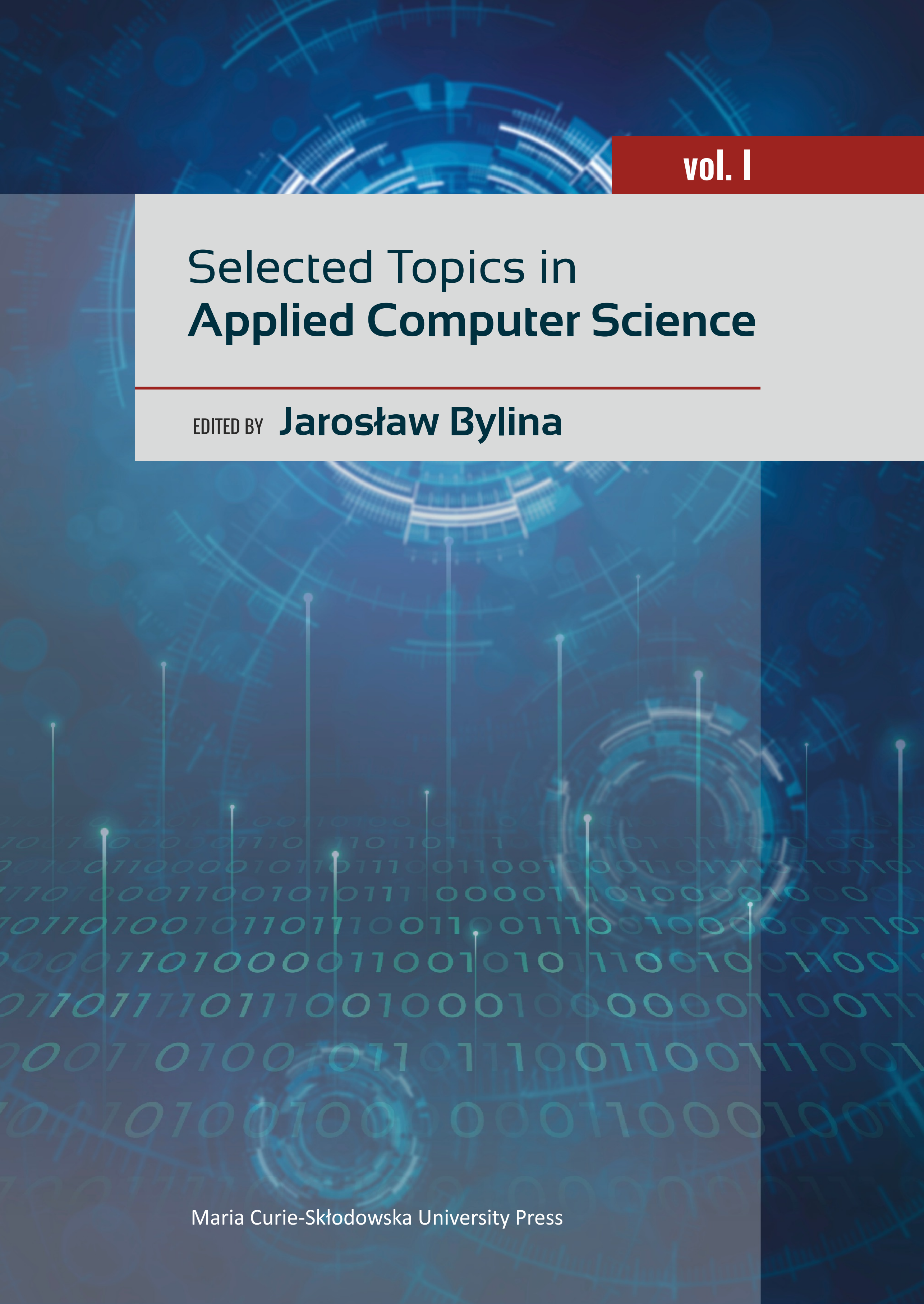


vol. I

Selected Topics in Applied Computer Science

EDITED BY **Jarosław Bylina**

Maria Curie-Skłodowska University Press

Selected Topics in
Applied Computer Science

UMCS

WYDAWNICTWO

vol. I

Selected Topics in Applied Computer Science

EDITED BY **Jarosław Bylina**

Maria Curie-Skłodowska University Press
Lublin 2021

Reviewers
Marcin Blachnik
Krzysztof Grochla
Dariusz Mikołajewski
Mateusz Nowak
Piotr Oleszczuk
Marek Pałkowski

Cover and front page design
Krzysztof Trojnar

Typesetting
Jarosław Bylina
Anna Nosalewicz

Cover illustration: pixabay.com, freepik.com

ISBN 978-83-227-9530-9

Maria Curie-Skłodowska University Press
ul. Idziego Radziszewskiego 11, 20-031 Lublin, Poland
tel. +48 81 537 53 04
www.wydawnictwo.umcs.eu
e-mail: sekretariat@wydawnictwo.umcs.lublin.pl

Sales Department
tel./fax +48 81 537 53 02
Online bookstore: www.wydawnictwo.umcs.eu
e-mail: wydawnictwo@umcs.eu

Contents

Preface 7

A Brief Review on Supervised Machine Learning 9

Automatic Syllable Repetition Detection Methods in Continuous Speech 43

Exploring Recent Advancements of Transformer Based Architectures in Computer Vision 59

Automating the Comparison of Areas and Data of Administrative Units From Different Periods on the Example of Poland (1937 and 2019) 77

Some Computational Aspects of Graph-Based Cryptography 95

DNA Based Cryptographic Key Storage System With a Simple and Automated Method of Primers Selection 111

Comparative Analysis of Selected Anthropomorphic Grippers Constructions 129

Data Mining Procedures in the Oil Production Prediction for Gas Lifted Wells 141

Spatial Databases and Their Use in Spatial Web Applications Based on the Exemplary Internet Service for Same Chosen Objects in the Old City in Zamość 155

Nested Loop Transformations on Multi- and Many-Core Computers With Shared Memory 167

Functioning of Transnational Civil Society Organisations (TCSOs) in Cyberspace 187

Applying Ethics to Autonomous Agents 199

Assessment of Attractiveness and Trust in Relation to Personality Traits — Literature Review and Research Proposal 223

Liquid State Machines for Real-Time Neural Simulations 233

Author index 247

Preface

The development of computer science has been growing more and more dynamic since several decades. Nowadays, this progress is marked by a number of trends. Just to name a few — artificial intelligence (most prevalent in its machine learning variety), cybersecurity, bioinformatics. However, the problems of general purpose algorithms and the creation of advanced information systems are still being investigated. All these issues create new opportunities, but generate many research questions. This book presents the latest research conducted at Marie Curie-Skłodowska University in Lublin — or in collaboration with its staff — in the broadly understood field of computer science.

Subsequent chapters deal with:

- machine learning and automation of detection in various data sets;
  - *A Brief Review on Supervised Machine Learning*
  - *Automatic Syllable Repetition Detection Methods in Continuous Speech*
  - *Exploring Recent Advancements of Transformer Based Architectures in Computer Vision*
  - *Automating the Comparison of Areas and Data of Administrative Units From Different Periods on the Example of Poland (1937 and 2019)*
- modern cryptography;
  - *Some Computational Aspects of Graph-Based Cryptography*
  - *DNA Based Cryptographic Key Storage System With a Simple and Automated Method of Primers Selection*
- industrial computer science issues;
  - *Comparative Analysis of Selected Anthropomorphic Grippers Constructions*
  - *Data Mining Procedures in the Oil Production Prediction for Gas Lifted Wells*
- more traditional approaches to data processing — like (geographical) data base systems and parallelization;
  - *Spatial Databases and Their Use in Spatial Web Applications Based on the Exemplary Internet Service for Same Chosen Objects in the Old City in Zamość*

  - *Nested Loop Transformations on Multi- and Many-Core Computers With Shared Memory*

- social quations related to information technology;
  - *Functioning of Transnational Civil Society Organisations (TCSOs) in Cyberspace*
- ethical/philosophical issues of artificial inteligence;
  - *Applying Ethics to Autonomous Agents*
- some research on the functioning of the human brain and nervous system and its simulation possibility.
  - *Assessment of Attractiveness and Trust in Relation to Personality Traits — Literature Review and Research Proposal*
  - *Liquid State Machines for Real-Time Neural Simulations*

The book is addressed to all those who want to become familiar with the areas of research conducted by employees of Maria Curie-Skłodowska University (in particular, the Institute of Computer Science at the Faculty of Mathematics, Physics and Computer Science). Thanks to concise and clear descriptions of many different issues related to the broadly understood computer science, it will be especially useful for IT/CS students and scientists as well as research staff of enterprises.

The book was created becasue of the huge commitment of its co-authors, reviewers, editors. Many employees of the Institute of Computer Science at the University worked on its final shape. I would like to express my deepest thanks to all of them, especially to Prof. Przemysław Stpiczyński, who encouraged all of us to prepare a contribution to this volume.

Jarosław Bylina
`jaroslaw.bylina@umcs.pl`

A Brief Review on Supervised Machine Learning

Monika Piekarz*

1 Introduction

What is Machine Learning? An area of artificial intelligence dedicated to algorithms that improve automatically through experience, i.e. exposure to data. Simply put, the machine perceives the pattern and tries to imitate it in some way, directly or indirectly. This comes down to two main types of machine learning: supervised and unsupervised. Supervised machine learning is a direct imitation of a pattern that exists between two data sets, the input set we want to convert to the output set. For example, one set containing data from a weather sensor, which we will use to predict the information from the second set, which is the probability of rain or snow. Unsupervised learning also transforms one data set into another, but the data set to which we transform the input is not known to us beforehand. So there is no correct answer here for which we are trying to get a duplicate model. Unsupervised learning is more about finding a pattern in the data and showing it. An example would be a cluster analysis of a data set. Cluster analysis transforms the sequence of observations into a sequence of labels groups. If 5 groups are discovered, the labels will typically range from 1 to 5. Other uses for unsupervised learning are: detecting anomalies, i.e. cases that are significantly different from the rest, assuming we do not know the characteristics that would make the difference [19], or autocode on detecting abstract features by deep neural networks, e.g. shapes visible in coded images or generating new images that are some combination of training images [5]. In other words, supervised learning requires a data set with a set of valid solutions, the training algorithm analyzes both sets to associate one set with the other and applies it later to unknown data to predict a correct answer. In unsupervised learning we also have a data set that is analyzed, but here there are no correct answers that we would like to predict. It is just about discovering the relationship between the analyzed data. Next, we will mainly talk about supervised learning.

Supervised and unsupervised machine learning is just one of the possible divisions that we can observe in machine learning. When we look at the development of machine learning research from its first seeds, we see that there are very different forms of the process involved. It is even difficult to introduce a full, unambiguous

*Corresponding author — `monika.piekarz@mail.umcs.pl` and complete division. The types of learning systems can be classified according to many criteria, resulting in different divisions. The most basic of them are: methods of representing knowledge and skills, the way knowledge or skills are used, the source and form of training information, the mechanism for acquiring and improving knowledge or skills.

Methods of representing knowledge are generally related to the technique we will use to solve the problem. We can distinguish here: decision trees, decision rules, predicate logic clauses, probability distributions, finite automata transition functions, formal grammar rules or regression function parameters.

If we are talking about the method of using knowledge, i.e. about the task facing a learning machine, then regression is one of the most typical tasks, where the hypothesis is a function and here we can distinguish classification (the result obtained is discrete information) and regression (the result obtained takes the form of a continuous) problems, which will be discussed in more detail in the next chapter. A hypothesis can also take the form of a rule that divides examples, then we talk about the so-called partitioning models. The machine learning algorithms used here are: decision trees, decision trees forest, enhanced decision trees. These algorithms can also be used for classification and regression.

The last of the mentioned criteria: knowledge acquisition mechanism — here we can distinguish: linear and non-linear regression, artificial neural networks, decision tree learning, Bayesian learning, predicate calculus, finite automatons and others. We'll take a closer look at the first to of them by discussing regression and classification problems.

The presented proposals of the classification of learning systems are to some extent orthogonal, i.e. they can be used independently, although sometimes the decision regarding one of the criteria limits the decision-making possibilities regarding other criteria. Each of these criteria breaks the learning models into groups that have some overlap. Making a systematic classification of the field of machine learning seems unlikely at the moment, as the field did not develop systematically.

2 The idea of supervised machine learning

We'll start by establishing the terminology. An **example** is an observed object of interest. A **feature** is an input variable that is used to characterize the example. Let x_i denote the input variables (vector of features) and y_i denote the output (target) variable. The pair (x_i, y_i) forms a single example. Let X be the set of values of the input variables, and Y be the set of values of the output variable. A **hypothesis** is a function h that maps the set of features x_i into some target y_i.

In supervised machine learning we have a set of examples, called training set, described by certain features x_i and for which we know the result of interest y_i described as pairs (x_i, y_i) (we will skip indexes where they are not necessary for proper interpretation of the record). For example, (x_i, y_i) represent a set of weather conditions described by the atmospheric pressure and air temperature x_i and the corresponding result describing the probability of rain y_i. The machine learning algorithm analyzes this data and its goal is to converge to the best as possible hypothesis h which returns the values of the output variable based on the values of the input variables:

$$h : X \to Y$$

Possible hypotheses create a hypotheses space (bias) and the task of the machine learning algorithm is to find in this space the hypothesis closest to the target function $f : X \to Y$ which is mostly unknown to us because we only have a sample of data included in the training set. Therefore, we cannot use f to judge how well our learning algorithm has performed. To evaluate the model we need to use a test data set. Training and test data sets must be separate, i.e. the training data cannot be used for model evaluation even once, and the test data cannot be used for training even once. From the hypothesis it is expected to be capable of dealing with examples which were never seen before.

The algorithm analyzes the training data to detect the relationship between the input and output variables. These data are analyzed many times, verifying various hypotheses h. The comparison of the hypotheses found by these algorithms requires the definition of a loss (error) function l, which determines the distance between the prediction results $\bar{y_i}$ and the actual values of the target variable y_i. Using the loss function on the training set, we can calculate the so-called training error $R_{emp}(h) = \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i, h)$. This error is called empirical error. It is the mean prediction error for training data set, n is the number of examples in training set. We can always limit the training error to zero. It is enough for the algorithm to remember all training data, of which there is always a finite amount. Our goal is to minimize the (real) test error computed based on data which the algorithm did not see during the learning process, i.e. the average prediction error for the test data set. This error is called a generalization error.

The errors can be an underestimation or overestimation of the resulting variable. Both types of errors should be taken into account, we can do it by counting the cases when the prediction was different from the target variable or by counting the sum of the absolute values of errors for individual examples or the sum of their squares. Here are the three most common loss functions used to evaluate prediction errors:

$$l_{0-1}(x_i, y-i, h) = \begin{cases} 1, & \text{if } h(x_i) \neq y_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$l_{abs}(x_i, y-i, h) = |h(x_i) - y_i| \tag{2}$$

$$l_{sqr}(x_i, y-i, h) = (h(x_i) - y_i)^2 \tag{3}$$

In [3] we can read that the goal of machine learning is for a given lost function l and a sample of D selected from the P population with an unknown distribution to find the h function with the smallest expected error ε for the P population with respect to the lost function l.

The presented idea is most easily reflected by the regression model, whose task is discover the relationship between the input variables and the output variable expressed by the hypothesis h. The simplest hypothesis of a regression model is based on the assumption of a linear relationship between the input variables and the output variable. So, assuming we have one output variable, we are looking for a hypothesis of the following form:

$$y_i = h(x_i) = b_0 + b_1 x_i$$

where the intercept b_0 is the value of the variable y_i for $x_i = 0$ and the regression coefficient (slope) b_1 determines by how much the value of the variable y_i changes with the change of the variable x_i. Such a model is called a linear regression model.

In practice, the value of the output variable is computed from multiple input variables, so our hypothesis will take the following form:

$$y_i = h(x_i) = b_0 + b_1 x_{i1} + b_2 x_{i2} + \ldots + b_k x_{ik} \tag{4}$$

In 1908, Karl Pearson called such a linear regression model multiple linear regression.

How do we proceed to find the right hypothesis h for our data? We can choose a certain hypothesis h (i.e. determine some b_0 and b_1 if we decide to apply linear regression assuming that we have one feature x describing the resulting variable y in our model), and then check how wrong we are and then correct our choice b_0 and b_1 so that the next hypothesis h' adopted by us fits better with data, that is, less wrong. To check how wrong we are we need a lost function l. One of the most frequently used lost is l_{sqr} function, so to compute empirical error and evaluate regression models is often used the mean square error (MSE). For one feature of x and n elements in the data set, the mean square error is as follows:

$$R_{emp}(b) = \frac{1}{n} \sum_{i=1}^{n} (b_0 + b_1 x_i - y_i)^2 \tag{5}$$

In the figure 1, we have a graphical illustration of the MSE for predictions range from -50 to 50, the target value is 10.

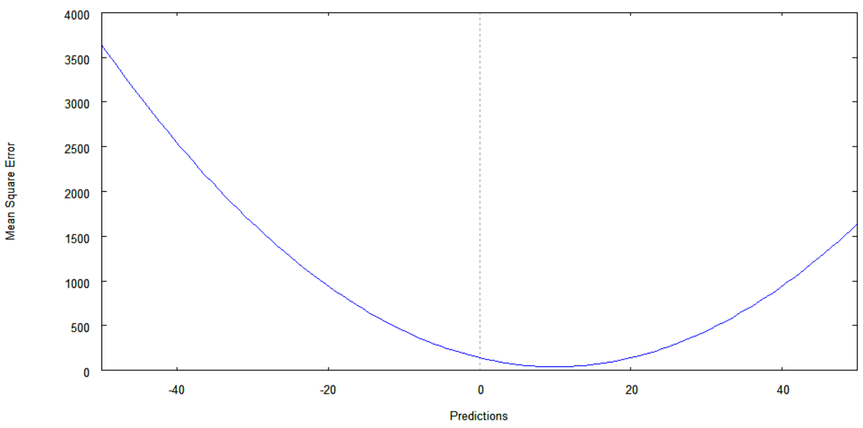


Figure 1: When using the lost function l_{sqr}, one big mistake costs the model much more than many small mistakes. Consider 10 examples. If the error of one example was 50 and the error for the remaining 9 cases was 0, then the mean square error would be 250. If the error for all 10 examples was 2, then the mean square error would be 4

This error increases with the square of the difference between the predicted and target values of the output variable. This means that the model will learn faster from large errors. An additional benefit of the MSE is that it has only one minimum, the global minimum. The lack of local minima greatly simplifies finding the global minimum. Overall, we want to minimize the error for our data set. The derivative is a measure of the rate of change of a function's value in relation to the smallest changes in its argument. The derivative represents the slope of the curve at a given point. This means that we can find the value of the MSE lower than the current value by moving in the opposite direction to the sign of the derivative. The derivative of the MSE should be calculated for all parameters of the model. In this way we will obtain a gradient of the lost function, denote it with the symbol ∇.

$$\nabla R_{emp}(b) = \left(\frac{\partial R_{emp}(b)}{\partial b_0}, \ldots, \frac{\partial R_{emp}(b)}{\partial b_k} \right) \tag{6}$$

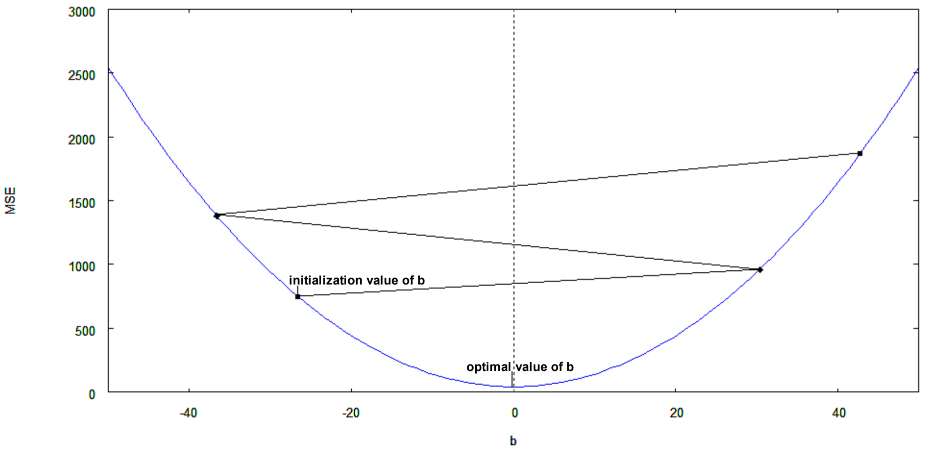


Figure 2: High learning coefficient

How is the learning process going? Initially, the parameter values are random. For these values, the empirical error and its gradient are calculated. Then the parameter values are changed towards the fastest gradient descent (eq. 7). The learning coefficient λ (another model hyperparameter) controls how many parameter values are changed at one time. A sufficiently low learning coefficient guarantees finding the minimum cost function, although it extends the learning process. A higher learning coefficient shortens learning but increases the risk of not finding the minimum lost function by overshooting the minimum (fig. 2). The method of optimizing the learning process has a direct impact on the speed of learning and indirectly on its result [2]. Training data can include millions of examples described by hundreds of features. Machine learning usually requires the repetition of several hundred to tens of thousands of cycles, i.e. multiple counting of the empirical error and its gradient for all examples from a data set, which may require large computing power and large memory, making training completion possible only for high learning coefficients.

