_and_down(-30.0, 30.0, 10.0) in
  run y:= up_and_down(-30.0, 30.0, 10.0) in
  run z:= up_and_down(-20.0, 20.0, 10.0)
```

FIGURE 7. Main node of the Lutin code, describing the environment's behaviour. The weights were chosen empirically, based on the oracle coverage rate. Inputs: the plane's roll rate (**a**), the left (**b**) and right (**c**) warning signs; outputs: joystick command (**x**), left (**y**) and right (**z**) adverse yaws.

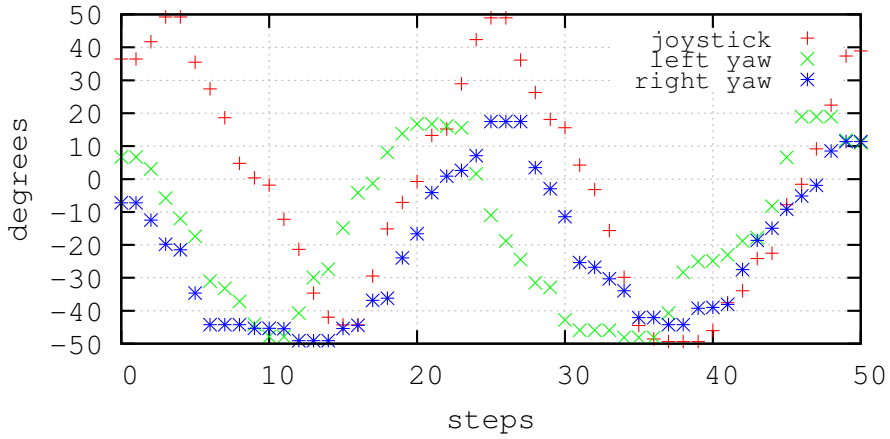


FIGURE 8. Test inputs generated with Lutin for the Roll Control SCADE system, through 50 iterations.

in the case of the Lutin environment for longer input sequences. This is due to the fact that the SUT chosen for the case study is a small SCADE model with few inputs and small inner state space. Thus, a high model coverage can be obtained even with random inputs, if the number of inputs is high enough.

3.4. **Oracle coverage analysis.** The oracle was implemented using Lustre observer nodes having access to both the input and output variables of the SUT. We focused on the requirements specifying the functioning of the roll control system's roll rate warning alarms. The complete description of the requirements can be seen in figure 9. We used two oracle nodes, as shown in code snippets from figure 10 and 11. The first oracle (figure 10) checks the correct functioning of the left warning alarm, while the second one (figure 11) supervises the right warning alarm.

Length of the input sequence	Random environment	Lutin environment
5	54.38%	68.42%
10	71.92%	75.43%
20	81.70%	84.21%
50	91.22%	91.22%
100	91.22%	91.22%

TABLE 1. MTC coverage rates obtained with the Lutin and the random environment.

	Roll Rate Warning Alarms
Short description	The roll rate warning alarms subsystem computes left and right warning alarms, which sound, respectively if the plane roll rate is strictly less than −15.0° per second or strictly greater than 15.0° per second.
Inputs	- Plane roll rate
Outputs	- Left warning alarm - Right warning alarm

FIGURE 9. Part of the roll control system's specification describing the functionality of the roll rate warning alarms. In our case study, we focused on this requirement when designing the oracle and oracle coverage. Source: official SCADE tutorial.

The oracle nodes fulfill two roles. On one hand, they check if the SUT generated outputs match the expected outputs for some given inputs according to the specification. On the other hand, the oracle nodes also contain coverage criteria. As explained in section 2.1, the coverage criteria are expressed as Lustre conditions. The values of these conditions are stored in boolean variables, which must also be added to the node's output list. Lurette calculates the oracle coverage rate based on the value of these boolean variables. A coverage criterion is considered covered if it has been true at least once during the test execution.

We used the oracle coverage rates computed by Lurette to manually refine the Lutin environment and generate more realistic test cases. In addition, we found that there is a correlation between the oracle coverage and the MTC coverage. Table 2 shows how the MTC coverage rates increased together with the oracle coverage rate.