The basic version of the fastest gradient descent method, Bath Gradient Descent, is to update the parameter values based on the gradient of the empirical error calculated for all examples from the data set, followed by a one-time update of all parameters:

$$b' = b - \lambda \nabla R_{emp}(b) \tag{7}$$

These two operations are called the cycle or the epoch.

This method guarantees finding the minimum of the MSE if the learning rate is small enough for problems characterized by a linear relationship between the values of the input features and the output variable. However, we must remember that the problem we are interested in does not always have to be described by such a simple relationship between the values of the input features and the output variable, more on that on page 23.

The stochastic gradient descent (STD), calculates a gradient based on one example (x_i, y_i) randomly selected from a data set in a given cycle.

$$b' = b - \lambda \nabla (b_0 + b_1(x_i) - y_i)^2 \tag{8}$$

The gradient of the lost function calculated in this way is not very stable, its values for subsequent cycles may be completely different. In general, it turns out that introducing an element of randomness prevents learning interruption in the local minimum of the function. This method significantly reduces the learning time by updating the method parameters just on the basis of one example. Often we deal with a situation where the direction of parameter changes can be correctly determined on the basis of the gradient determined for one example. Then it takes us unnecessarily time to evaluate all the other examples in the same cycle. Another modification is the fastest gradient descent method for a mini group, Mini-Bath Gradient Descent. It combines the advantages of both of the above methods, determining the value of the gradient of the lost function from a small subset of examples from the data set. The size of this subset controls the hyperparameter — batch size g. The higher the value of g, the slower the learning process.

$$b' = b - \lambda \nabla \frac{1}{g} \Sigma_{i=1}^{g} (b_0 + b_1(x_i) - y_i)^2 \tag{9}$$

It is widely believed that this modifications give the best learning outcomes [6].

3 Data preparation

When we start working with the self-learning algorithm, there are four tasks to be done:

1. Preparation of data that is undoubtedly the most important in machine learning.

2. Selecting the model/algorithm.

3. Carrying out the learning process.
4. Model evaluation, which will tell us if we have achieved success or if the previous steps should be repeated.

Data preparation consists of two steps: data selection and data transformation.

3.1 Data selection

Machine learning algorithms extract information hidden in data in several ways. The methods used by these algorithms have various limitations that can most effectively be overcome by using more and better quality data for training [4]. The data used must be in the form of a table and arranged so that the rows represent single examples and the columns of the values of individual variable [24].

After organizing the data, analyze the individual variables. We divide the variables into *numerical* (quantitative) and *categorical* (qualitative). Numeric variables are variables of such types as: bool, int, float, data, i.e. variables with only numerical values. Due to the distribution, numerical variables are divided into *discrete* (finite set of values) and *continuous* (infinite set of values). Numeric variables can be compared with each other, the distance between them can be determined, arithmetic operations can be performed. Categorical variables are variables of other types, mainly of the text type. The power of the collection of their values is always limited. *Ordinal categorical* variables are those whose values can be compared with each other, the rest are *regular categorical* variables. We cannot perform arithmetic operations on categorical variables or determine the distances between them. In order to evaluate the type of a variable, one needs to understand its meaning, e.g. the class variable taking the values: 1, 2, 3 is a discrete numeric variable. If one takes into account that it describes the class of the purchased train compartment, then it should be considered an ordinal categorical variable.

Understanding the data starts with getting to know the individual variables. The easiest way to do that is to know the descriptive statistics. Some are the same for numerical and categorical variables, some have the same meaning but they are counted differently, e.g. range, and still others can be determined only for numerical variables, e.g. mean.

The distribution of the values of numerical variables is described by ten main statistics:

- place of the greatest concentration — measures of central tendency: median, arithmetic mean and mode;
- value differentiation — measures of dispersion: range, interquartile range, variance, standard deviation and coefficient of variation;
- comparison of the shape of the distribution of the variable with the normal distribution — measures of symmetry of the distribution: skewness and kurtosis.

Categorical variables are assessed using frequency tables and histograms.

By measuring the entropy of a variable, it is possible to evaluate its usefulness in the case of participatory models. Entropy is a measure of the information contained in [21].

Missing data is also a big problem because it has to be completed either by providing information about it from other sources or by supplementing it with substitute values, e.g. 0, NULL values or an empty string depending on the variable. Sometimes a variable that has a lot of data missing needs to be removed because it becomes useless.

It may also be important to present the data in charts. Although statistics provide us with knowledge about the data, they do not contain complete information and may be misleading. This problem is illustrated by the example given by Francis Anscombe [1]. He considered four data sets with almost identical statistics (the arithmetic mean of variable X equal to 9 and variable Y equal to 7.5, and the standard deviation of variable X equal to 11 and variable Y equal to 4.125). Each of these four sets has the same linear regression model: $y = 3 + 0.5x$, a correlation coefficient of 0.816 and a coefficient of determination R^2 of 0.666. On the other hand, the scatter plots of these sets show how much they differ from each other (fig. 3).

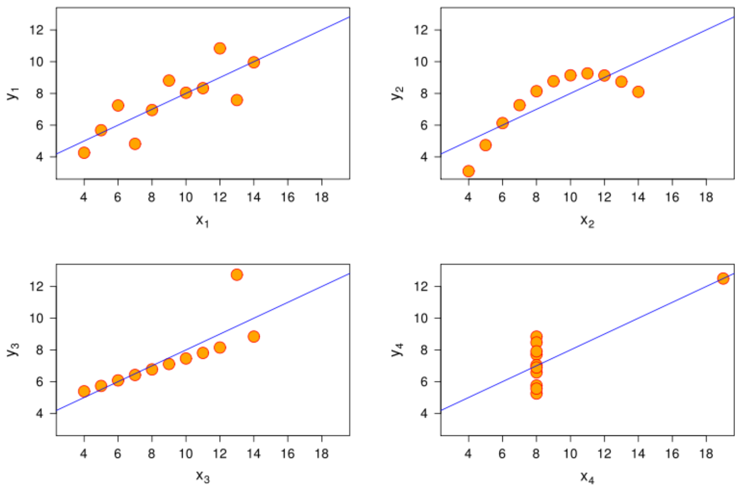


Figure 3: The points represent the data, the straight line represents the regression model. Source: https://commons.wikimedia.org/wiki/File:Anscombe.svg

Examining the data in the graphs is the more important the more the distribution of the variable deviates from the normal distribution. Especially if it has a non-unimodal distribution. For example, if it has a bimodal distribution, and the most common values are extreme, e.g. 5 and 100, the mean value and median will not matter, they will not represent a typical case, as there are two typical cases in such a set.

The collected information about the data will allow us to identify and solve problems with their quality. First of all, remove the variables which we are sure will not be useful, that is, variable with unique values as identifiers, invoice numbers, PECEL, NIP. Failure to remove them may result in an excessive fit of the model to the training data, who will learn to recognize the value of the result variable based on unique information that will never be repeated in the test data set, instead of looking for a general relationship between the input variables and the result variable that will also appear in the test data. Remove as well the variables with only one value, variables with the majority of missing values.

The greatest benefit of statistical data analysis is the ability to assess its representativeness. Knowing the central tendency and distribution of values, we can judge whether they are as expected. This assessment is best done in a discussion with an expert in the field.

It is important that our sample of data accurately represents the entire population. In any case, the model will learn fictitious, sample-based but not real-world relationships between the data. Contrary to popular belief, simply increasing the sample does not always prevent us from doing so. There is a rule in the statistics that if the examples were collected in specific conditions, the sample should not exceed 10% of the population. This is because more examples will more accurately reflect local characteristics. The ideal situation would be to select randomly from the entire population. This means that each case should have the same, non-zero chance of being sampled.

However, if the sample is representative, more examples lead to better learning outcomes due to convergence to actual distribution. As the sample increases, the distribution of the values of the variables changes until the distribution is identical to the distribution of the entire population.

Another very important element that should be noted is the correlation between the variables. This is so important because machine learning is about looking for relationships between the input variables and the result variable.

Two variables are correlated with each other if knowing the value of one variable helps in determining the value of the other variable. The stronger the correlation, the better we can predict the value of one variable from the value of the other. The output variable is also called dependent. This means that the output variable should be correlated with the input variables.

On the other hand feature selection, by removing correlated input variables is an effective way to simplify models, improve their quality and reduce learning time [13]. Removing unhelpful variables is important because with each additional variable the number of possible combinations of their values increases significantly. The number of possible combinations of a set of n variables is equal to the sum of the number of combinations of 2-element, 3-element, ..., n-element. We do not consider single-element combinations because we are not interested in the correlation of the variable with itself. And so, 4 variables give 11 ways of juxtaposing them, 8 — 247 ways, and 16 — 55371 ways.

In addition, there is also a problem with the number of variables called the curse of dimensionality. In machine learning we have a finite number of data samples in a multidimensional feature space with each feature having a range of possible values. We should provide several samples in the training set with each combination of the value of the input variable, so the more features we take into account, the more training data we need.

Suppose we have 10 examples and each variable can take values from 0 to 10. If we have a one-dimensional variable x, the examples can be illustrated as line points in the range $[0, 10]$, and their position will correspond to the value of x. Suppose we have the second variable y. Now we need to put our sample points on the plane in the places defined by the (x, y) value pairs. So the size of our space has increased from 10 to 100 ($10^2 = 100$). This will increase the distance between the points. After adding another variable, the positions of the points will be determined by the three (x, y, z) coordinates and the size of the space will increase to 1000, which will greatly increase the distance between the points, of which there are still 10.

There is another reason why increasing the dimensions can make easy-to-solve problems in spaces with a small number of dimensions more difficult. For example, let's take ten equally spaced points. In one-dimensional space, each of these points will have two equally distant closest neighbors. In two-dimensional space, the number of nearest neighbors of each point increases to 4, in three-dimensional space to 6, and so on. By increasing the number of dimensions we increase the number of coordinates of points and the number of nearest neighbors, if these points are equally spaced from each other, their number is twice the number of coordinates. Thus, in a 20-dimensional space, a point will already have 40 neighbors. This is a problem especially for machine learning algorithms based on comparing the distance between points, e.g. k nearest neighbors [25] or the k-means clustering algorithm [10, 18].

Given a fixed number of training samples, the predictive power of the model first increases with the number of features (dimensions) used and then decreases [12], which is known as the Hughes effect or the peak effect [11].

3.2 Data transformation

The effectiveness of the various methods of optimizing the learning process strongly depends on the training data. For example, the steepest gradient training process is most effective for data that has an average of 0; decision tree splitting methods favor categorical variables with a greater number of states; the loss function l_{sqr} is very sensitive to outliers. Data transformation aims to adapt them to knowledge representation algorithms, learning optimization methods and loss functions.

We distinguish five areas of data transformation.

3.2.1 Coding

The easiest way to process is numerical variables, we have many mathematical and statistical methods to operate on them. Hence, sometimes it is worth coding categorical variables as numeric. Some algorithms, especially those used in regression methods, even require only numerical variables.

- Most often categorical variables are coded with the one-hot encoding method. It consists in replacing a categorical variable with as many binary variables (so-called dummy variables) as many as many states per a categorical variable. For a given example, only one of the variables created in this way takes the value 1 — the one that represents the actual state of the coded variable, the value of the remaining ones is set to 0. Coding with this method does not require special knowledge, it can be performed automatically and does not change the state distribution of the original variable. However, its disadvantage is that it increases the number of input variables, especially in the case of categorical variables assuming many states. Another disadvantage is that the less frequent states will be represented by variables taking the value 1 very rarely, so that the usefulness of these variables for the model will be negligible. Therefore, before using this method, the variables should be generalized.

- We can also simply replace categorical variables with numbers, this way the new variable will take as many values as the states have the original variable but they will be numbers, this method is called state encoding. When using this method, we need to know that we are introducing to the model additional new false information about the ordering of states that may worsen its quality. State encoding can also be applied to ordinal variables.

3.2.2 Generalization

A large number of states of a categorical variable can be a problem as it may mean that some of them are very rare, so rare that the training data set may not contain combinations of these states with other typical values of other variables. We could remove such states, but in order not to lose information, the generalization of rarely occurring states is more often used, i.e. we replace many states with one. Often these rare states are the result of errors, such as typing or writing the same information in different forms.

3.2.3 Rounding

We round to decimals, units, hundreds, etc, thus frequently changing the unit of the variable. This is a similar process to generalization but applies to numerical variables. This leads to replacing the continuous variable (infinite set power) with a discrete variable (finite set power). Such a change simplifies the model and reduces the risk of its overfitting, as discussed in section 5. Variables that are expressed in a unit so small that there are concerns about measurement errors should be rounded.

3.2.4 Discretization

The reduction of the number of values of a numerical variable consists in its transformation into a categorical ordinal variable. For this purpose, the indicated ranges of values of the original variable are assigned to new states. Discretization is a transformation that can significantly change the distribution of a variable's value and reduce the amount of information contained in it. The three most popular discretization methods:

- division into fixed width intervals, here we try to keep the original variable distribution,

- division into different ranges, here we try to keep the entropy of the variable,
- division into ranges with defined boundary values, this division is used to introduce knowledge from the field of the experiment to the model.

3.2.5 Scaling

Scaling a numeric variable can be to reduce these variables to a common scale, often to a range of values between 0 and 1. It can also be aimed at changing the distribution of the variable, for example reducing it to a standard natural distribution. Some machine learning algorithms are very sensitive to differences in variable scales, such as artificial neural networks that automatically scale input variables by default, or k-means clustering algorithms that favor variables with a large range at the expense of variables with a small range. Scaling methods:

- Min-Max method
$$x_i' = \frac{x_i - \min(x_i)}{max(x_i) - min(x_i)}$$
This way we get values from 0 to 1 or from -1 to 1 if there were negative values and we dock the original distribution. This method is sensitive to outliers.

- The IQR method consists in replacing the lowest value with the first interquartile, and the range with a quartile range. It is based on the difference between the 75th and 25th percentiles.
$$x_i' = \frac{x_i - Q1(x_i)}{Q3(x_i) - Q1(x_i)}$$
This is the better method for data with outliers.

- Logarithmic scaling — it is used to flatten the distribution of variables assuming outliers. Scaled variables cannot take negative values.
$$x_i' = \frac{e^x}{1 - e^x}$$

- Standardization — transforms the variable so as to obtain a variable with a normal distribution (average 0 and standard deviation 1)
$$x_i' = \frac{x_i - \bar{x_i}}{\sigma}$$
After standardization, values less than the mean will have negative values, and values greater than the mean will be positive. Usually standardized values are in the range from -4 to 4.

4 Review of selected methods of supervised machine learning

In this section, we will focus on a more detailed approximation of a few selected methods of supervised machine learning, paying attention to their application to two types of problems that we can consider when talking about machine learning: classification and regression. Although, as we wrote at the beginning of this chapter, we have many methods of supervised machine learning, we will focus on two, namely the widely used artificial neural network and linear regression models. We will also discuss methods for evaluating the classification and regression models. Finally, we will introduce the applications of the Python methods discussed here.

Classification is the oldest and most widely used method of supervised machine learning. The purpose of the classification is, for a given set of training data $(x_1, \ldots, x_k, y)$ to find the classifier hypothesis $h : X \to Y$, which assigns the class $y \in Y$ to the object $x \in X$. The data must be in the form of a vector of input variables x described by the output variable y. Input variables can be continuous numeric or discrete. The output variable is discrete and takes at least two states. The states of the output variable are called class labels, or classes for short. If the result variable takes two states then it is a binary classification, if it has more states it is a multi-class classification.

Regression is understood as a supervised learning method that consists in finding correlations hidden in the data $(x_1, \ldots, x_k, y)$ to find the hypothesis $h : X \to Y$ in order to most accurately estimate the output variable y which is continuous numeric in this case.

4.1 Artificial neural network

Artificial intelligence was inspired by brain research conducted in the 1940s by two neuroscientists: Warren McCulloch and Walter Pist. The basic cell of the brain is the neuron. Neurons transmit signals to each other via synapses. The counterpart of a neuron in an artificial neural network is a node, which is an independent unit of computing. It multiplies the input variables by their weights (parameters) and then sums them up. The value thus obtained is passed to the activation function f. The model of the artificial neural network node is shown in the figure 4, at the same time it is the simplest model of the artificial neural network — the Perceptron. Perceptron was proposed by Rosenblat in 1957 [20]. Perceptron is a model of a linear binary classifier, i.e. he can find the boundary between two linearly separable (separable by one: a straight line, a plane, and in spaces with more dimensions — a hyperplane) classes of examples [15] and consists of one McCulloch-Pist neuron (the first mathematical model of an artificial neural network neuron).

A node has multiple inputs and one output. The output value is calculated as the weighted sum of the input values.

$$\hat{y}_i = f\left(\sum_{j=1}^{k} w_j x_{i,j} + w_0\right)$$

The input signals are 0 or 1 and the activation function is a threshold function (eq. 10). If the weighted sum of the input signals exceeds the threshold value, the output will be 1, otherwise the neuron returns -1 or sometimes 0. The weight w_0 plays the role of the threshold value. We get the following mathematical model:

$$o(x_{i,1}, \ldots, x_{i,k}) = \begin{cases} 1 & \text{if} \quad \sum_{j=1}^{k} w_j x_{i,j} + w_0 \quad > 0 \\ -1 & \text{if} \quad \sum_{j=1}^{k} w_j x_{i,j} + w_0 \quad \leq 0 \end{cases} \tag{10}$$

Learning the perceptron is about changing weights when the answer was incorrect.

$$w_j = w_j + \triangle w_j$$

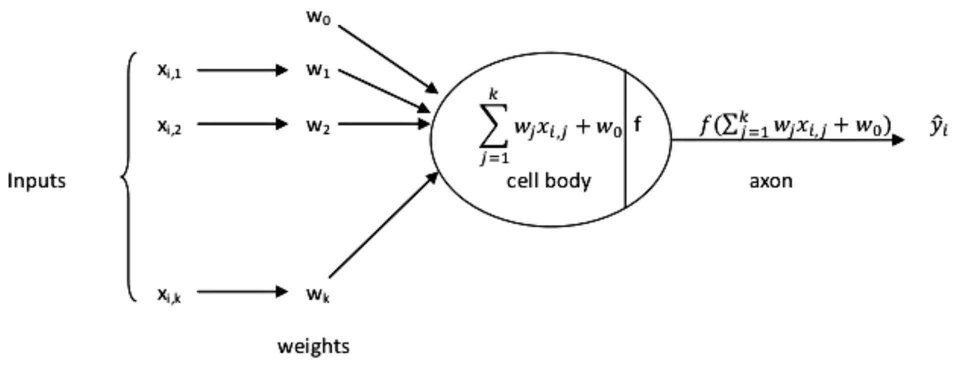
where

$$\triangle w_j = \eta(y_i - o)x_{i,j}$$

η is the learning coefficient, the value of which is usually set to 0.1.

For a correctly classified example, the weights are not changed. If the model incorrectly classifies an example from class 1 to class -1, then the difference $y_i - o$ will be 2, and incorrect classification of an example from class -1 to class 1 will give a difference of -2. Learning is as follows:

- reading and prediction of the next example;
- checking the prediction result:
  - if it is consistent with the result variable value, the weights are not changed;
  - if it is -1 and should be 1, the weights of the example variables are incremented;



w₀
x_{i,1}
w₁
x_{i,2}
w₂
Inputs
x_{i,k}
w_k
weights
$\sum_{j=1}^{k} w_j x_{i,j} + w_0$
f
cell body
$f(\sum_{j=1}^{k} w_j x_{i,j} + w_0)$
$\hat{y}_i$
axon

Figure 4: The Perceptron: input $x_i \in R^k$, k connection weights w_j and activation function f. The result output is given by $\hat{y}_i = f(\sum_{j=1}^{k} w_j x_{i,j} + w_0)$

– if it is equal to 1 and should be -1, the weights of the example variables are decreased.

The procedure is repeated for all examples. We repeat the learning process many times on the set of training data. Increasing the weights causes that after some time the sum of weighted values for the analyzed example will exceed the threshold function, so the example will be correctly assigned to class 1, while decreasing the weights will cause that after some times the sum of weighted values for the analyzed example will fall below the threshold, so the example will be correctly assigned to class -1. The applied learning method guarantees that weights will be able to correctly classify all the examples, as long as the correct solution exists, i.e. the examples are linearly separable.

For example, teaching perceptron the AND function (logical conjunction) is possible. Conjunction returns true only when both factors are true (1), so it is enough to set the threshold value w_0 to 0 so that the weighted sum of both variables is greater than zero only when both variables are positive. Similarly, the perceptron of the OR (logical alternative) function, which is false only when both components are false (-1), can be simulated. It is enough to take the threshold value w_0 equal to 1. Consider the XOR function. This function returns true if the factor values are different, and false if they are the same. Let's look at the graphical representation of the XOR function (fig. 5) to see that there is no single straight line that separates the cases belonging to different classes (false and true).

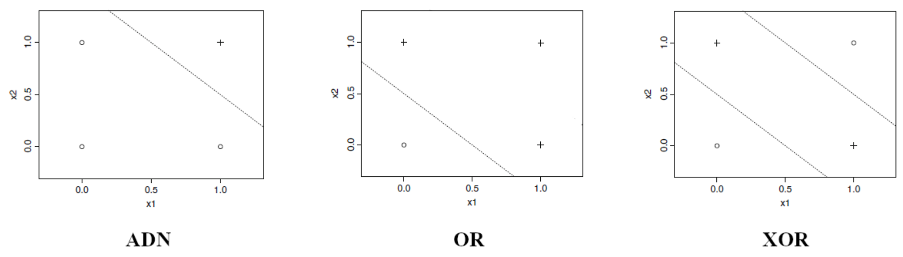


Figure 5: On the axes there are values of both variables, pluses symbolize true values and circles — false values of AND, OR and XOR functions. The XOR function is not linearly separable

Many real problems are not linearly separable. You can deal with them by adding extra neurons. Each neuron in a hidden layer divides the example space into two subspaces. The result of these splits is summed up in the result node. As a result, we obtain a subspace which is a convex polyhedral set. If one of the two subsets of examples with true and false values is contained in this convex set, it can be placed in the appropriate class.

The idea of a one-way artificial neural network with hidden layers is called the Multi-layer Perceptron. The question of updating the hidden layer weights remains to be resolved. Learning a network requires knowing the expected responses of the neurons of each layer. Unfortunately, they are known only for the output layer, they are not defined for hidden layers. This problem contributed to the first crisis in the development of artificial neural networks, blocking the ability to effectively learn multilayer networks [17]. Only the development of a method that allowed the mathematical determination of the error made by the neurons of the hidden layers, based on the error of the output layer, and using it to correct the weights of neurons of these layers, allowed for the development of artificial neural networks. A method that was intensively worked on in the 1960s and 1970s is called the error backpropagation method [14]. It was popularized by G. Hinton [8, 9] and its idea is commonly used to train multilayer networks.

The error back propagation algorithm defines the weight correction procedure in a multi-layer network using gradient optimization methods. The correction of the net weight vector is based on the minimization of the loss function, which was defined as the sum of squared errors at the network outputs. The learning cycle using the error backpropagation method consists of the following stages:

1. Determining the response of the output layer neurons and hidden layers to a given input signal.
2. Determining the error made by neurons located in the output layer and sending it towards the input layer.
3. Updating weights.

The responses of all output neurons of the network are described as follows:

$$\hat{y}_i^{out} = f\left(\sum_{j=1}^{l} w_{i,j} y_j^{out-1}\right)$$

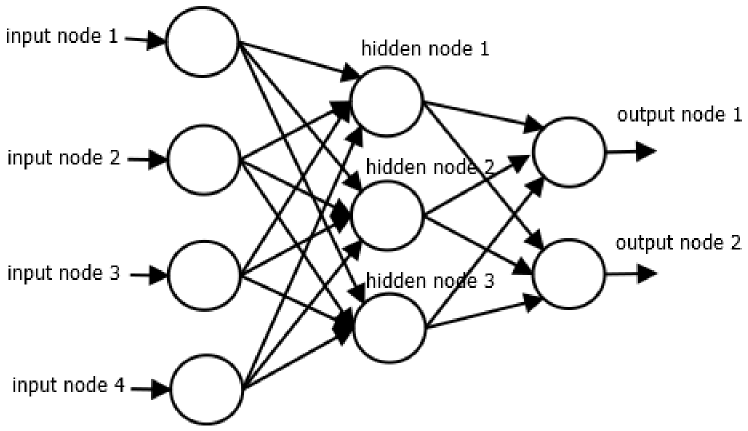


Figure 6: One-way neural network with one hidden layer where each neuron is connected to all neurons from adjacent layers

where l is the number of neurons in the layer preceding the output layer. Then we need to calculate the errors of all neurons of the output layer:

$$\delta_i^{out} = y_i - \hat{y}_i^{out}$$

The next step is to calculate the errors in the hidden layers (remember that in order to find the error in the $h-1$ layer, you need to know the error in the following layer - h):

$$\delta_j^{h-1} = \frac{df(u_j^{h-1})}{du_j^{h-1}} \sum_{k=1}^{p} \delta_k^h w_{k,j}^h$$

where p is the number of neurons in the following layer h, u_j^{h-1} is the output of j-th neuron hidden $h-1$ layer (weighted sum of input values from hidden $h-2$ layer). Finally, the weights are updated:

$$w_{j,i}^{h-1} = w_{j,k}^{h-1} + \eta \delta_j^{h-1} y_i^{h-1}$$

In practice, the method of backward error propagation is very effective, unfortunately its disadvantage is the long learning time. The course of the learning process of the backpropagation network strongly depends on the value of the learning coefficient η, its too high value often leads to process discrepancies, and too low a very long time. Unfortunately, there are no rules that could precisely define its value.

We can easily extend a binary classification to a multi-class classification using one of the two methods:

- The One-vs-Rest (OvR) strategy splits a multi-class classification into one binary classification problem per class, where this class is treated as a positive class and examples from the remaining classes are considered to be negative class objects. Another classifier is used to classify new, untagged training data (assigned to a negative class) and so on. As a result, we construct l classifiers, where l is the number of class labels.

- The One-vs-One (OvO) strategy splits a multi-class classification into one binary classification problem per each pair of classes. As a result, n classifiers are created, according to the formula: $n = l(l-1)/2$ for an l-class problem. During classification, a voting scheme is used: all n classifiers select their class. The class that has been selected by the largest number of classifiers is selected.

The use of neural networks for regression problem requires the use of a learning procedure other than that used in perceptron, so that the prediction output is the value of the target variable. To do this, you need to replace the threshold activation function with another one. Choosing the linear function $w_0 + w_1x_1 + \ldots + w_kx_k$ would reduce any deep artificial neural network to a single neuron. Thus, a nonlinear differentiable activation function must be chosen. Unfortunately, the use of such an activation function causes the MSE loss function to gain local minima, i.e. we lose the guarantee of finding the optimal solution [17]. After 20 years, thanks to, among others, the works of G. Hinton, it became clear that local minima are not such a serious problem. First of all, finding one of the local minima often gives a fairly good solution. Secondly, optimization of the fastest gradient descent method allows to avoid local minima by changing the learning coefficient. In low with one or more hidden layers of artificial neural networks the most popular functions are sigmoid:

$$f_{sigmoid} = \frac{1}{1+e^{-y}}$$

where $y = w_0 + w_1 * x_1 + w_2 * x_2 + \ldots + w_k * x_k$ and returns values from 0 to 1, and hyperbolic tangent:

$$f_{htan} = \frac{e^y - e^{-y}}{e^y + e^{-y}}$$

returns values from -1 to 1.

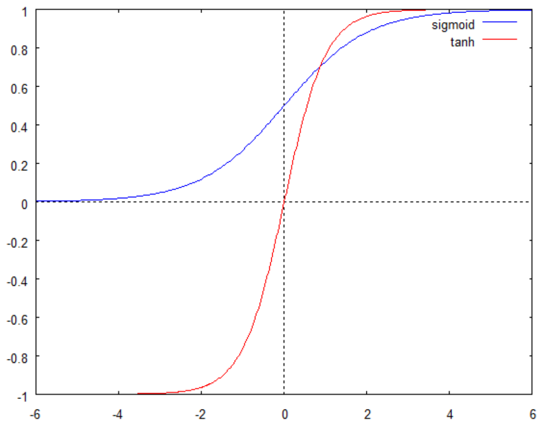


Figure 7: Plot of sigmoid and tangent activation functions

The sigmoid function is used in the output layer, and hidden layers use both. In deep neural networks with several dozen or more hidden layers, the most frequently used activation function is ReLU (Rectified Linear Unit) [7], which is a combination of a threshold function and a linear function and returns 0 on the output or the input signal if it exceeds the threshold value:

$$o(x) = \begin{cases} w_0 + w_1 x_{i,1} + \ldots + w_k x_{i,k} & \text{if } w_0 + w_1 x_{i,1} + \ldots + w_k x_{i,k} > 0 \\ 0 & \text{if } w_0 + w_1 x_{i,1} + \ldots + w_k x_{i,k} \leq 0 \end{cases}$$

This function is simple and has little effect on learning time.

4.2 Linear regression models

Regression analysis is typically used for forecasting and prediction, and its application largely covers the area of machine learning. It can be used to determine causal relationships between the independent variable and the dependent variable.

Regressions show the relationship between a dependent variable and a fixed set of different variables. The main idea of linear regression is presented on page 11.

Linear regression methods are among the most widely used machine learning models. Their advantages include simplicity, which makes learning fast and does not require large computing power or large RAM memory. Linear regression models are likely to work if the input data meets four requirements:

1. Input variable must be linearly correlated with the output variable. The strength of linear correlation between numerical variables can be measured by the r-Pearson coefficient, the value of which range from -1 to 1, where -1 means one hundred percent negative correlation (with an increase in the value of one variable, the value of the other variable decreases), 0 no correlation, 1 means one hundred percent positive correlation (as the value of one variable increases, the value of the other variable also increases).
2. The values of the input variables must be normally distributed, that is, the data must not contain outliers.
3. Input variables cannot be strongly correlated with each other.
4. Input variables cannot have autocorrelation, i.e. the current value cannot depend too strongly on the preceding values. To fulfill them, monotonic variables such as timestamps, cycle numbers should be removed.

We will focus here on linear regression models, especially on two: logistic regression, which is used for classification problems, and ordinary least squares linear regression, for regression problems.

Logistic regression is supervised learning, but contrary to its name, it is not a regression but a classification method. It assumes that data can be classified (separated) by a line or n-dimensional plane, that is, it is a linear model. Classification is performed by calculating the value of the output variable as the value of a first-order polynomial of the following form:

$$y = w_0 + w_1 * x_1 + w_2 * x_2 + \ldots + w_k * x_k$$

where x is the input data (vector of feature), w_i weights assigned to these features, and k is the number of input variables (features). The next step is to pass the result y obtained through a logistic function (e.g. sigmoid or hyperbolic tangent).

The sigmoid function gives values in the range [0, 1] and its result can be interpreted as a probability. After obtaining such a value, we can classify the input data to group A or group B on the basis of a simple rule: if $y >= p$ then class A, otherwise class B. The parameter p means the threshold value, defaults to 0.5.

To carry out the learning process, we can use the fastest gradient descent method, so we need the loss function l. In the case of binary classifiers, the error consists in returning a class other than the target one, that is assigning examples of class A to class B or vice versa. In the simplest case, model training could consist in minimizing errors understood in this way, i.e. using a threshold (zero-one) loss function. Unfortunately, it is not suitable for the fastest gradient descent method. We need a differentiable convex function. The three most frequently used loss functions in the classifiers are:

- hinge function: $l_{hinge}(y_i, \hat{y_i}) = \max[0, 1 - y_i\hat{y_i}]$
- logistic function $l_{\log}(y_i, \hat{y_i}) = \log(1 + \exp[-y_i\hat{y_i}])$
- square function $l_{sqr}(y_i, \hat{y_i}) = (y_i - \hat{y_i})^2$

By ordinary least squares linear regression we understand the aforementioned linear regression model with a hypothesis in the form of a first order polynomial as in the case of logistic regression, except that the output variable may be a continuous numeric, with loss function l_{sqr} and fastest gradient descent method used to find the correct hypothesis, page 13.

4.3 Evaluation of machine learning models

There is no universal machine learning algorithm. This statement is a consequence of the NFL (no free lunch) theorem of David Wolpert and William Macready appears in [16]. In its simplest version, the NFL theorem is as follows: the generalized average error of any machine learning model is 50%, which is equal to random guessing. This can be proved by noting that for each h hypoethesis which has the generalized error of $0.5 + \delta$, there is an alternative hypothesis h' which has the generalized error of $0.5 - \delta$ so the mean of these hypotheses will be 50%. By the generalized error we understand the accuracy of the model prediction on the test data.

So we should not assume that some machine learning algorithm is better able to deal with the problem than another machine learning algorithm. Moreover, a model that coped well with one task, i.e. its generalized error was high, may not solve another problem, i.e. the generalized error of a model with the same architecture will be low. So there is no perfect combination of model hypermarameter. The answer to the question why sometimes more complex models (such as multilayer neural networks) cope with tasks worse than simple models (logistic regression) is the issue of overfitting, and more on this in section 5.

In practice, this means that we have to build and compare many models with each other in order to choose the best one.

4.3.1 Evaluation of classification models

We most often evaluate models in terms of the accuracy and credibility of the predictions. Binary classifiers assign examples to one of the two classes, we can symbolically call them negative and positive. So we have three possibilities, an example can be correctly classified, an example of a positive class can be assigned a negative class, and an example of a negative class can be assigned a positive class. By counting individual cases, we can create a model error matrix.

In the table, the symbols TP, TN, FP and FN mean, respectively: the number of correctly classified examples as positive (True Positive), the number of correctly classified examples as negative (True Negative), the number of incorrectly classified examples as positive (False Positive), called errors of the first type and the number of misclassified examples as negative (False Negatives) known as second type errors. Based on the error matrix, the following types of measures are introduced to evaluate classification models.

Table 1: Error matrix. In the top row we have true class labels, and the labels in the first column of the following rows correspond to the class predictions

	1	0
1	TP	FP
0	FN	TN

- Accuracy: measures the proportion of correct classifications to all classifications.

  $$ACC = \frac{TP + TN}{TP + FN + FP + TN}$$

  Accuracy is sometimes expressed in the number of classification errors (error rate)

  $$ER = 1 - ACC = \frac{FP + FN}{TP + FN + FP + TN}$$

- Positive Predictive Value: measures the proportion of correct positive classifications to all positive alignments. It answers the question: How many positively classified cases are well classified?

  $$PPV = \frac{TP}{TP + FP}$$

- Negative Predictive Value: Positive Predictive Value measures the proportion of correct negative classifications to all positive alignments. It answers the question: How many of the negatively classified examples have been classified well? We use PPV i NPV when we want to limit errors of the first type.

  $$NPV = \frac{TN}{TN + FN}$$

- Sensitivity (true positive rate): measures the proportion of correctly classified positive examples against all positive examples. It answers the question: How many of the positive examples have been classified well? We use it when we want to limit errors of the second type.

  $$TPR = \frac{TP}{TP + FN}$$

- The equivalent of sensitivity for a negative class is specificity, also called true negative rate.

  $$TNR = \frac{TN}{TN + FP}$$

- F-score: is the harmonic mean of precision and sensitivity. This measure takes values from 0 to 1, where 1 means a faultless model. We use it to evaluate the model if the costs of both types of errors are close.

  $$F\text{-}score = 2\frac{PPV * TPR}{PPV + TPR}$$

- Cohen's Kappa coefficient: compares the measured accuracy of the model with the expected, i.e. random guessing validity. The expected validity is calculated by multiplying the number of examples of a given class by the number of examples assigned to that class by the model and dividing the result by the number of all examples. This measure takes values from -1 to 1. Value 0 means a random model, the closer to 1 the better the model, negative values mean the model is worse than random.

$$Kappa = \frac{ACC - expACC}{1 - expACC}$$

The quality of classifiers can also be presented using curve in a two-dimensional space the ROC (Receive Operating Characteristics). On the abscissa axis we place: $1 - TNR$, and on the ordinate axis TPR. Properties of the ROC space:

- the point with the coordinates (0, 0) represents a model that always returns negative predictions;
- point (1, 1) represents a model that always returns positive predictions;
- point (0, 1) represents an ideal model;
- points on the diagonal, i.e. the $TPR = 1 - TNR$ line, represent a random model.

The correct models are those represented by points in the upper part of the chart. The more on the top left side the model is, the better it is — the greater the specifity and sensitivity. To plot the ROC curve, prediction probabilities are needed, based on which we classify the cases using a threshold value (e.g. 0.5). By changing the threshold value of the model prediction we obtain a sequence of points representing the classifier in the ROC space. By combining these points, we get the ROC curve, it shows the relationship between the number of correctly classified examples from the positive TP class and the number of incorrectly classified examples from the negative FP class. The advantage of the ROC curve is its independence from the adopted prediction threshold. The area under the ROC curve (AUC) is also used to compare classifiers. The ideal classifier has an AUC value of 1 and for a random classifier it is 0.5. AUC can be interpreted as the probability that the model will score a randomly selected positive item higher than a randomly selected negative item (Wilcoxon's test).

We covered the topic of evaluating binary classification models. When it comes to multiclass classification, the same is true. Their assessment is also based on the error matrix, in which the number of rows and columns corresponds to the number of classes. In the columns of the error matrix, we have counted frequencies of classes, and in the rows, the predictions are counted. The single values represent the number of class examples that have been assigned to the class by the model. Hence, the diagonal of the error matrix will contain correct predictions, the others contain model errors.

Metrics such as accuracy or sensitivity can also be calculated for each class separately treating all other classes as meta-class different, i.e. using the OvR approach. Then the metrics calculated for each class can be averaged. We are talking then about the so-called macro and micro measures. For example, the accuracy macro is expressed by the formula:

$$macroPPV = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i+FP_i}}{l}$$

where l is the number of classes. This type of measure is good for balanced data. However, when our data is not balanced, the better choice are micro measures that take into account the size of individual classes:

$$microPPV = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i + FP_i)}$$

4.3.2 Evaluation of regression models

Regression models estimate the actual value of the output variable, so it is best to evaluate them by comparing the prediction results with the true values of the output variable. Models built to solve the same task should be assessed on the basis of one criterion, often the same criterion is selected for the evaluation of the model that was used for the loss function. This approach can give the best results because the models learn to minimize the same error that we then use to evaluate them. The most common measures used to evaluate regression models are following.

- Mean absolute error (MAE) corresponding to the loss function l_{abs}.

  $$MAE(\hat{y}, y) = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n}$$

  where n is the number of examples, i another example, $\hat{y}_i$ the prediction of the output variable y_i the target value of the output variable. MEA takes values from 0 to infinity, lower values mean a more accurate model. This measure returns the result on the same scale as the output variable and is interpreted as an average model error.

- Root mean square error (RMSE) corresponding to the loss function l_{sqr}.

  $$RMSE(\hat{y}, y) = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

  RMSE takes values from 0 to infinity, lower values mean a more accurate model. It is never less than the MAE and it makes the model sensitive to large errors because the errors of the single predictions are squared instead of being averaged. Therefore, one large mistake can have a greater effect on the size of the measure than several small ones. It is characterized by a stronger punishment of large mistakes.

- Coefficient of determination R^2 (convergence coefficient $1 - R^2$)

  $$R^2(\hat{y}, y) = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y}$ is the mean of the actual values of the output variable. R^2 takes values from 0 to 1 and is a measure of the quality of the model fit to the training data. A value of 0 means that the model has no predictive power and is no better than the random model, the fit of the model is the better, then closer the value of R^2 is to 1. R^2 is often multiplied by 100 to get a percentage score on a scale of 0 to 100.

4.4 Python machine learning examples

We now move on to the sample Python applications discussed in this section of machine learning methods.

We start with some classifiers using the Python `sklearn` library. To train and then test our model, we will use data from the Iris data set. It is flower data set created in 1936 by Ronald Fisher. This set describes iris flowers using 4 attributes: the length and width of the petals and sepal, and assigns each flower its actual species. The flowers are divided into three species: setosa, versicolor and virginica, which are assigned values 0, 1 and 2 respectively in the data set. The data set contains a total of 150 examples, 50 for each species.

The first model that we will present will be the perceptron model . We will start with importing the `sklearn` library elements. We need the iris data set, and a class that creates a perceptron model.