```
node too_low(b,c,t:bool; a,x,y,z:real)
        returns (ok, c1, c2, c3:bool);
let
  ok = true -> a>=-15.0 or b;
  c1 = a<-15.0;
  c2 = a=-15.0;
  c3 = a>-15.0;

tel
```

FIGURE 10. Oracle in Lustre, checking requirement 9. If the roll rate of the plane is **smaller than -15° per second**, the left warning alarm should turn on. Boolean inputs **b** and **c** represent the state of left and right warning alarms, respectively. Real input **a** holds the value of the plane roll rate, while **x** the joystick command, **y** the left adverse yaw, and **z** the right adverse yaw. The oracle coverage is expressed as the constraints **c1**, **c2** and **c3**. It covers the cases where the roll rate of the plane lies around the lowest allowed limit (-15° per second).

```
node too_high(b,c,t:bool; a,x,y,z:real)
        returns (ok, c4, c5, c6:bool);
let
  ok = true -> a<=15.0 or c;
  c4 = a<15.0;
  c5 = a=15.0;
  c6 = a>15.0;

tel
```

FIGURE 11. Oracle in Lustre, checking requirement 9. If the roll rate of the plane is **greater than 15° per second**, the left warning alarm should turn on. Boolean inputs **b** and **c** represent the state of left and right warning alarms, respectively. Real input **a** holds the value of the plane roll rate, while **x** the joystick command, **y** the left adverse yaw, and **z** the right adverse yaw. The oracle coverage is expressed as the constraints **c4**, **c5** and **c6**. It covers the cases where the roll rate of the plane lies around the greatest allowed limit (15° per second).

4. Related Work

The research domain of model-based testing has been explored in a number of references. We mention a few of the related articles which emphasize the practical applicability of environment-model based testing methods to real-life problems. We focus especially on articles in which Lutin and Lurette were

Oracle coverage	MTC coverage
33%	**73.68%**
50%	**75.43%**
66%	**77.19%**

TABLE 2. Oracle and MTC coverage rates obtained with the Lutin environment. The length on the input sequence was 20.

used to perform the testing, but we also briefly review the work related to other environment-model based testing tools.

In [9] a case study is presented where Lutin and Lurette are used for checking the correctness of reactive systems developed incrementally. These tools are also used for elaborating and refining formal, consistent and accurate functional requirements. The authors use the oracle coverage rate for early validation of both the system model and the formal requirements, and also to improve the test scenarios. The case study presented is based on a collaboration with industrial developers of nuclear power plant control systems. One of the tested systems was developed using SCADE.

Two further case studies are presented in [8]. One of them presents a dynamic system which simulates the behaviour of the temperature and pressure of a fluid in a pipe, while the other one Lutin is used to automate the execution of timed test plans. The authors concluded that Lutin and Lustre allow the design of test plans that are more robust to software (or specification) evolutions.

The usage of Lurette for automatic generation of realistic input sequences is demonstrated also in [11]. Three industrial case studies in testing reactive embedded programs are presented here. The examples were extracted from an application developed in SCADE. The first example is a SCADE node that converts resistance to temperature, the second one is a component that computes the position of a propulsion nozzle according to the values of two sensors that measure electric tension, while the last one is a typical example of a fault-tolerant heat controller. In the presented examples, the Lucky language was used to model the environment. Working with similar modelling techniques, Lucky was basically the previous version of Lutin.

In all of the above mentioned articles — just like in our case study — oracle coverage is analysed and used to refine the environment model. However, unlike in our work, the authors of these articles don't investigate the structural coverage of the SUT model.

Beside Lutin and Lurette, there are several other languages and tools available for performing environment-model based testing on reactive systems. Bousquet et al. [2, 4] present a specification-based language called Lutess,

while Marre et al. [12, 13] describe Gatel, a test generation tool for Lustre programs. However, we could not find any case studies discussing how these tools can improve the structural coverage of the SUT model.

5. Conclusion and Future Work

We presented a case study where environment-model based testing was performed on a reactive system in form of a SCADE model. The system modelled the behaviour of a plane's roll control. To create an environment model, we used Lutin, a language that was originally designed to program stochastic reactive systems. Through the use of Lutin, we were able to design realistic environments. For automating the test decision, as well as to define the oracle coverage criteria, we used several Lustre nodes.

We analysed the SUT model coverage using the SCADE Suite MTC tool. In order to investigate whether the Lutin environment has improved the model coverage, we compared the MTC coverage rate achieved by the Lutin environment with the coverage rate obtained by a random environment. Beside the MTC coverage, we also computed the oracle coverage rate based on the Lustre oracle coverage criteria. We used this coverage rate as feedback in the process of refining the environment model and obtaining realistic test cases with Lutin.