We will import the method of automatically dividing data into a training and test set too.

```
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
```

Next we load the data:

```
x_var, y_var = X, y = load_iris(return_X_y=True)
```

There are three classes in the Iris data set of examples. The perceptron originally divides the examples into two classes. So let's choose from the data only those examples for which the output variable takes the value 0 and 1.

```
x_var_new, y_var_new = x_var[(y_var==0)|(y_var==1)],
y_var[(y_var==0)|(y_var==1)]
```

and divide them into the training set `x_train`, `y_train` and the test set `x_test`, `y_test` in the variables `x_train` and `x_test` describe the features of the examples, while `y_train` and `y_test` the output values, i.e. numerical representations of iris species.

```
x_train, x_test, y_train, y_test = train_test_split(x_var_new,
y_var_new, test_size=0.3, stratify=y_var_new)
```

The `train_test_split` function divides the data set randomly, randomizing 30% examples (`test_size = 0.3`) to the test set, making sure that the output variable values are evenly distributed between the training set and the test set (`stratify=y_var_new`).

The `Perceptron` implements a simple classification algorithm suitable for large scale learning which allows you to update weights based only on errors without additional regularization. We can set there several parameters, including

- `tol`, which by default takes the value: $1e - 3$ and defines the stop criterion when the improvement in the model prediction is less than the value set in `tol`, i.e. `loss > previous_loss - tol`;
- `eta0` (learning coefficient η), which defaults to 1 and stands for constant by which the updates are multiplied.

Now we will create a perceptron model with `eta0 = 0.1`

```
p = Perceptron(eta0=0.1)
```

and at the end train the model on the training data:

```
p.fit(x_train,y_train)
```

The `fit` method starts with a weight vector that contains small random numbers generated by a normal distribution with a standard deviation of 0.01. After fitting the model, we can check its operation on the test data:

```
yhat=p.predict(x_test)
```

and check how many times our model has made a mistake on the test data:

```
print('Incorrectly classified samples:%d'%(y_test!=yhat).sum())
```

In our case, we got the answer:

```
Incorrectly classified samples: 1
```

Now we build another classifier for data from Iris data set, using the Python `sklearn` library again and the `LogisticRegression` class performs linear regression using the logistic function l_{log} as the loss function. This model can also be used for multiple classification. The OvR method is used to extend the binary to multi-class classification as well as in the case of `Perceptron`. So we're going to split the entire iris data set into training and test data sets and apply the `LogisticRegression` model to divide the examples into three classes (as originally found in the iris database).

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
```

```
x_train, x_test, y_train, y_test = train_test_split(x_var, y_var,
test_size = 0.3, random_state = 0,stratify=y_var)
p = LogisticRegression(multi_class='ovr')
p.fit(x_train,y_train)
```

After fitting the model, we can check its operation on the test data:

```
yhat=p.predict(x_test)
```

and check how many times our model has made a mistake on the test data:

```
print('Incorrectly classified samples:%d'% (y_test!=yhat).sum())
```

In our case, we got the answer:

```
Incorrectly classified samples: 4
```

We will now check how the evaluation of our multiple classification `LogisticRegression` model.

For this purpose, we import from sklearn methods determining evaluation measures for classification problem:

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score
```

Since the Iris data set is balanced, there are 50 examples of each class, and the division of the data into the training set and the test set is also carried out in a balanced way, we will use macro measures(`average='macro'`).

```
print('ACC: %.2f'% accuracy_score(y_test,yhat))
print('PPV: %.2f'% precision_score(y_test,yhat,average='macro'))
print('TPR: %.2f'% recall_score(y_test,yhat,average='macro'))
print('F-score: %.2f'% f1_score(y_test,yhat,average='macro'))
print('AUC: %.2f'% roc_auc_score(y_test, p.predict_proba(x_test),
multi_class='ovr'))
```

we got:

```
ACC: 0.93
PPV: 0.94
TPR: 0.93
F-score: 0.93
AUC: 0.99
```

And next we build a linear regression model for regression problem using the Python `sklearn` library like earlier. To train and then test our model, we will use data from the Iris data set too. The data set, similarly to the classification, will be divided by randomly selecting 30% of the examples for the test set, using the rest as a training set.

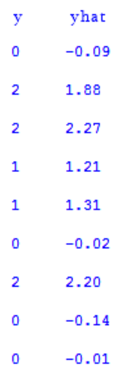
Now we will create a linear regression model and train the model on the training data. This model implements multiple linear regression.

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Now it remains to check the learning result of the model and evaluate it. We will run our model for a test dataset.

```
yhat = lr.predict(x_test)
```

The presetting result is shown in the figure 8.



y	yhat
0	-0.09
2	1.88
2	2.27
1	1.21
1	1.31
0	-0.02
2	2.20
0	-0.14
0	-0.01

Figure 8: Prediction result for several examples from the test data set

Let’s compute the errors for our linear regression model. We import the necessary methods:

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
print('Mean absolute error: %.2f',
% mean_absolute_error(y_test, yhat))  print('Root mean square error:
%.2f',
% math.sqr(mean_squared_error(y_test, yhat)))
print('Coefficient of determination: %.2f' % r2_score(y_test,
yhat))
```

The result obtained is:

```
Mean absolute error: 0.19
Root mean square error: 0.24
Coefficient of determination: 0.89
```

while this model errors and model fit for training data set was:

```
Mean absolute error: 0.16
Root mean square error: 0.21
Coefficient of determination 0.94
```

At the end we will use the Python `sklearn` library again to test the operation of an artificial neural network in order to perform the regression problem. We will be based on the Iris dataset to. We will use `MLPRegressor` class which implements the idea of Multi-layer Perceptron. The `MLPRegressor` method builds an artificial neural network model with the ReLU activation function and is based on a Mini-Bath Gradient Descent with a bath size equal to $\min(200, n)$, where n is the number of examples. There are less than 200 examples in our training set, so the weights will be updated after each epoch.

```
from sklearn.neural_network import MLPRegressor
p = MLPRegressor(max_iter=500)
p.fit(x_train,y_train)
yhat=p.predict(x_test)
```

Fitting the model took 160 iterations to fit model.

Now we will present the same error measurements as in the case of the `LinearRegression` model for model of artificial neural network regression:

The result of prediction for test data set is:

```
Mean absolute error: 0.19
Root mean square error: 0.24
Coefficient of determination 0.89
```

while this model errors and model fit for training data set was:

```
Mean absolute error: 0.17
Root mean square error: 0.22
Coefficient of determination 0.93
```

5 Methods of preventing overfitting and underfitting

An important consideration in learning the target function from the training data is how well the model generalizes to new data. Generalization is important because the data we collect is only a sample, it is incomplete and noisy. The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain.

We will refer to the Statistical Learning Theory [23, 15] a theoretical framework for understanding and assessing the learning process. The probability density function (PDF) helps us to estimate the probability of a given x event, assuming a value in a certain range. Given the probability density function $f(x)$, the probability of the interval A is given by the area under the function $f(x)$ on the interval A:

$$P(A) = \int_A f(x)dx$$

where is required:

$$f(x) \geq 0 \quad \text{for all } x, \qquad \int_{\mathbb{R}} f(x)dx = 1$$

The hypothesis $h : X \to Y$ is an approximation of the special probability density function $P(x, y)$, also known as the join probability density function, which describes the join behavior of two variables, X the input space of examples and Y as the output variable, and is required to satisfy the following conditions:

$$P(x, y) \geq 0 \quad \text{for all } x, y, \qquad \int\int P(x, y)dxdy = 1.$$

Next we will refer to the concept of a performance measure for any hypothesis built on a certain sample from all possible examples from the $X \times Y$ space that have a fixed join probability density function $P(x, y)$. This measure is called expected risk and is defined by the expected value of the loss function l of the h hypothesis when all possible examples of $(x, y) \in X \times Y$ are evaluated.

$$R(h) = E(l(x, y, h(x))) = \int l(x, y, h(x))dP(x, y)$$

Difference $|R_{emp}(h) - R(h)|$ allows us to understand how a certain hypothesis h works on unseen examples, i.e. how this hypothesis behaves with new data and tells us when the empirical risk is a good estimate of $R(h)$ supporting the selection of the best hypothesis based on R_{emp}. We say that a given hypothesis h generalize when this difference is sufficiently small which means it can deal with seen and unseen examples alike. Let F be a set of possible hypotheses for our machine learning model, F_{all} be a set of all possible hypotheses (all possible functions from X to Y). Let f_{Bayes} is the best possible hypothesis for our problem from the set F_{all}. The Bayes consistency states that, as the training data set size increases, the learning algorithm must approach the best hypothesis, i.e.:

$$\lim_{n\to\infty} E[l(x, y, h(x))] = R(f_{Bayes})$$

In practice, F_{all} is outside the model range, the model has a range limited to a certain bias $F \subseteq F_{all}$. We will say that the learning algorithm is consistent when it coincides with the best hypothesis in F:

$$\lim_{n \to \infty} E[l(x, y, h(x))] = R(f_{best})$$

Let us introduce the following concepts to further considerations about situation when the resulting model fails to learn, i.e. it will overfit or underfit examples. Let's define two learning errors:

- Estimation error — illustrates how far our solution h_i is from the best possible hypothesis f_{best}. This error is the result of the uncertainty contained in the training data set (noise, data errors): $R(h_i) - R(f_{best})$.
- Approximation error — illustrates how far the best possible hypothesis f_{best} is from the best hypothesis out of the whole pool F_{all}, f_{Bayes}. This error is the result of an error imposed by the algorithm (bias): $R(f_{best}) - R(f_{Bayes})$.

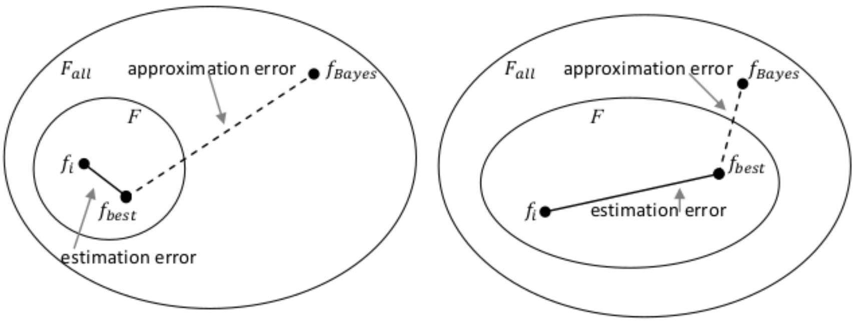


Figure 9: Illustration of estimation and approximation error

The total error — illustrates how far our solution h_i is from the best hypothesis f_{Bayes} is defined in terms of estimation and approximation errors:

$$R(h_i) - R(f_{Bayes}) = (R(h_i) - R(f_{best})) + (R(f_{best} - R(f_{Bayes}))).$$

From this perspective, we can define underfitting and overfitting as follows:

- underfitting — for a small bias F, estimation error is small but approximation error is large;
- overfitting — for a large bias F, estimation error is large but approximation error is small.

For example for problem XOR, the perceptron results in underfitting since f_{best} is not even enough to represent the training data. On the other hand, if we take a neural network with a large number of neurons in hidden layers, the result can be a much larger hypothesis subspace than necessary, leading to overfitting. In fact, convergence to f_{Bayes} becomes more difficult due to estimation error. The resulting function is likely to be much more complex than the problem requires, causing little empirical risk but high expected risk when the invisible examples represent any little variation in the data.

This shows how important it is to select a model, a machine learning algorithm. We should use the bias subspace that will be sufficient and necessary for the given problem. By using an insufficient or overcomplex bias we can fall into under and overfitting respectively. Overfitting and underfitting are the two biggest causes for poor performance of machine learning algorithms.

Underfitting is often not discussed because it is easy to spot, just take a good evaluation measure. The solution is to try alternative machine learning algorithms.

Overfitting refers to a model that models training data too well. It is a threat to high bias models, i.e. models in which there are many potential hypotheses, hence they are more complex. Good fit to training data also means adjusting to the noise contained in it or random data fluctuations. The models most prone to overfitting are non-parametric and non-linear models, which are more flexible in learning the target function. Therefore, additional parameters and techniques are used in such algorithms to prevent overfitting.

The most popular forms of overfitting prevention are $l1$ and $l2$ regularization. For the loss function:

$$l = (wx + b - y)^2$$

$l1$ regularization (called the Lasso Regularization) is adding a term $\alpha|w|$ to the loss function:

$$l1 = (wx + b - y)^2 + \alpha|w|$$

and $l2$ regularization (called Ridge Regularization) is adding a term αw^2 to the loss function:

$$l2 = (wx + b - y)^2 + \alpha w^2$$

for $\alpha > 0$.

With the gradient descent for $l1$ regularization we have:

$$w' = w + \lambda \frac{\partial l1}{\partial w} = w - \lambda(2x(wx + b - y) + \alpha \frac{d|w|}{dw})$$

$$= \begin{cases} w - \lambda(2x(wx + b - y) + \alpha & w > 0 \\ w - \lambda(2x(wx + b - y) - \alpha & w < 0 \end{cases}$$

and for $l2$ regularization:

$$w' = w + \lambda \frac{\partial l2}{\partial w} = w - \lambda(2x(wx + b - y) + 2\alpha w)$$

Note that the fit of w depends on the model (w at start and b) and the data (x and y). Updating the weights only based on the model and data can lead to overfitting. By adding the predefined α parameter, the final w value is influenced not only by the model and data, but also by a value independent of the model and data. In this way, we can prevent overfitting if we set an appropriate α value, although too high a value will result in a serious mismatch of the model.

How does α affect the model for regularization $l1$: if w is positive, the regularization parameter $\alpha > 0$ will make w less positive by subtracting α from w. Conversely, if w is negative, α will be added to w, making it less negative. Hence, it causes a shift of w to 0. This would obviously be pointless in the case of linear regression with one input variable, but in the case of more complex models, such as multivariate regression, it works in such a way that some features for which w_i closes to 0 cease to affect the model or will be completely eliminated which simplifies the model preventing overfitting.

In the case of regularization of $l2$, we always decrease w, in this case we will not reduce the number of model features, but we will reduce their influence on the model by reducing their coefficients. Both types of regularization are helpful, which is why a combination of them, ElasticNet, is often used:

$$ElasticNet = (wx + b - y)^2 + \alpha(\beta|w| + \frac{1-\beta}{2}w^2)$$

where β is the mixing parameter between $l2$ ($\beta = 0$) and $l1$ ($\beta = 1$) regularization.

References

[1] F. J. Anscombe, Graphs in statistical analysis. American Statistician., 27, 17–21, 1973.

[2] Léon Bottou, Frank E. Curtis, Jorge Nocedal, Optization Methods for Large-Scale Machine Learning, 2018, https://arxiv.org/pdf/1606.04838.pdf

[3] H. Daumé Course in III, A Course in Machine Learning, Self-published, 2017

[4] P. Domingos, A Few Useful Things to Know about Machine Learning, 2012, https://homes.cs.washington.edu/ pedrod/papers/cacm12.pdf

[5] I. Goodfellow, J. Pouget-Abadie , M. Mirza, X. Bing, D. Warde-Farley,S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks, 2014, http://arxiv.org/pdf/1406.2661.pdf

[6] M. Hardt, B. Recht, Y. Singer, Train faster, generalize better: Stability of stochastic gradient descent, Proceedings of The 33rd International Conference on Machine Learning, PMLR 48:1225–1234, 2016.

[7] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, In Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814, 2010

[8] G. E. Hinton, S. Nowlan, D. Plaut, Experiments on learning by back-propagation. Technical Report CMU-CS-86-126. Department of Computer Science, Carnegie-Mellon University, 1986

[9] G. E. Hinton, D. E. Rumelhart, R. J. Williams, Learning representations by back-propagating errors. Nature, 323, 533–536, 1986

[10] J. A. Hartigan, and M. A. Wong: A K-Means Clustering Algorithm, Applied Statistics, Vol. 28, No. 1, pp. 100–108, 1979

[11] G. Hughes, On the mean accuracy of statistical pattern recognizers, IEEE Transactions on Information Theory. 14 (1): p. 55–63, 1968

[12] S. T. Konstantnos Koutroumbas,Pattern Recognition (4th ed.), Burlington, ISBN 978-1-59749-272-0, 2008

[13] H. Liu; H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, The Springer International Series in Engineering and Computer Science Ser., New York, NY: Springer, ISBN 879-1-4613-7604-0, 1998

[14] S. Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's Thesis (in Finnish), Univ. Helsinki, 1970. See chapters 6–7 and FORTRAN code on pages 58–60. PDF. See also BIT 16, 146–160, 1976

[15] R. F. de Melo, M. A. Ponti, Machine Learning. A Practical Approach on the Statistical Learning Theory, Springer, ISBN 978-3-319-94988-8, https://doi.org/10.1007/978-3-319-94989-5, 2018

[16] W. G. Macready, D. H. Wolpert, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation 1, 67, 1997

[17] M. Minsky, S. Papert, Review of 'Perceptrons: An Introduction to Computational Geometry', 1969

[18] S. Na, L. Xumin and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm", 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, pp. 63–67, doi: 10.1109/IITSI.2010.74, 2010

[19] G. Pang, C. Shen, L. Cao, and A. van den Hengel, Deep Learning for AnomalyDetection: A Review.ACM Comput, Surv.1, 1, Article 1 (January 2020), https://doi.org/10.1145/3439950

[20] F. Rosenblatt, The perceptron: a perceiving and recognizing automaton, Technical report 85-460-1, Cornell Aeronautical Laboratory, 1957

[21] C. E. Shannon, A Mathematical Theory of Communications, The Bell System Technical Journal, 1948, http://math.harvard.edu/ ctm/home/text/others/shannon/entropy/ entropy.pdf

[22] V. N. Vapnik, Statistical Learning Theory Adaptive and Learning Systems for Signal Processing, Communications, and Control, Wiley, Hoboken, 1998

[23] V. N. Vapnik, The Nature of Statistical Learning Theory, Information Science and Statistics, Springer, New York, 1999

[24] Hadley Wickham, Tidy Data, Journal of Statistical Software, Volume 59, Issue 10, August 2014, https://www.jstatsoft.org/article/view/v059i10/v59i10.pdf

[25] Z. Zhang, Introduction to machine learning: k-nearest neighbors, Ann Transl Med, 4(11): 218. doi:10.21037/atm.2016.03.37, 2016

Automatic Syllable Repetition Detection Methods in Continuous Speech

Adam Kobus[*]
Ireneusz Codello
Wiesława Kuniszyk-Jóźkowiak
Grzegorz Marcin Wójcik

1 Introduction

The stuttering has a significant influence to the quality of stutterers life. It could cause the self-imposed isolation, permanent stress and low self-esteem, which could effect the whole social status, wealth and it could exclude stutterers from multiple proffessions. The stuttering could manifest in many different ways: as prolongations, interjections, word repetitions, syllable repetitions, blockades. Automatic detection of the stuttering will provide multiple benefits for the affected persons. They will be able to train the proper speech on their own more effectively, they will have fast feedback about their achievements and they will feel that they have more influence to their well-being. This paper shows the current works on automatic disfluency detection. Additionally it shows two recent works of the authors about syllable repetition detection with the use of LPC [1] and CWT [2].

1.1 Related work

There were several approaches to the syllable detection problem. First attempts were made by Howell [3–5] in 1995 and 1997 but it is still not fully resolved problem. Probably the reason is that the speech recognition problems are solved without syllable discrimination.

First attempts to the automatic detection of disfluency were based on speech fragments selection. Such approach were proposed in several articles through the two decades [6–19].

Regardless of this method researchers as Howell [3–5] and the team Kuniszyk-Jóźkowiak, Codello, Kobus, Suszyński and Wiśniewski [1, 2, 20–28] tried

[*]Corresponding author — adam.kobus@mail.umcs.pl

to detect disfluencies without the word extraction. Their material contained 4s long recording with the only information if that speech sample contain disfluent parts. Features extracted from the speech sample but still arranged over time were compared with the classified examples and it gave good results.

Specific problem is the syllable repetition. Despite that it is defined as the stuttering the speech the signal could be very fluent and very similar to the proper speech. Authors proposed two detection methods of this disorder. One is based on the linear prediction with k-means algorithm used as a classifier [1]. Second method is based on the wavelets and a correlation method as a classifier [2]. In this paper authors extend these results on the larger and more differential material.

Table 1: Research summary of speech disorders automatic detection

Year/Author	Features	Classifiers	Best results:
1995 Howell, Sackin [3]	Manual segmentation. Spectrum from 19-channel vocoder, correlation coefficients, speech envelope	ANN (Artificial Neural Network) classifier.	Repetitions: 82% Prolongations: 77%
1997 Howell, Sackin [4, 5]	Manual segmentation. Whole word and part word length, its count and their energy variation. Silence length, its count and their energy variation. Spectrum coefficients.	ANN classifier.	Fluent: 95% Repetitions: 53% Prolongations: 62%
2000 Nöth, Niemann [29]	Manual segmentation. HMM (Hidden Markov Model).	ANN classifier	Repetitions: - Prolongations: - Unwanted silences: -
2003 Czyżewski, Kaczmarek [7]	Manual segmentation. Cosine transform spectrum. Frequency of a base tome, first/second/third formant. Amplitude of the first/second/third formant.	ANN and rough set classifiers.	Stop-gaps: 91% Repetitions: 65% Prolongations: 97%
2003 Suszyński, Kuniszyk-Jóźkowiak [14]	Automatic segmentation. FFT (Fast Fourier Transform) spectrum, 1/3 octave filters	fuzzy logic	Prolongations: 91%
2004 Suszyński, Kuniszyk-Jóźkowiak [30]	Manual segmentation. FFT spectrum, 1/3 octave filters	fuzzy logic	Non-fluent stops: 95%
2005 Suszyński, Kuniszyk-Jóźkowiak [31]	Manual segmentation. FFT spectrum, 1/3 octave filters	correlation coefficients	Syllables repetitions: 70%

Table 1: Research summary of speech disorders automatic detection (cont.)

Year/Author	Features	Classifiers	Best results:
2006 Suszyński, Kuniszyk-Jóźkowiak [32]	Manual segmentation. FFT spectrum, 1/3 octave filters	correlation coefficients	Phoneme injection: 80%
2006 Szczurowska (Świetlicka), Kuniszyk-Jóźkowiak [16]	Manual segmentation. FFT spectrum, 1/3 octave filters, Kohonen network reducing	MLP (Multi-Layer Perceptron), RBF (Radial Basis Function) and Kohonen maps classifiers.	Non-fluent stops: 96%
2007 Wiśniewski, Kuniszyk-Jóźkowiak [25]	Manual segmentation. HMM based on MFCC (Mel-Frequency Cepstral Coefficients).		Repetitions: 80% Prolongations: 62%
2007 Wiśniewski, Kuniszyk-Jóźkowiak [26]	Manual segmentation. HMM based on MFCC.		Prolongations: 80%
2008 Świetlicka, Kuniszyk-Jóźkowiak [33]	Manual segmentation. FFT spectrum, 1/3 octave filters, Kohonen network reducion.	MLP, RBF and Kohonen maps classifiers.	Phonem repetition: 95%
2009 Świetlicka, Kuniszyk-Jóźkowiak [18]	Manual segmentation. FFT spectrum, 1/3 octave filters, Kohonen network reducion.	MLP, RBF and Kohonen maps classifiers.	Prolongations: 99%
2008 Ravikumar, Reddy [11]	Manual segmentation. MFCC.	MLP classifier.	Syllables repetitions: 83%
2009 Ravikumar, Rajagopal [12]	Manual segmentation. MFCC.	SVM (Support Vector Machine) classifier.	Syllables repetitions: 94%
2009 Sin Chee, Chia Ai [34]	Manual segmentation. MFCC.	kNN, LDA classifiers.	Avarage for Repetitions and Prolongations: 91%
2009 Sin Chee, Chia Ai [35]	Manual segmentation. LPCC.	kNN, LDA classifiers.	Avarage for Repetitions and Prolongations: 90%

Table 1: Research summary of speech disorders automatic detection (cont.)

Year/Author	Features	Classifiers	Best results:
2010 Kobus, Kuniszyk-Jóźkowiak [22]	Manual segmentation. LP (Linear Prediction) coefficients, Kohonen network reducing.	RBF and MLP classifiers.	Non-fluent stops: 76%
2010 Wiśniewski, Kuniszyk-Jóźkowiak [27]	Automatic segmentation. HMM based on MFCC.		Prolongations: 80%
2011 Wiśniewski, Kuniszyk-Jóźkowiak [28]	Manual segmentation. HMM used in HTK toolkit.		Repetitions: 89%
2011 Ravikumar, Ganesan [13]	Manual segmentation. MFCC.	Simplified SVM classifier.	Syllables repetitions: 85%
2012 Chia Ai, Hariharan [6]	Manual segmentation. MFCC and LPCC (Linear Predictive Cepstral Coefficients).	kNN (k-Nearest Neighbor) and LDA (Linear Discriminant Analysis) classifiers.	Avarage for Repetitions and Prolongations: 95%
2012 Codello, Kuniszyk-Jóźkowiak [36]	Automatic segmentation. CWT, Kohonen network reducing.	MLP classifiers.	Sound repetitions: 86%
2012 Codello, Kuniszyk-Jóźkowiak [21]	Automatic segmentation. CWT, Kohonen network reducing.	MLP classifiers.	Prolongations: 92%
2013 Codello, Kuniszyk-Jóźkowiak [2]	Automatic segmentation. CWT	correlation coefficients	Syllable repetitions (sensitivity and precision): 80%
2013 Hariharan, Fook [37]	Manual segmentation. DWT (Discrete Wavelet Transform) package, sample entropy.	kNN, LDA, SVM classifiers.	Avarage for Repetitions and Prolongations: 97%
2013 Kobus, Kuniszyk-Jóźkowiak [23]	Manual segmentation. LP coefficients, Kohonen network reducing.	RBF and MLP classifiers.	Prolongations: 75%
2013 Fook, Muthusamy [38]	Manual segmentation. WLPCC (weight linear predictive cepstral coefficients), MFCC, PLP (perceptral linear predictive) coefficients.	kNN, LDA, SVM classifiers.	Average for Repetitions and Prolongations: 95%

Table 1: Research summary of speech disorders automatic detection (cont.)

Year/Author	Features	Classifiers	Best results:
2015 Oue, Marxer [39]	MFCC, LPCC	DNN (Deep belief Neural Networks)	Repetitions (accuracy): 86%, Non-speech sound (accuracy): 75%
2015 Mahesha, Vinod [40]	MFCC	SVM, GMM (Gaussian Mixture Model) supervector	Prolongation and repetition (accuracy): 98%
2016 Kobus, Kuniszyk-Jóźkowiak [1]	Automatic segmentation. LP coefficients	k-means and thresholding	Syllable repetitions: 80%
2016 Mahesha, Vinod [41]	MFCC	GMM	Syllable repetition, word repetition, prolongation and interjection (accuracy): 96%
2019 Manjutha, Subashimi [42]	MFCC, PSO (Particle Swarm Optimization), SFO (Synergistic Fibroblast Optimization)	SVM, NB (Naive Bayes)	Stuttering (accuracy): 96%
2019 Fassetti, Fassetti [43]	Spectrograms, MFCC	Deep Learning	Disfluency (accuracy): 96%

2 Methodology

2.1 First attempt

2.1.1 Algorithm

The first attempt is shown in Figure 1.

2.1.2 Input data

Materials were recorded with the use of Creative Wave Studio on the SoundBlaster card with the frequency 22050Hz, 16 bps.

2.1.3 Preprocessing

Each recording was split into 512-sample frames without overlapping. For each frame there was performed multiplication by the Hanna window function.

2.1.4 Feature extraction

Time-frequency spectrum was extracted from each frame. For this purpose Linear Prediction Coefficients were obtained by the Levinson-Durbin [44] method. Coefficients were used to evaluate continuous frequency spectrum with 300 stripes accuracy. The order 15 was chosen for that purpose [45].

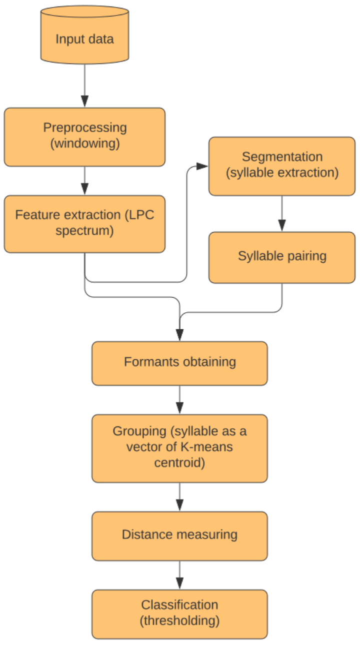

Input data
Preprocessing (windowing)
Segmentation (syllable extraction)
Feature extraction (LPC spectrum)
Syllable pairing
Formants obtaining
Grouping (syllable as a vector of K-means centroid)
Distance measuring
Classification (thresholding)

Figure 1: Algorithm of syllable repetition detection with the use of the LPC coefficients and the k-means

Table 2: Syllable counts in analysed utterances

są	im	ni	wy	zie	ci	wi	za	po	mu	do	wsze
1	2	3	2	2	1	1	3	15	3	2	1
dio	dzie	przy	mło	na	wa	be	ge	spo	bo	kie	pe
4	5	1	5	7	1	1	5	1	8	6	2
cia	ko	ub	mo	du	nie	ba	pom	sko	tro	łą	cha
2	10	2	5	2	2	1	1	1	3	3	3
Syllables: 117, Examined: 9 men (30 samp., 11–25yo), 5 women (26 samp., 13–24yo)											

2.1.5 Segmentation

Linear prediction coefficients were used in the method of splitting speech into syllables. Energy from obtained spectrum stripes was averaged for each frame. Two first vector values were averaged and this value increased by the 3 decibels was defined as the threshold for syllable start.

2.1.6 Pairing

Two consecutive syllables were defined as a pair to comparison. If the first syllable was shorter than the second one, the second syllable was shorten.

2.1.7 Formants extraction

The spectrum of each frame was used also to obtain speech formants [45–47]. Each formant was the local maximum with bandwidth condition fulfilled [45, 48] and is defined by three dimensions: time, amplitude and frequency. Each frame is represented by P formants.

2.1.8 Clusterisation

The frequency and the amplitude of each formant were used as an input of k-means clusterization. It reduced the time dimension and the number of points to the set of k centers. All of these dimensions were normalized by the variance of the point values.

2.1.9 Distance measuring

The normalization allowed to provide distance measuring. In the research the distance between vectors was defined as a sum of minimal Euclidean distances between the first and the second vector centers in the pair.

2.1.10 Classification

Two thirds of the 262 pairs of vectors were used as a training part. One third was a testing part. There were three types of categorization tested from which the minimal distance to the experimentally chosen threshold between groups of pairs classified as fluent and disfluent.

2.2 Results of the first attempt

The best results for the syllable repetition classification were reached for $6 \leq k \leq 7$. This could be caused as a result of insufficient representation of the speech signal when the number of formants is less than 5. Despite of the sufficient representation when $8 \leq k$ the overrepresentation could have an influence in the distance measuring, especially in case when the lower number of formants is already sufficient for speech signal representation.

2.3 Second attempt

2.3.1 Algorithm

The second attempt is shown in Figure 3.

2.3.2 Input data

In this research the speech data of 5 people was used. Each disorder was grabbed in the context of 4s length and all of these 106 utterances created speech recording $5m26s$ long.

2.3.3 Feature extraction

The spectrogram from Continuous Wavelet Transform was created for the entire sample. In the CWT algorithm Morlet wavelet was used as a Mother wavelet. It allows to define a center frequency F_c by changing cosine argument. As the scale the frequencies of Bark scale was used and the offset was defined as 50% of the wavelet's length.

The recording has been split to 512-sample non-overlapping frames. For each frame the spectrogram was obtained. Each scale has his own time offset and in that case there is a different number of CWT coefficients for each scale. These values were averaged and the vector of the single values for each scale in the bark scale was used for creation of the spectrogram.

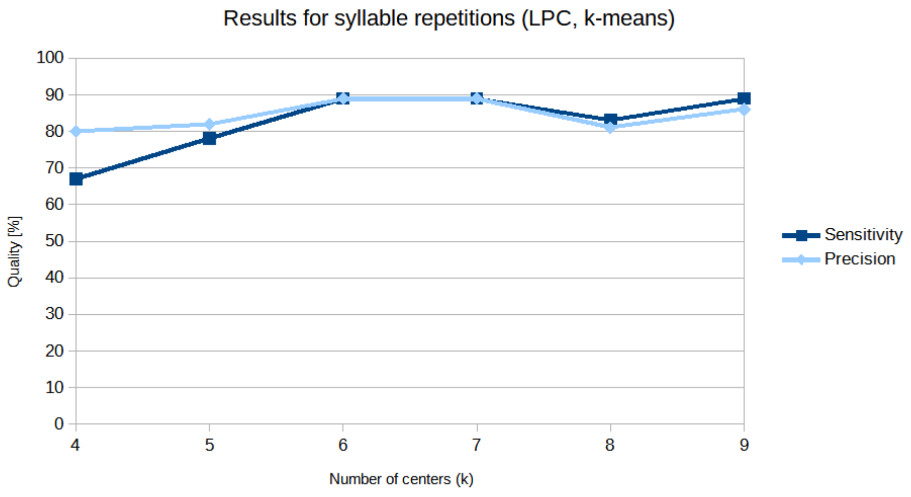


Figure 2: Sensitivity and precision percentage of classification in relation to the number k of centres

2.3.4 Segmentation

To define speech fragments from the whole utterance the thresholds between $-50dB$ and $-60dB$ were used. All parts where the speech signal has values larger than the threshold are defined as a fragments of speech.

2.3.5 Correlation obtaining

To define the similarity of the consecutive fragments the correlation of the vectors of CWT coefficients was evaluated. If there is a difference of the length between two consecutive fragments A and B, the fragment B is pruned to the length of A or if it's larger, it is extended. The correlation was defined by an equation 1.

$$C = \frac{\sum_{j=0}^{S-1}\sum_{i=0}^{N-1}(VA_{ij} - \overline{VA})(VB_{ij} - \overline{VB})}{\sqrt{\sum_{j=0}^{S-1}\sum_{i=0}^{N-1}(VA_{ij} - \overline{VA})^2 \sum_{j=0}^{S-1}\sum_{i=0}^{N-1}(VB_{ij} - \overline{VB})^2}} \tag{1}$$

where S is a number of the scales, N is the number of vectors of CWT coefficients for both fragments, VA and VB are vectors of CWT coefficients for two consecutive fragments A and B, $\overline{VA}$ and $\overline{VB}$ are means of all coefficients of all scales for the corresponding fragments.

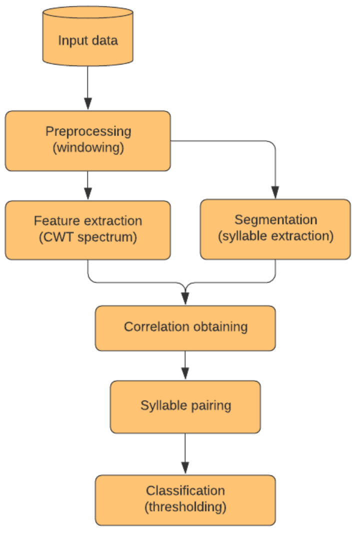

Input data
Preprocessing (windowing)
Feature extraction (CWT spectrum)
Segmentation (syllable extraction)
Correlation obtaining
Syllable pairing
Classification (thresholding)

Figure 3: Algorithm of syllable repetition detection with the use of the CWT coefficients and the correlation

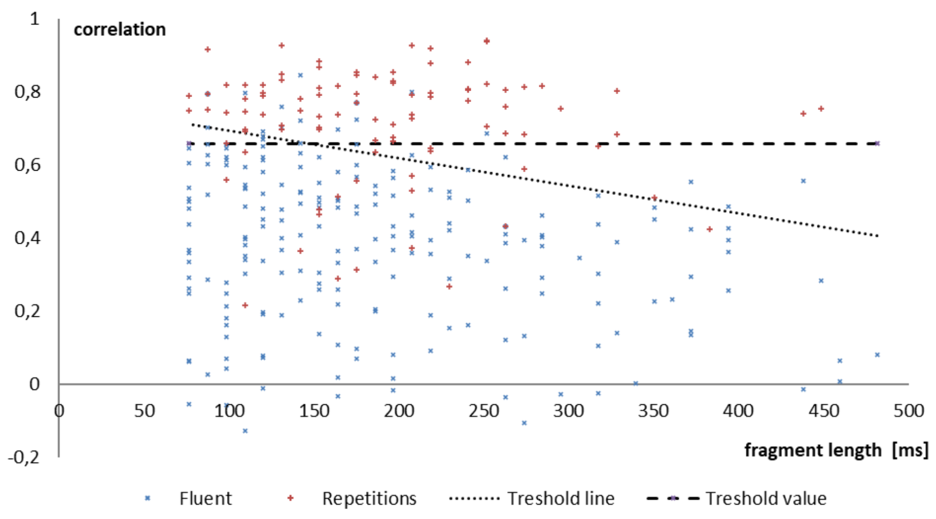


Figure 4: Correlation factor depending on the length of the recording

2.3.6 Syllables pairing

The consecutive fragments were limited by the time distance between adjacent fragments A and B. If the distance was larger than $50ms$ then such a pair was not considered. Also when the length of the fragment B was shorter than fragment A more than $100ms$, such a pair was also skipped as a bad candidate for syllable repetition.

2.3.7 Classification

Each remaining pair had obtained a correlation factor. There were tested threshold values despite of time length of fragments and threshold lines with time included between correlation values for pairs marked as repetitions and fluent pairs. For each threshold there were sensitivity and predictability evaluated to define the best threshold.

2.4 Results of the second attempt

The best results were achieved when first 5 bark scales were skipped. It also occurs that the results were better for three segmentation thresholds: $-55dB$, $-58dB$ and $-60dB$. Better results for them were achieved when the center frequency of the Morlet wavelet was higher. The threshold value was not discriminative for the results.

The best results for the threshold line were achieved also when the center frequency of the Morlet wavelet was higher.

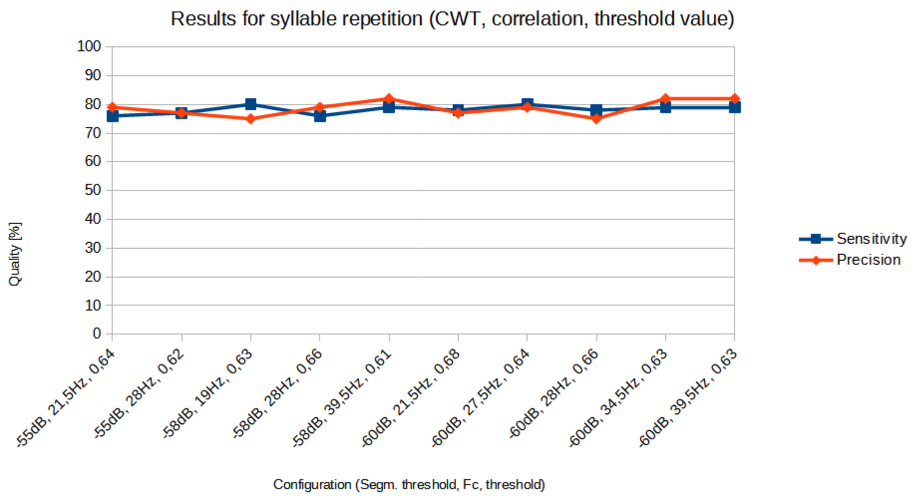


Figure 5: Sensitivity and precision results for the syllable repetition detection with the use of CWT and correlation depends on threshold value

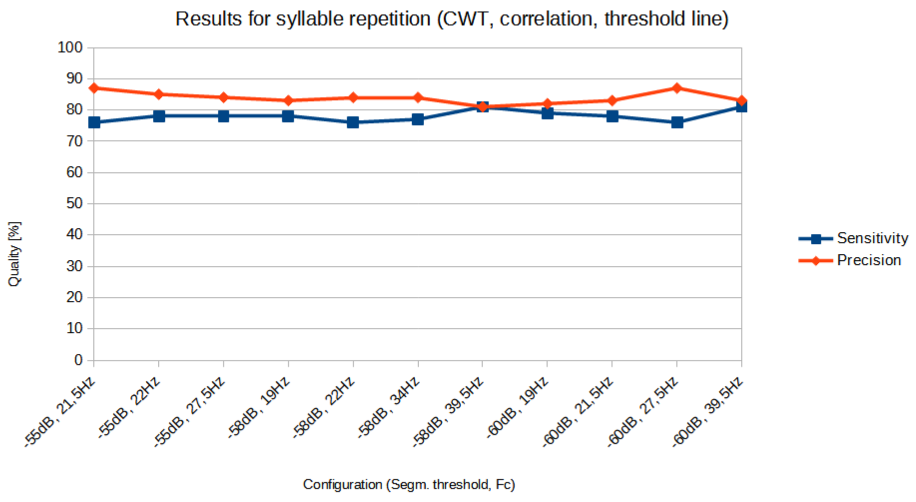


Figure 6: Sensitivity and precision results for the syllable repetition detection with the use of CWT and correlation depends on threshold line

3 Summary

The current works on the automatic detection of stuttering is based on the MFCC coefficients and with the use of SVM and Deep Learning classifiers. In general most of the attempts take as a features MFCC, LPCC, CWT coefficients and FFT spectrum. As the classifiers correlation coefficients, MLP, SVM, LDA and Deep Learning were used in most of the tries. Specifically the syllable repetitions are detected with the use of: FFT spectrum, MFCC, LPC spectrum and CWT and as the classifier: correlation coefficients, MLP, SVM, k-means thresholding and GMM.

All experiments prove that automatic syllable repetition recognition is possible and algorithms provided by the research could be practically used in software for stutterers. Detection quality gained in both of two algorithms quoted here reach 80% of precision and sensitivity. Also the other algorithms recalled in table 1 have quality between 70% and 94%.

References

[1] A. Kobus, W. Kuniszyk-Jóźkowiak, and I. Codello. Automatic syllable repetition detection in continuous speech based on linear prediction coefficients. *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, Advances in Intelligent Systems and Computing*, 403:295–304, 2016.

[2] I. Codello, W. Kuniszyk-Jóźkowiak, E. Smołka, and A. Kobus. Automatic disordered syllables repetition recognition in continuous speech using cwt and correlation. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013. Advances in Intelligent Systems and Computing*, 226:867–876, 2013.

[3] P. Howell and S. Sackin. Automatic recognition of repetitions and prolongations in stuttered speech. *In Proceedings of the first World Congress on fluency disorders*, pages 1–4, 1995.

[4] P. Howell, S. Sackin, and K. Glenn. Development of a two-stage procedure for the automatic recognition of dysfluencies in the speech of children who stutter: I. psychometric procedures appropriate for selection of training material for lexical dysfluency classifiers. *Journal of Speech, Language, and Hearing Research*, 40(5):1073–1084, 1997.

[5] P. Howell, S. Sackin, and K. Glenn. Development of a two-stage procedure for the automatic recognition of dysfluencies in the speech of children who stutter: Ii. ann recognition of repetitions and prolongations with supplied word segment markers. *Journal of Speech, Language, and Hearing Research*, 40(5):1085–1096, 1997.

[6] O. Chia Ai, M. Hariharan, S. Yaacob, and L. S. Chee. Classification of speech dysfluencies with mfcc and lpcc features. *Expert Systems with Applications*, 39:2157–2165, 2012.

[7] A. Czyżewski, A. Kaczmarek, and B. Kostek. Intelligent processing of stuttered speech. *Journal of Intelligent Information Systems*, 21(2):143–171, 2003.

[8] S. M. Hiroshima. A spectrographic analysis of speech disfluencies: characteristics of sound/syllable repetitions in stutterers and nonstutterers. *Proceedings 24th IALP Congress Amsterdam*, pages 712–714, 1999.

[9] Y. V. Geetha, K. Pratibha, R. Ashok, and S. K. Ravindra. Classification of childhood disfluencies using neural networks. *Journal of Fluency Disorders*, 25(2):99–117, 2000.

[10] W. Kuniszyk-Jóźkowiak, W. Suszyński, E. Smołka, and M. Dzieńkowski. Automatic recognition and measurement of durations of fricative prolongations in the speech of persons who stutter. (in polish). *Speech and Language Technology*, 8, 2004.

[11] K. Ravikumar, B. Reddy, R. Rajagopal, and H. Nagaraj. Automatic detection of syllable repetition in read speech for objective assessment of stuttered disfluencies. *In Proceedings of world academy science, engineering and technology*, pages 270–273, 2008.

[12] K. M. Ravikumar, R. Rajagopal, and H. C. Nagaraj. An approach for objective assessment of stuttered speech using mfcc features. *ICGST International Journal on Digital Signal Processing, DSP*, 9(1):19–24, 2009.

[13] K. M. Ravikumar and S. Ganesan. Comparison of multidimensional mfcc feature vectors for objective assessment of stuttered disfluencies. *Advanced Networking and Applications, Int. J.*, 02 (05):854–860, 2011.

[14] W. Suszyński, W. Kuniszyk-Jóźkowiak, E. Smołka, and M. Dzieńkowski. Automatic recognition of nasals prolongations in the speech of persons who stutter. *Structures - Waves - Human Health*, pages 175–184, 2003.

[15] W. Suszyński. Computer analysis and speech dyspluency recognition. doctoral dissertation. (in polish). *Politechnika Śląska, Gliwice*, 2005.

[16] I. Szczurowska, W. Kuniszyk-Jóźkowiak, and E. Smołka. The application of kohonen and multilayer perceptron networks in the speech nonfluency analysis. *Archives of Acoustics*, 31(4):205–210, 2006.

[17] I. Szczurowska, W. Kuniszyk-Jóźkowiak, and E. Smołka. Speech nonfluency detection using kohonen networks. *Neural Computing & Applications*, 18:667–687, 2009.

[18] I. Świetlicka, W. Kuniszyk-Jóźkowiak, and E. Smołka. Artificial neural networks in the disabled speech analysis. *Computer Recognition System 3, Springer Berlin/Heidelberg*, 57/2009:347–354, 2009.

[19] T. Tian-Swee, L. Helbin, A. K. Ariff, T. Chee-Ming, and S. H. Salleh. Application of malay speech technology in malay speech therapy assistance tools. *International Conference on Intelligent and Advanced Systems*, pages 330–334, 2007.

[20] I. Codello, W. Kuniszyk-Jóźkowiak, E. Smołka, and A. Kobus. Disordered sound repetition recognition in continuous speech using cwt and kohonen network. *Journal of Medical Informatics & Technologies*, 17:123–130, 2011.

[21] I. Codello, W. Kuniszyk-Jóźkowiak, E. Smołka, and A. Kobus. Automatic prolongation recognition in disordered speech using cwt and kohonen network. *Journal of Medical Informatics & Technologies*, 20:137–144, 2012.

[22] A. Kobus, W. Kuniszyk-Jóźkowiak, E. Smołka, and I. Codello. Speech nonfluency detection and classification based on linear prediction coefficients and neural networks. *Journal of Medical Informatics & Technologies*, 15:135–144, 2010.

[23] A. Kobus, W. Kuniszyk-Jóźkowiak, E. Smołka, I. Codello, and W. Suszyński. The prolongation-type speech non-fluency detection based on the linear prediction coefficients and the neural networks. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013, Advances in Intelligent Systems and Computing*, 226:887–897, 2013.

[24] W. Suszyński. Automatic detection of speech non-fluencies. (in polish). *50th Opened Acoustic Seminar*, 2003.

[25] M. Wiśniewski, W. Kuniszyk-Jóźkowiak, E. Smołka, and W. Suszyński. Automatic detection of disorders in a continuous speech with the hidden markov models approach. *Computer Recognition Systems 2, Springer Berlin/Heidelberg*, 45/2008:445–453, 2007.

[26] M. Wiśniewski, W. Kuniszyk-Jóźkowiak, E. Smołka, and W. Suszyński. Automatic detection of prolonged fricative phonemes with the hidden markov models approach. *Journal of Medical Informatics & Technologies*, 11:293—298, 2007.

[27] M. Wiśniewski, W. Kuniszyk-Jóźkowiak, E. Smołka, and W. Suszyński. Improved approach to automatic detection of speech disorders based on the hidden markov models approach. *Journal of Medical Informatics & Technologies*, 15:145–152, 2010.

[28] M. Wiśniewski and W. Kuniszyk-Jóźkowiak. Automatic detection and classification of phoneme repetitions using htk toolkit. *Journal of Medical Informatics & Technologies*, 17:141–148, 2011.

[29] Nöth E., Niemann H., Haderlein T., Decher M., Eysholdt U., Rosanowski F., and Wittenberg T. Automatic stuttering recognition using hidden markov models. *Proc. Int. Conf. on Spoken Language Processing*, pages 65–68, 2000.

[30] W. Suszyński, W. Kuniszyk-Jóźkowiak, E. Smołka, and M. Dzieńkowski. Automatic recognition of non-fluent stops. *Annales UMCS Inform.*, pages 183–189, 2004.

[31] W. Suszyński, W. Kuniszyk-Jóźkowiak, E. Smołka, and M. Dzieńkowski. Speech disfluency detection with the correlative method. *Annales UMCS Inform.*, AI 3:131–138, 2005.

[32] W. Suszyński, W. Kuniszyk-Jóźkowiak, E. Smołka, and M. Wiśniewski. Automatic detection. automatyczna detekcja wtrąceń. *Varia Inform., Algorytmy i programy, Polskie Towarzystwo Informatyczne, Instytut Informatyki Politechniki Lubelskiej*, pages 105–113, 2006.

[33] I. Świetlicka, W. Kuniszyk-Jóźkowiak, and E. Smołka. Detection of syllable repetition using two-stage artificial neural networks. *Polish J. of Environ. Stud.*, 17:462–466, 2008.

[34] L. Sin Chee, O. Chia Ai, M. Hariharan, and S. Yaacob. Mfcc based recognition of repetitions and prolongations in stuttered speech using k-nn and lda. *Proceedings of IEEE Student Conference on Research and Development*, pages 146–149, 2009.

[35] L. Sin Chee, O. Chia Ai, M. Hariharan, and S. Yaacob. Automatic detection of prolongations and repetitions using lpcc. *Proceedings of IEEE International Conference on Technical Postgraduates*, pages 1–4, 2009.

[36] I. Codello, W. Kuniszyk-Jóźkowiak, E. Smołka, and A. Kobus. Automatic disordered sound repetition recognition in continuous speech using cwt and kohonen network. *Annales UMCS Inform*, 2012.

[37] M. Hariharan, C.Y. Fook, R. Sindhu, A. H. Adoma, and S. Yaacob. Objective evaluation of speech dysfluencies using wavelet packet transform with sample entropy. *DSP*, 23:952–959, 2013.

[38] C. Y. Fook, H. Muthusamy, L. Sin Chee, S. B. Yaacob, and A. H. B. Adom. Comparison of speech parameterization techniques for the classifcation of speech disfluencies. *Turkish J. of Electrical Engineering & Computer Sciences*, 21:1983–1994, 2013.

[39] S. Oue, R. Marxer, and F. Rudzicz. Automatic dysfluency detection in dysarthric speech using deep belief networks. *SLPAT Interspeech 2015*, pages 60–64, 2015.

[40] P. Mahesha and D.S. Vinod. Support vector machine-based stuttering dysfluency classification using gmm supervectors. *Int. J. Grid and Utility Computing*, 6, Nos. 3/4:143–149, 2015.

[41] P. Mahesha and D.S. Vinod. Gaussian mixture model based classification of stuttering dysfluencies. *J. Intell. Syst.*, 25(3):387–399, 2016.