Experimental results with the MTC tool showed that, in the case of the realistic Lutin environment, the model's coverage rate increased significantly for short input sequences of length 5, 10, and 20, compared to the coverage rate obtained with the completely random environment. However, no improvements could be observed for longer sequences with length 50 or 100. If the input sequence was long enough, a coverage rate close to 100% could be achieved. Further research however should test the method on a more complex SCADE model in the railway automation domain, coming from our industrial partner, Siemens. Here we expect more spectacular results.

Furthermore, to fully benefit of the possibilities offered by Lutin's syntax, we plan to add weights to the choice operators from our Lutin code. This way, we will have more control over the non-determinism of the environment. In addition, we plan to experiment with different learning techniques in order to improve the Lutin parameters automatically, and increase the model coverage, with input sequences as short as possible. One possibility would be the implementation of the Differential Evolution [6] technique in order to automatically fine-tune the environment model. We will use the parameters of Lutin's non-deterministic operators to build up the population of the evolutionary algorithm, and both the MTC and oracle coverage rates to measure the fitness of a population.

6. Acknowledgement

This material is based upon work supported by the Siemens international Railway Automation Graduate School (iRAGS) and the SCADE Academic Program. Thanks for Erwan Jahier from Verimag Research Center[6], for providing technical support with Lutin.

References

[1] Paul Ammann and Jeff Offutt. *Introduction to Software Testing*. Cambridge University Press, New York, NY, USA, 1 edition, 2008.

[2] L. du Bousquet and N. Zuanon. An overview of Lutess: A specification-based tool for testing synchronous software. In *Proceedings of the 14th IEEE International Conference on Automated Software Engineering*, ASE '99, pages 208–215, Washington, DC, USA, 1999. IEEE Computer Society.

[3] P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice. Lustre: A declarative language for real-time programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '87, pages 178–188, New York, NY, USA, 1987. ACM.

[4] L. du Bousquet, F. Ouabdesselam, J.-L. Richier, and N. Zuanon. Lutess: A specification-driven testing environment for synchronous software. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 267–276, New York, NY, USA, 1999. ACM.

[5] Francois Xavier Dormoy. Scade 6 a model based solution for safety critical software development. ERTS 2008, 2013.

[6] S. Das and P.N. Suganthan. Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on*, 15(1):4–31, Feb 2011.

[7] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous data flow programming language lustre. *Proceedings of the IEEE*, 79(9):1305–1320, Sep 1991.

[8] Erwan Jahier, Simplice Djoko-Djoko, Chaouki Maiza, and Eric Lafont. Environment-model based testing of control systems: Case studies. In Erika Ábrahám and Klaus Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 636–650. Springer Berlin Heidelberg, 2014.

[9] Erwan Jahier, Nicolas Halbwachs, and Pascal Raymond. Engineering functional requirements of reactive systems using synchronous languages. In *International Symposium on Industrial Embedded Systems, 2013. SIES'13.*, Porto, Portugal, 06 2013.

[10] Erwan Jahier and Pascal Raymond. Generating random values using binary decision diagrams and convex polyhedra. In *Trends in Constraint Programming*, pages 349–356. ISTE, London, UK, may 2007. http://www.iste.co.uk/index.php?isbn=9781905209972.

[11] Erwan Jahier, Pascal Raymond, and Philippe Baufreton. Case studies with lurette v2. *Int. J. Softw. Tools Technol. Transf.*, 8(6):517–530, October 2006.

[6] http://www-verimag.imag.fr/?lang=en, downloaded on May 26, 2015

[12] Bruno Marre and Agnes Arnould. Test sequences generation from lustre descriptions: Gatel. In *Proceedings of the 15th IEEE International Conference on Automated Software Engineering*, ASE '00, pages 229–, Washington, DC, USA, 2000. IEEE Computer Society.

[13] Bruno Marre and Benjamin Blanc. Test selection strategies for lustre descriptions in gatel. *Electronic Notes in Theoretical Computer Science*, 111:93–111, Jan 2005.

[14] P. Raymond, X. Nicollin, N. Halbwachs, and D. Weber. Automatic testing of reactive systems. In *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, pages 200–209, Dec 1998.

[15] Pascal Raymond, Yvan Roux, and Erwan Jahier. Lutin: A language for specifying and executing reactive scenarios. *EURASIP J. Emb. Sys.*, 2008, 2008.

[16] Mark Utting and Bruno Legeard. *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

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

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