[42] M. Manjutha, P. Subashini, M. Krishnaveni, and V. Narmadha. An optimized cepstral feature selection method for dysfluencies classification using tamil speech dataset. *ISC2 2019*, pages 671–677, 2019.

[43] F. Fassetti, I. Fassetti, and Nisticò S. Learning and detecting stuttering disorders. *AIAI 2019*, pages 319–330, 2019.

[44] L. R. Rabiner and R. W. Schafer. Theory and applications of digital speech processing, ch. 9. *Pearson Higher Education, Inc*, 2011.

[45] R. C. Snell and F. Milinazzo. Formant location from lpc analysis data. *IEEE Transactions on Speech and Audio Processing*, 1, No. 2:129–134, 1993.

[46] B. Halberstam and L. J. Raphael. Vowel normalization: the role of fundamental frequency and upper formants. *Journal of Phonetics*, 32:423–434, 2004.

[47] T. Millhouse, F. Clermont, and P. Davis. Exploring the importance of formant bandwidths in the production of the singer's formant. *Proceedings of the 9th Australian International Conference on Speech Science & Technology*, pages 373–378, 2002.

[48] C. Kim, K. Seo, and W. Sung. A robust formant extraction algorithm combining spectral peak picking and root polishing. *EURASIP Journal on Applied Signal Processing*, 2006:1–16, 2006.

Exploring Recent Advancements of Transformer Based Architectures in Computer Vision

Michał Chromiak[*]

1 Introduction

The problem of *sequence transduction* [11] in tasks such as speech recognition, text-to-speech transformation, machine translation, protein secondary structure prediction, Turing machines etc. has wide scope of applications. That is why any progress in this area of research has a profound meaning. Emergence of the Transformer based architectures was motivated by the complexity of existing solutions like *Wavenet*, *Bytenet* $O(nlogn)$ or *ConvS2S* $O(n)$, but also their inability to learn distant position dependencies in sequences [16]. Moreover, the sequential nature of preceding solutions based on RNNs encoder-decoder schemes, have prohibited parallelization. While the parallelization was possible with the CNNs, it has another major drawback, which is the fact that the number of calculations in parallel computation of the hidden representation, for input-output position in sequence, grows with the distance between positions. Solving the *sequence-to-sequence*(seq2seq) problems with the mentioned drawbacks of pre-Transformer architectures was the motivation to invent a new approach.

Transformer has based its main idea purely on self-attention mechanism, that has reduced the number of computations relating to input/output symbols to only $O(1)$, allowing to model dependencies regardless of their position in sequence, while at the same time enabling the parallelization of computations. Transformer gains this advantage with *multi-head self-attention* (MHSA) that enables it to model dependencies regardless of their position in input, or output sentence.

1.1 Eliminating recurrence in machine translation

Transformer architecture is focused on the sequence-to-sequence model for Statistical Machine Translation (SMT) as introduced in [7]. It is based on an encoder-decoder scheme. The *encoder* is transforming input sequence into its latent representation, which is later decoded by the *decoder*, which in turn generates output in form of a new sequence.

[*]Corresponding author — `michal.chromiak@mail.umcs.pl`

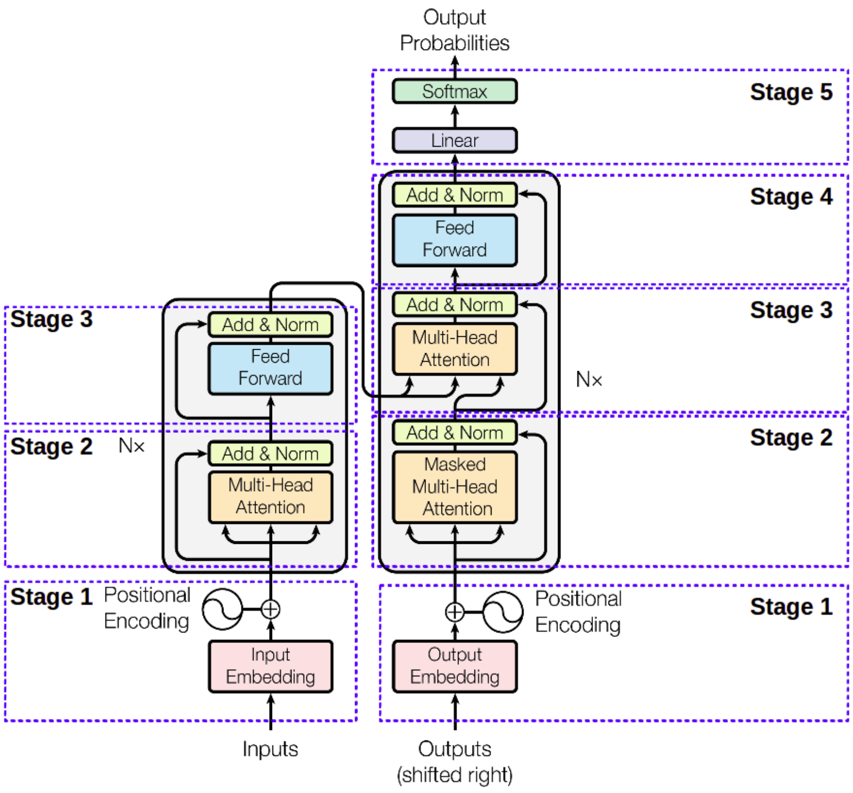


Figure 1: General transformer architecture

Encoder's output encodes entire source sentence in form of fixed size vector named: *sentence embedding* (for NLP tasks). Encoder includes six *multi-head attention* blocks followed by position-wise FNN. The *decoder*, similarly to ByteNet or ConvS2S, is stacked on top of the encoder.
Attention is general best described in [35]: "(...) as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key."[1].
Decoder is also composed of the same types of layers as encoder. Decoder's input is the output embedding of the encoder. Second layer is masked multihead-attention which is modified variant to prevent positions from attending to subsequent positions. Additionally, stages 2, 3 and 4 (see Figure 1) use residual connections following with normalization layer.

X-former models Due to the surge to improve parts of the vanilla Transformer, there has been multiple research that aimed at providing new features, and op-

[1]For details of the scaled dot-dot product and multi-head attention please refer to [35] due to limited scope of this paper.

timization changes[2]. Additionally, some of the far-reaching Transformer modifications were so deep that it even resulted with elimination of the decoder and using only the encoder to learn representation — e.g. in case of the *Bidirectional Encoder Representations from Transformers* — BERT [9] that uses masked language model.

1.2 Reducing the inductive bias with transformers

The attention-based approach was first applied in *Long Short-Term Memory* (LSTM) networks [5], but has also been used successfully in a variety of natural language processing tasks such as abstractive summarization, textual entailment, or sentence representation learning [26, 28, 24].

With results on-pair, or improving state of the art, there has been extensive research of Transformer based solutions for multiple modalities and tasks. The discoveries made, lead to interesting general conclusions. It is worth noting that, rather than being yet another architecture, Transformer is a general computation approach, and by this, it is more of a general counterpart of multilayer perceptron (MLP), than an architecture.

The specialized architectures based on LSTMs or CNNs are a very good solutions however, they tend to introduce *inductive priors*. They work surprisingly well as a built-in part of the architectures by incorporating processing knowledge that does not have to be learned by e.g. MLP — thus, saving valuable training resources.
MLP, as a form of a FFN, delivers an approximation engine that could technically learn any function. However the problem is that it is unstable in terms of small changes of data. The inductive priors such as CNNs help to refer local relations of pixels, or regions and to generalize — similarly to how human vision could be interpreted, allowing to infer based on the convolution based architecture. In NLP, LSTMs play a similar role of solid *inductive bias* for language models by incorporating prior words information (in the form of memory) into understanding the next word in a sentence.

The reason behind pre-wiring deep networks with generic, inductive innate knowledge structures, that were considered "appropriate" for specific tasks, was mostly due to insufficient amounts of data to learn a more general approaches. To help models learn, the inductive bias components — such as CNNs or LSTMs — were incorporated, enabling models to be useful, but at the same time making the models biased towards a certain architectural solution.

Currently the amounts of available data has significantly increased thus, the biases that initially were helpful, becomes a constraint for the neural network's function estimator, in terms of limiting its possibilities to learn better, or even achieve the perfect match for the target estimated function. Therefore, in a situation with enough data, the biased model will perform worse than the unbiased one. With such interpretation, the CNN/LSTM could be treated as a specialized MLPs, while the Transformer would be a more general version of MLP.

[2]The scope of this article is limited, but for a comprehensive Transformer model survey please refer to [32] to understand the taxonomy of modern landscape for solutions such as: Reformer, Linformer, Performer, Longformer, etc.

In this context, the Transformer is a more general version of compute framework than MLP, as it is not only containing the every-to-every connection from MLP, but also those connections are computed dynamically. With such design, Transformer becomes the most general approach that the current mainstream deep learning research is focused on. In general, with a higher number of data, the Transformer can learn to approximate functions that with smaller amounts of data had to be manually imposed within the architecture.

The effectiveness of Transformer applications are thus, more relying on its generic approach and amounts of training data, than any pre-designed elements — which is conversely to what takes place in case of the architectures based on inductive biases.

2 Replacing recurrence for object detection

One of the major advantages of Transformers is the parallel processing of a sequence. It has been originally applied to the NLP tasks, but it can also be adopted into the computer vision domain. Object detection is one of the major computer vision tasks. Its goal is to predict a set of correct category labels and image bounding boxes within the image. The *DEtection TRansformer* [3] is an architecture that actually is applying the parallelization approach from Transformer paradigm to image analysis. This allows to predict multiple objects in an image at once. Additionally, it greatly simplifies the pipeline flow for this task comparing to previous state-of-the-art (SOTA) solutions by eliminating the need for hand-designed[3], custom elements that contain prior knowledge.

Previous state-of-the-art results for object detection task were achievable mostly thanks to the highly optimized *Faster R-CNNs* [31]. Faster R-CNNs predicts object bounding boxes by filtering from large set of candidate regions. However, the recurrence element of the approach shares similar drawbacks as in case of the NLP tasks. Similarly to how vanilla Transformer has eliminated the recurrence from NLP domain, the DETR eliminates recurrent layers from object detection networks. The DETR's paper authors claim that it also is the first object detection framework to embrace Transformer as its main component.
With earlier object detection solutions, predictions were focused on predicting only one object from an image at a time, and repeating until all objects are accounted for. Predecessors have also addressed object detection with bounding boxes and classes in an indirect way and were based on multiple heuristics. Those include: hand-designed elements of non-maximum suppression (removing duplicate regions), or anchor generation — which actually required prior knowledge about the task. Additionally, the previous attempts for object detection were also based on surrogate regression and classification problems with large set of region proposals [30, 2] and solutions such as anchors [22] or window centers [42, 33].

The key idea behind DETR is to combine non-autoregressive, thus parallel decoding, with bipartite matching loss between ground truth and prediction, that is invariant by a permutation of predictions, due to being based on Hungarian [19] algorithm.

[3]Such as spatial anchors, or non-maximal suppression.

Transformer in this algorithm plays important role in terms of assuring that for large bounding boxes their long range dependencies are communicating.

2.1 Advantages of Transformer based object detection

The Faster R-CNNs have been leading the object detection task benchmarks for four years until the DETR has been introduced [3]. The DETR paper points four characteristics to compare to Faster R-CNNs:

- architecture simplicity (see Figure 2)
- improved performance for detection of larger objects (due to Transformer non-locality), however with lower performance for smaller objects
- improved precision (from 0.402 to 0.420)
- predicting objects in parallel (non-autoregressive decoding) instead of sequentially (autoregressive decoding with RNNs)

In the computer vision domain, the CNNs are very useful to prepare images for the neural network thus, DETR is using CNN to extract image features. The CNN scales down an image with three channels into a higher lever representation with multiple feature channels, while still preserving information about feature location in the image.
Such a representation is fed into the transformer encoder-decoder block. The output of Transformer component is a fixed number of N predictions in form of constant size set of bounding boxes and classes predictions, per image. Each of those predictions is a tuple of a *class* (set of the classes including '*no object*' — $\varnothing$) and a *bounding-box*.

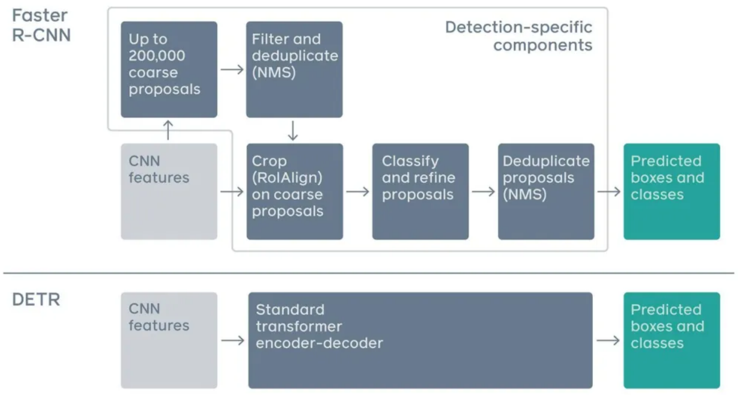

Faster R-CNN
Up to 200,000 coarse proposals
Filter and deduplicate (NMS)
Detection-specific components
CNN features
Crop (RoIAlign) on coarse proposals
Classify and refine proposals
Deduplicate proposals (NMS)
Predicted boxes and classes
DETR
CNN features
Standard transformer encoder-decoder
Predicted boxes and classes

Figure 2: DETR [3] simplifies architecture comparing to Faster R-CNN by leveraging a standard Transformer to perform (potentially non-differentiable) object detection operations

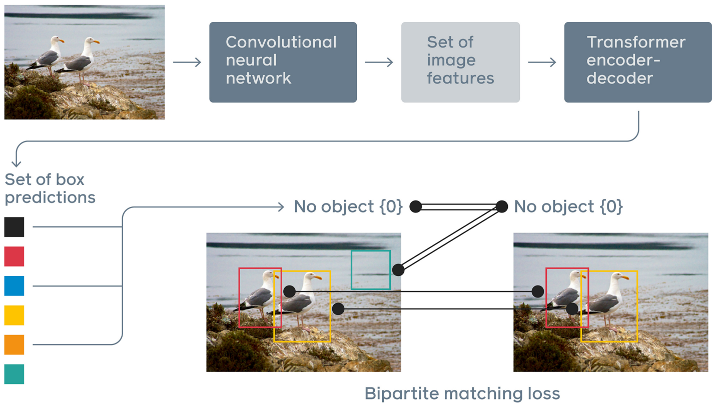


Figure 3: Combining CNN to detect features and Transformer encoder-decoder architecture to detect objects, enables DETR [3] to directly predict final set of detected object in parallel

The detection transformer algorithm uses a database of human-annotated bounding boxes that will be used as a target for a supervised learning task. Output of the algorithm will often result with bounding boxes that would overlap each other, or contain only a part of the actual object in the image. Additionally, the annotated images do not contain the *no-object* class label. The answer that DETER has to those issues, is to train end-to-end with a set-based global loss, enabling unique assignment between the ground-truth objects and the predictions (using *bipartite matching loss*). The final detection predictions are produced by FFN layers that process the object features into classes and bounding boxes while the unassigned predictions becomes the *no object*.

Compared to original Transformer [35], the difference is in the decoder phase. It is based on the fact that there are N objects decoded in parallel in each decoder layer, while in the vanilla Transformer, the model is autoregressive and predicts the output one by one in the form of a sequence. The decoder outputs N embeddings that are next independently decoded by FFN into the final (class, bounding-box) predicted tuples. One important parameter is N — which is the max number of object to detect from an image, and therefore is expected to be significantly larger than average number of objects in an image.

2.1.1 Bipartite matching loss

One of DETR's key characteristics is the ability to emit predictions in parallel. It is possible by dint of permutation invariant loss function that uniquely assigns a prediction to ground truth object.

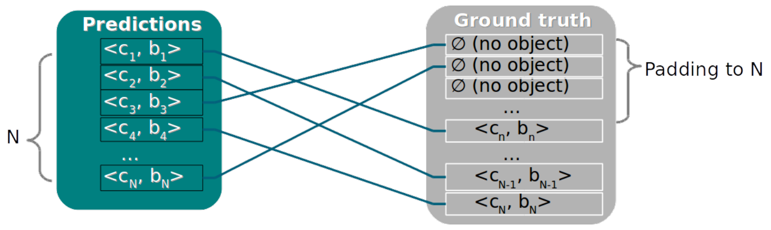


Figure 4: Bipartite matching loss

The scoring of N predictions[4] set $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ against the ground truth set[5] y is produced by the proposed loss. Predictions are assigned 1-to-1 to ground truth, such that the total loss is minimized.
To minimize such bipartite matching, one needs to search for a permutation of N elements: $\sigma \in \mathfrak{S}_N$[6] with the lowest pair-wise matching cost $\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$ computed efficiently with Hungarian algorithm. This cost considers similarity of class and the bounding box predictions to the ground truth.

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) \tag{1}$$

2.2 General architecture of detection transformer

The DETR's image processing backbone is based on CNN feature map extraction, using 1×1 convolution. As an outcome it produces the higher level image representation in low resolution, scaled down, and with many more feature channels. This transformation however, preserves the feature location information. As transformer is expecting a sequential input, the input for encoder is first collapsed into one dimension vector. Because transformers are permutation invariant, the flattened input vector is supplemented with a positional encoding. In the next step, the decoder attends to encoder output with learned positional embeddings (*object queries*). The decoder's output embedding is fed to a shared FFN that predicts a detection (class, bounding box) or a *no object*. Finally, FNN predicts the normalized center coordinates and size: height and width of the box with respect to the input image, while the linear layer predicts the class label using a softmax function.

The set of N learned *object queries* enables inference about relations between objects and their relations within an entire image context. Decoder takes the output embedding produced by encoder and N *object queries*, as an input. The decoder's output is the result of transformation made to object queries conditioned on the encoder outputs (see Figure 6).

[4]The *bounding box* from the prediction tuples is expressed here as a *position (x,y)* and *size (height,width)*.
[5]Padded to the size of N with ∅ (no object) — see Figure 4
[6]$\mathfrak{S}_N$ denotes a group of all permutations of N-element set

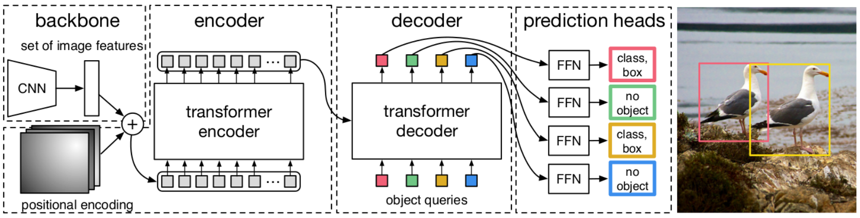


Figure 5: DETR [3]: The CNN representation of image supplemented with positional encodings is fed into the Transformer encoder

The output embeddings of decoder — produced from object queries — goes independently to FFN classifier, that decodes them into N (class, bounding box) predictions (see Figure 5). The *object queries*, in form of learned positional encodings, are required as the decoder is permutation-invariant and thus, for different results requires different input embeddings. The object queries might be perceived as specialized detectors for specific size of bounding boxes, within specific areas of the image.

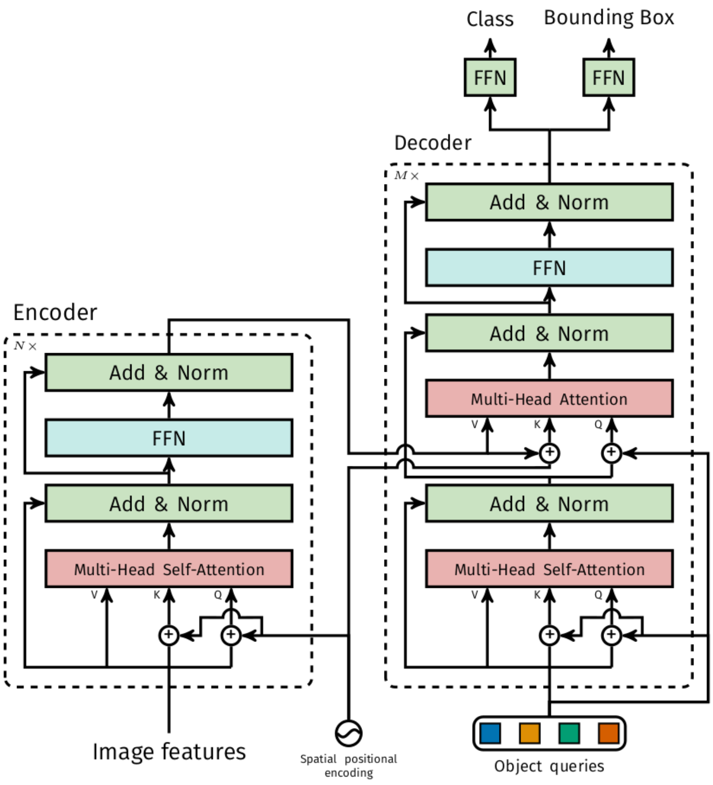


Figure 6: The *object queries* in DETR are transformed based on the image information coming from the encoder's output

Specialization of object queries is progressing upwards, with higher layers of decoder. The object queries (in the form of vectors) are trained to ask different questions depending on the size and location of the bounding box. It is exactly what attention mechanism is after. It will attend more to specific parts of the encoder embeddings — ie. the side input of decoder — getting more information as it sends requests in form of *Query* vectors (as of the attention mechanism) to the encoder output, and receiving the *Value* responses from parts that correspond to the features requested by the sent query — Q (see Figure 6).

Thanks to different object queries, the decoder output is going to be focused around different classes and different bounding boxes in different regions. The important insight is that — as this is transformer — the vectors not only incorporate information form the image, but also can attend (communicate) with each other, in all of the layers. This is useful to determine that each object query is focused on unique region, class and bounding box size. Such architecture impose results to be more, and more precise along the higher layers of decoder.

2.2.1 Performance comparison and potential improvements

DETR has been compared with Faster R-CNN in quantitative evaluation on COCO dataset [23] (see Table 1). While DETR has proven to be less efficient for detecting smaller objects comparing to Faster R-CNN, it should be noted, that the Faster R-CNN has been equipped with multiple design improvements since the baseline publication. DETR is a relatively new model and there is still space for potential improvements, similarly to how Feature Pyramid Networks (FPN) [21] did for Faster R-CNN.

Table 1: For COCO validation ser the comparison of DETR with Faster R-CNN with ResNet-50 and ResNet-101 backbone [3]

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

3 Replacing CNNs in computer vision

While the Transformers in natural language processing (NLP) has become a leading standard, in vision — the dominant solution are based on convolution architectures [20, 18, 12]. Even the recent Transformer based solutions such as DETR [3] include CNN components in their end-to-end pipeline.

The difficulty with replacing convolutions by transformers is computation complexity of attention layers. For sequence processing, transformers takes a set of tokens as an input and process it with quadratic operation in form of attention layers[7]. It is one of the main limitations of transformers in terms of memory and computing requirements. For this reason, raster images that are to be processed by algorithm, even for a small resolution of a couple hundred pixels, means that for an image of $n \times n$ there will be $(n^2)^2$ operations which do not scale well for realistic input image sizes.

The hybrid approach is to combine the convnets with forms of adaptation of the self-attention. This is the approach already mentioned in section 2, but also in [37, 14, 39] where the CNN output is being processed with self-attention for object detection task. Other solutions are based also on augmenting feature maps for image classification [1] or even unsupervised image detection with so called *slot attention* [25].

A more radical approach[8] [27, 29] was to eliminate convolutions at all, and focused on restricting the global self-attention mechanism to attend local neighborhoods for each query pixel. This arrangement resembles convolutions with learning sliding kernels that are local thus, the receptive field is growing by depth[9] across the layers. A local multi-head, self-attention blocks approach has been manifested also in: [15] — defining layer weights based on composability of local pixel pairs similarity, [29] — by swapping spacial convolution layer with self-attention layer in ResNet model, or in [41] — replacing convolution with *pairwise*[10], and *patchwise*[11] self-attention.

Another approach is to use scalable approximation on global self-attention. It is based on sparse factorizations of attention matrix into several, faster attention operations that are later combined to approximate dense attention operation [6]. Finally there has been also research aimed at decomposing attention into blocks of varying sizes [38], or along axis with models employing local region attention stacking[12] in the form of *axial attention* [36, 13].

The previous approaches of eliminating convolutions ([29, 36]) have presented promising experiments however, as claimed in [10], due to use of specialized attention patterns those other solutions require complex engineering to be implemented efficiently on hardware accelerators thus, have not yet been scaled to use their full potential.

On top of the previous, above-mentioned research, a new (as of 2020) approach

[7]This is due to the fact that attention requires pair-wise inner product between each pair of input tokens.

[8]From some of the authors of original the Transformer paper.

[9]In contrast, transformers are able to attend within a single layer to every input token.

[10]Which is a set operator rather than sequence operator — as the convolution. Unlike convolution it does not couple stationary weights to specific locations and is invariant to permutation and cardinality

[11]Which is not permutation-invariant, cardinality-invariant nor a set operation however, the weight computation can index feature vectors individually by location, while associating information from different locations.

[12]The locality constraint limits model's receptive field which is an issue especially for tasks such as high resolution segmentation tasks.

has been developed. This time the difference is how the input image is preprocessed. Every image is dissected into a set of non-overlapping patches. Next, a full self-attention is applied on top. It is based on research [8] which shows that the original Transformer is competitive to state-of-the-art convnets, by proving that the multi-head self-attention (MHSA) layer can represent a convolutional filter.
The idea has been developed further in a very simple solution of fully Transformer based image recognition model [10]. The patch size has been increased from 2×2 (applicable to small resolution images) to 16×16 pixels — enabling use of medium size images, and tested on large scale experiments. Experiments presented in the paper show that, for small/mid-sized images, benchmarks produce comparable results to CNN based networks with less computational resources used for training.

3.1 Vision Transformer architecture

The transformers are able to attend with a single layer to the entire area of interest. In case of vision this area is an image. ViT use global attention by decomposing image into patches of 16×16 pixels. As all layers of Transformer use constant latent vector, the flattened patches are mapped to fixed size with a trainable linear projection by multiplying with an embedding matrix. The result is referred to as *patch embedding*. As transformers are also permutation invariant, the unrolled set of patches is combined with position, learnable embedding vectors, that are added to the patch embeddings to encode the position information. Such output becomes an input to vanilla Transformer encoder (see Figure 7). Similarly to BERT [9] model, a dedicated classification learnable embedding token[13] `[class]` is prepended to the encoder input and the final image classification is based on its output in last layer.

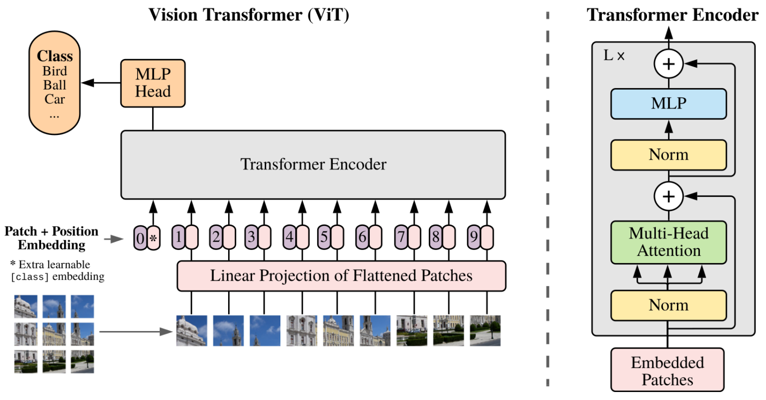


Figure 7: Vision Transformer general architecture [10]

[13]Not associated with any patch.

3.2 Advantages of using only Transformer for vision

The *Vision Transformer* (ViT) [10] is one of the most recent advances in the area of fully Transformer based approach for vision tasks. It is a manifestation of the inductive bias reduction (as discussed in section 1.2 by replacing the CNN part.

As the experiments show, the amounts of data that has been used (14M-300M images)[14] was sufficient, for Transformer to learn the knowledge representing CNN-alike structures, that has been manually incorporated in the previous solutions (inductive bias — see 1.2). In contrast to CNN[15], ViT contain the locality and translation equivariance only with MLP layers, while the self-attention layers are global.
ViT also does not inject all information about the 2D nature of an input image, such as position and spacial relations between patches. The only part that explicitly uses the concept of 2D nature is splitting the image into patches and adjusting positional embeddings for images of different resolutions.

Despite the fact that ViT do not introduce image-specific inductive biases into its design, it is able to learn similar characteristics as CNN based architectures. This can be easily seen when referring to the learned filters of the initial linear embeddings of RGB values (see Figure 8 — left). We can see some kind of — *learned* — filters that are very similar to the filters[16] that in the past were manually built into the architectures like convnets.

Additionally, in contrast to CNNs, some of the ViT's heads are already attending to large distances from the pixel even for small network depth (see Figure 8 — right), instead of retaining constant distance as in CNNs. This way, the Transformer can already pay attention to long distance relations in the initial layers. With increase of the network's depth, the attention/receptive distance grows for all heads becoming almost global. With convnet (or local attention), this would have been more of a constant increase in size of the effective receptive field.

Interestingly, recent research [34] also shows that vision transformers similarly to humans are more biased towards shapes than convnets.

3.3 ViT performance comparison

The competing solutions were based on supervised transfer learning with large ResNets — *Big Transfer* (BiT) [17], and the *EfficientNet* defined in a semi-supervised way with removed labels [40] — *Noisy Student*. In both comparisons the ViT used substantially less computation resources for pre-training and outperformed, or was comparable to the remaining solutions.

3.4 Self-supervision in vision transformer

One of the last interesting aspect (however not fully investigated) of full Transformer approach to vision with ViT is use self-supervised approach similarly to

[14]Pre-trained ImageNet-21k and JFT-300M datasets.
[15]The locality, 2D neighborhood structure plus the translation equivalence are present in each layer of the architecture.
[16]Like the filter at position (1,3) Figure8 — which looks much like a wavelet filter.

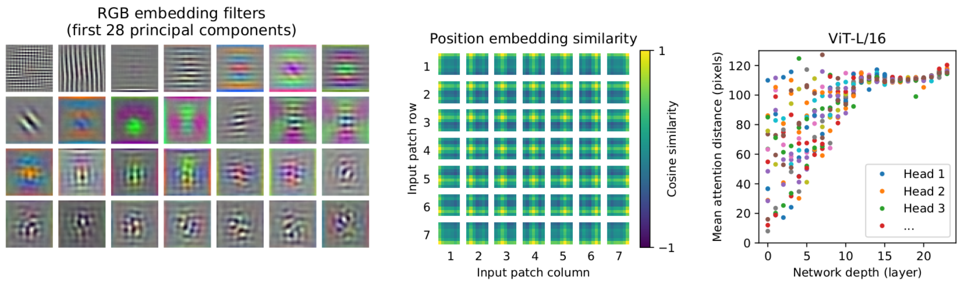


Figure 8: **Left:** Learned linear embedding filters **Center**: Learned positional embeddings of patches and their similarity across row and column. **Right**: Relation between size of attended area by head and network depth — mean attention distance. Source: [10]

what have been done with BERT's [9] masked approach, or GPT/iGPT [4] language modeling. The *masked patch prediction* has been proven on prediction task for 3bit, mean color (512 colors in total) of every corrupted patch with 79,9% accuracy on ImageNet which is 4% worse than supervised pre-training.

Table 2: ViT [10] state-of-the-art comparison benchmarks

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

4 Summary

The general nature of Transformer based architectures has proven to be currently state of the art solution for solving problems across multiple domains of natural language processing, vision or speech recognition. All of the preceding, hand designed inductive biases, such as the long short-term memory or convolutional neural networks and recurrence, that has been compared against Transformer counterparts are matched, or exceeded, in terms of either performance or accuracy. However, immense amounts of data that is currently available for the training phase, favors the learning approach over imposed design. The experiments show that the approaches based on biases learned from scratch, are not only similar, but more exact for the evaluated tasks, than a hand crafted, built-in ones.

Transformer as a computing framework has multiple flavors especially in computer vision, but there is still space to propose even more generic design.

References

[1] Irwan Bello, Barret Zoph, Quoc Le, Ashish Vaswani, and Jonathon Shlens. Attention augmented convolutional networks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3285–3294, 2019.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: high quality object detection and instance segmentation. *CoRR*, abs/1906.09756, 2019.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[4] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020.

[5] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *CoRR*, abs/1601.06733, 2016.

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019.

[7] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[8] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[11] Alex Graves. Sequence transduction with recurrent neural networks, 2012.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *CoRR*, abs/1912.12180, 2019.

[14] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[15] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. *CoRR*, abs/1904.11491, 2019.

[16] John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. 2001.

[17] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Large scale learning of general visual representations for transfer. *CoRR*, abs/1912.11370, 2019.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[19] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*. Springer International Publishing, 2014.

[24] Zhouhan Lin, Minwei Feng, C'ıcero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017.

[25] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *CoRR*, abs/2006.15055, 2020.

[26] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933, 2016.

[27] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR, 10–15 Jul 2018.

[28] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July 2006. Association for Computational Linguistics.

[29] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[31] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[32] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *CoRR*, abs/2009.06732, 2020.

[33] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[34] Shikhar Tuli, Ishita Dasgupta, Erin Grant, and Thomas L. Griffiths. Are convolutional neural networks or transformers more like human vision?, 2021.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[36] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan L. Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. *CoRR*, abs/2003.07853, 2020.

[37] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[38] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *International Conference on Learning Representations*, 2020.

[39] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *CoRR*, abs/2006.03677, 2020.

[40] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2020.

[41] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. *CoRR*, abs/2004.13621, 2020.

[42] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points, 2019.

Automating the Comparison of Areas and Data of Administrative Units From Different Periods on the Example of Poland (1937 and 2019)

Filip Berezowski
Jarosław Bylina[*]

1 Introduction

This manuscript is devoted to the issue of automatic comparison of areas and data of administrative units from different periods. State administrative boundaries change over the years. As a result, statistical information cannot be compared 100% in many cases. For this reason, methods are needed that can do this with the most accurate result possible.

We can divide the essence of this problem into two issues. The first is the geometry of the administrative units that we compare with each other. We must be able to automatically determine whether certain areas intersect with each other, and if so, to what extent. This is very important because it affects the accuracy of the statement. The second is the methodology — how to compare the same statistical data once we can determine the geometric similarity of administrative units. For example, the administrative boundaries from 1937 and the current boundaries from 2019 will be used to compare the program. We will use the Python language to implement the program with the use of relatively easy-to-use libraries that can read, analyze and write spatial data. The compatible Shapely [1] and Fiona libraries are ideal for this purpose.

The foundations of the contemporary Polish territorial division, made after World War I, are related to the post-partition era [2]. The partitioning states (Russia, Austria and Prussia) introduced their own administrative and territorial institutions in place of the Polish territorial division. In the second half of the 19th century, changes in the administrative division in the former Polish-Lithuanian

[*]Corresponding author — `jaroslaw.bylina@umcs.pl`

Commonwealth deepened. The reborn Polish State after its partitioning powers received an extremely diversified territorial and administrative shape of the lands.

For us, the act on changes to the borders of the Pomeranian, Poznań, Warsaw and Łódź voivodeships of June 12, 1937, is particularly important [3]. This act is an example of legal regulations aimed at blurring the borders of the partition. The administrative units included in this act are of key importance for identifying the compliance of maps with the assumptions of the act (Figure 1). We will use this boundary data later in the work to compare these time periods.

The territorial division currently existing in Poland was introduced according to the Act of August 24, 1998 on the introduction of the basic three-tier territorial division of the state [4].

Another important legal act regulating the territorial division of the state at the county level was the ordinance of the Council of Ministers of August 7, 1998 on the establishment of counties [5]. At that time, a total of 308 counties were created in Poland, including 65 cities with county status (the so-called urban counties). In many cases, this rank was given to cities which, as a result of the reform of the administrative division, lost the rights of voivoideship cities and the related privileges.

Until now, the number of voivoideships in Poland has not changed. The state of affairs as of January 1, 2019 is as follows: 16 voivoideships, 2478 communes (also included 66 cities with county rights), 314 counties, 66 cities with counties rights, 930 cities and 52,497 rural towns [6].

2 Methodology of the algorithm

2.1 Recreating the geometry in the program

The administrative units used by the program are polygons, but this classification is too narrow to fully recreate the geometry for operations in Shapely classes. Each object must be assigned to the category: polygon without rings, a polygon with rings, multipolygon without rings, multipolygon with rings (Figure 3).

The coordinates of the points are written in two-element tuples. The polygon is saved as a list of tuples with coordinates inside the next list. If the first-order list is single-element, then it is a polygon without rings, otherwise, it is a polygon with rings. A multipolygon is an object composed of many polygons. There is an additional list in its notation. Now the number of elements in the first-order list corresponds to the number of polygons that make up the multi-polygon. The number of elements in the second-order list shows whether the polygon included in the multipolygon has rings. Items in the third-order list are the point coordinates for the corresponding parts. The program includes a function that is responsible for taking a polygon from a vector layer and converting it into a Shapely library class.

```
def get_polygon_from_feature_class(fc_pol):
    geom_type = fc_pol['geometry']['type']
    geom_coor = fc_pol['geometry']['coordinates']
    geom_no_part = len(geom_coor)
```

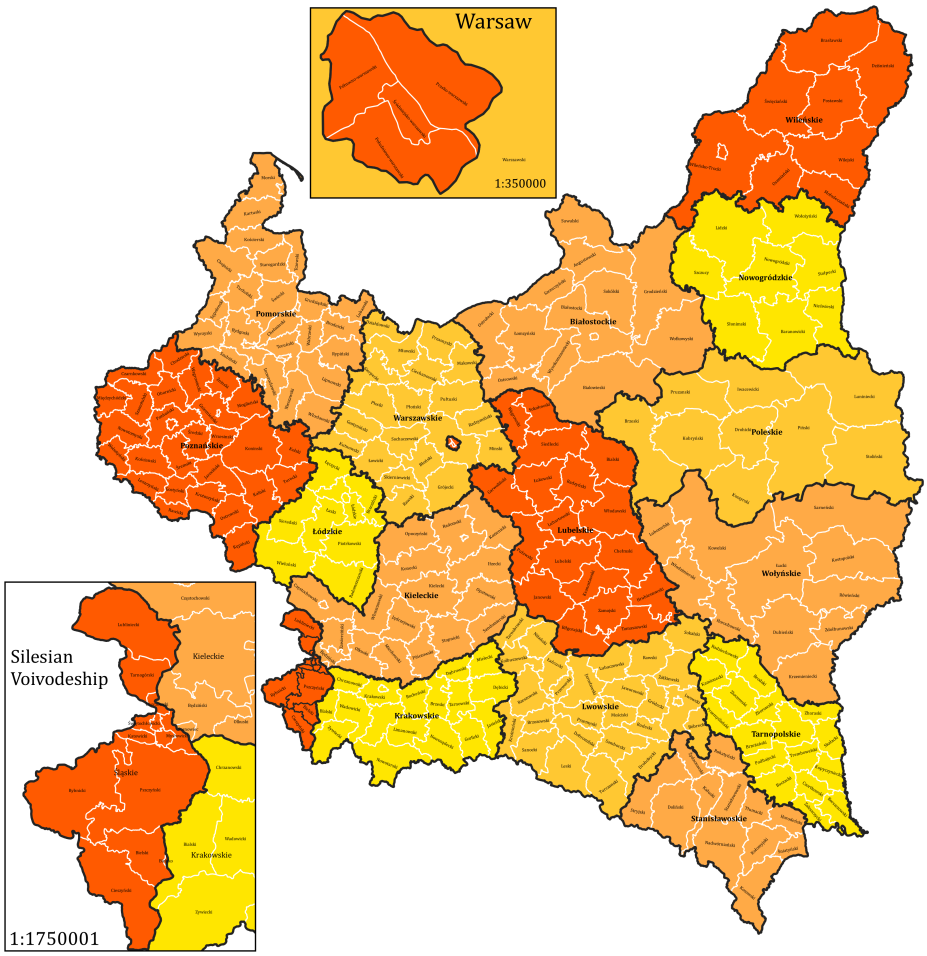


Figure 1: Administrative division of the Republic of Poland from 1937 (own study)

```
if geom_type == 'Polygon':
    #Polygon without rings
    if geom_no_part == 1:
        shapely_pol =  Polygon(geom_coor[0])
    #Polygon with rings
    else:
        shapely_pol =  Polygon(geom_coor[0], geom_coor[1:])

elif geom_type == 'MultiPolygon':
    pol_list = []
    for i in range(geom_no_part):
        geom_no_ring = len(geom_coor[i])
```

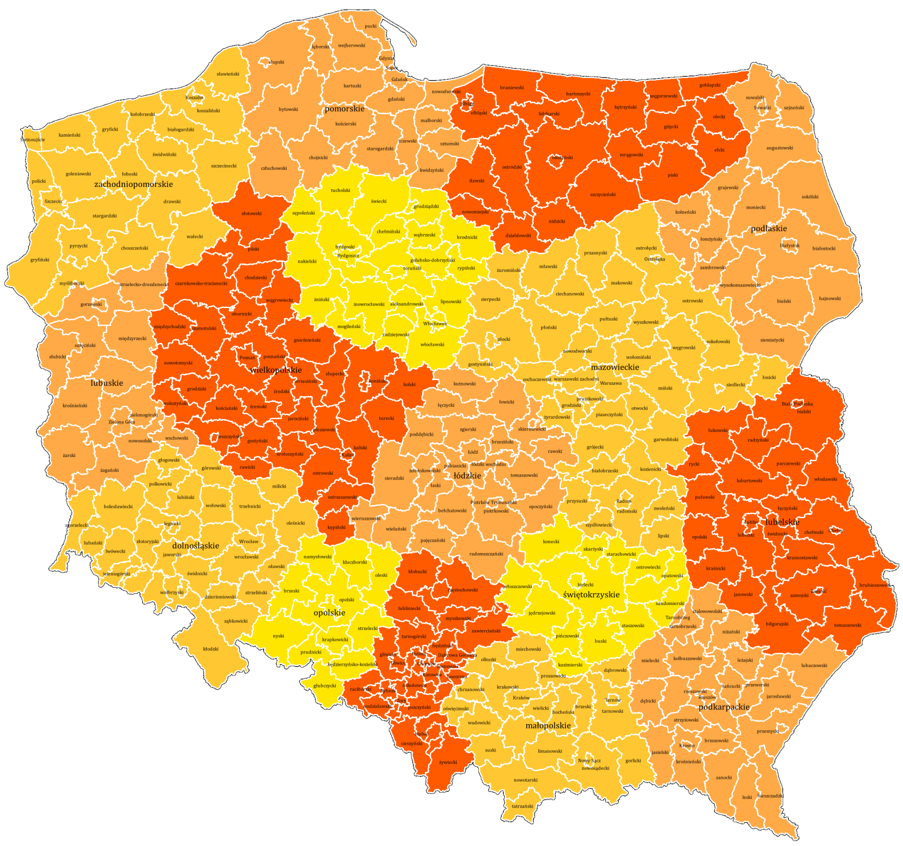


Figure 2: Administrative division of Poland based on data from the state register of borders and areas of territorial division units of the country — PRG (as of 28 April 2020; own study)

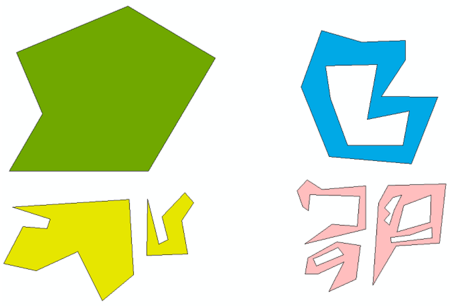


Figure 3: Polygon geometry types: polygon without rings (green), polygon with rings (blue), multipolygon without rings (yellow), multipolygon with rings (pink)

```
            #Multipolygon without rings
            if geom_no_ring == 1:
                shapely_multi_pol =  Polygon(geom_coor[i][0])
            #Multipolygon with rings
            else:
                shapely_multi_pol =  Polygon(geom_coor[i][0],
                                             geom_coor[i][1:])
            pol_list.append(shapely_multi_pol)
        shapely_pol = MultiPolygon(pol_list)
    return shapely_pol
```

2.2 Comparing the geometry of administrative units

An important part of the program is the compilation of the geometry of administrative units in different variants. For each of them, data preparation will be different. All options must also return some values. These values are indicators that show the ratio of the common area to the area for a unit or administrative units from one period, as well as the geometry of the common area. There are four variants in the program:

- 1:1 — comparison of one 'old' county with one 'new' county
- 1:2 — comparison of one 'old' county with two 'new' counties
- 2:1 — comparison of two 'old' counties with one 'new' county
- 2:2 — comparison of two 'old' counties with two 'new' counties

The comparison for the 1:1 variant is the easiest to implement. We do not have to interfere with the geometry of objects within one layer. Instead, we can proceed with the statement immediately. We will use the `comparasion_one_to_one` function for this. In the first step, this function checks if the polygons intersect with each other. If not, the function returns a tuple (0, 0, 0). Otherwise, a polygon is created that is a common part of the polygons. Then the indicators are calculated according to the formulas:

$$ind_{otn} = \frac{area_c}{area_o} \tag{1}$$

$$ind_{nto} = \frac{area_c}{area_n} \tag{2}$$

where:

- ind_{otn} — ratio of the common area to the area of the 'old' county
- ind_{nto} — ratio of the common area to the area of the 'new' county
- pow_c — the common area of 'old' and 'new' counties
- pow_n — the area of the 'new' county

- pow_o — the area of the 'old' county

The formulas and the function are so universal that after some data preparation, they are used by all created variants. The described function looks like this:

```
def comparasion_one_to_one(pol1, pol2):
    if pol1.intersects(pol2):
        pol_inter = pol1.intersection(pol2)
        old_in_present_per = pol_inter.area/pol2.area*100
        present_in_old_per = pol_inter.area/pol1.area*100
        return old_in_present_per, present_in_old_per, pol_inter
    else:
        return 0, 0, 0
```

Option 2:2 is the most difficult to implement. First, we need to find pairs of units that are adjacent to each other. Only connections that meet this condition will be considered for comparison. Before that, however, the polygons are merged into a single feature. In the program, the function `create_duo_polygon_list` is responsible for this problem. When the criterion is met, such a polygon is stored inside the list with additional parameters such as the coordinates of the enclosing frame needed in the next sorting process of the list, an ID for both polygons from the pair needed for later writing information in the list. The function is also used in variants 1:2 and 2:1. It is then performed only once for a specific layer. The described function looks like this:

```
def create_duo_polygon_list(list_pol):
    list_pol_duo = []
    for i, shapely_pol_1_1 in enumerate(list_pol):
        for j, shapely_pol_1_2 in enumerate(list_pol):
            if j < i:
                if shapely_pol_1_1[1].intersects(
                   shapely_pol_1_2[1]):
                    pol_duo = shapely_pol_1_1[1].union(
                              shapely_pol_1_2[1])
                    list_pol_duo.append(
                        [pol_duo.bounds, pol_duo, i, j])
    return list_pol_duo
```

This way the question is for comparison, but to speed up the process there is mergesort. Sorting using this method consists of dividing the list into n list of one-elements[1]. Successive lists are combined into larger sorted lists until one sorted n-element list is obtained. The described function looks like this:

```
def mergesort(list_1):
    if len(list_1)>1:
        mid = len(list_1)//2
```

[1]For us, this means a list with four elements, where the sorting element is the minimum longitude value of the frame surrounding the connected polygons.

```
    list_left = list_1[:mid]
    list_right = list_1[mid:]
    mergesort(list_left)
    mergesort(list_right)
    i = 0
    j = 0
    k = 0
    while i<len(list_left) and j<len(list_right):
        if list_left[i][0][0]<list_right[j][0][0]:
            list_1[k] = list_left[i]
            i += 1
            k += 1
        else:
            list_1[k] = list_right[j]
            j += 1
            k += 1
    while i<len(list_left):
        list_1[k] = list_left[i]
        i += 1
        k += 1
    while j<len(list_right):
        list_1[k] = list_right[j]
        j += 1
        k += 1
```

2.3 Thresholds for further calculations

At this stage, we managed to compare the geometry of the administrative units in four different variants. However, not every intersection of two polygons will allow us to compile statistics for individual counties. It should be verified through thresholds which polygons can be used for further analysis.

First of all, it is worth noting that among the compared objects there are not those that are identical. This is due to errors made during data vectorization or errors resulting from a certain simplification of the data compared to the actual boundaries. Due to the circumstances, it is worth considering the percentage threshold of layer interpenetration, after which it should be considered that these units are identical. Too low a threshold reduces the credibility of the results, while too high a threshold increases the number of operations that the program must perform. A consistency of more than 95% in both ind_{swn} and ind_{nws} indicators happens very rarely, so it was chosen as the threshold in this case. Then the border deviations are very small (Figure 4).

On the other hand, it is worth considering at what point we can speak of a comparison of, for example, two counties. Returning the results when there is overlapping in 10% of the area can cause significant errors in the calculation of areas that have changed a lot in the periods checked. For different input data, the threshold will be variable, because we will be balancing between the coverage of the largest possible area and the errors resulting from the small influence of

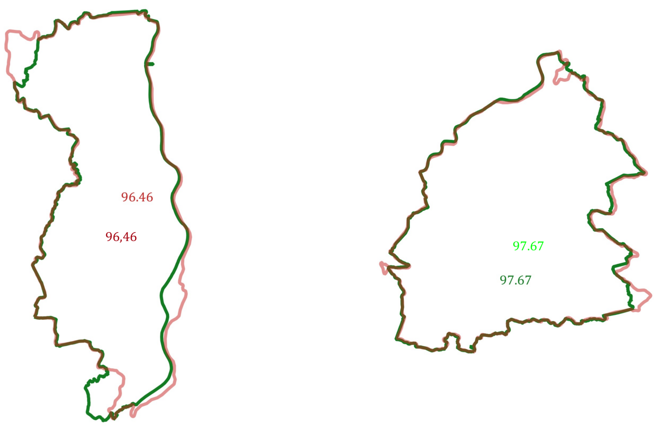


Figure 4: Comparison of two pairs of counties: Tczewski (left) and Mielecki (right). The borders from 1937 are marked in green and the present ones in red. Numerical values are relevant indicators for a given period

the area overlap on the statistical data. This threshold allows minimizing errors related to the vectorization of polygons. This is very important for the credibility of the results. For counties that are the subject of our research, three thresholds of 60%, 65% and 70% (Figure 5, 6, 7 i 8) have been checked. The comparison shows that the 60% threshold is the most effective. Almost the entire area overlapping can be expressed in one of the variants. Only three areas do not meet the criterion: Warsaw and its surroundings, Łódź and its surroundings, and the area of the former Śląskie Voivodeship (Figure 9), Łódź and Warsaw have significantly enlarged over the course of 100 years. For this reason, the borders of the neighboring counties have changed significantly compared to the situation in the Second Polish Republic. The entire present-day Silesian Voivodeship has a completely different structure than that of 1937. and it is not possible to compare them under the assumptions made.

2.4 Population calculations

In the previous steps, we made an initial selection among the surveyed counties. Now we are able to count the number of people for the common part of administrative units. The data that we have obtained from various sources allows us to approach the matter in two ways — using information about the population number or using information about the population density in the counties. In both cases, the results for the 1:1 variant are identical.

The first one is important because we will have more information on this topic. We assume that the population density for the entire area is constant. Then the populations in the common part can be expressed as the product of the ratio of the area of the common part to the area of the county and the population of the whole county.

$$pop_{os} = pop_{otn} \cdot pop_o \tag{3}$$

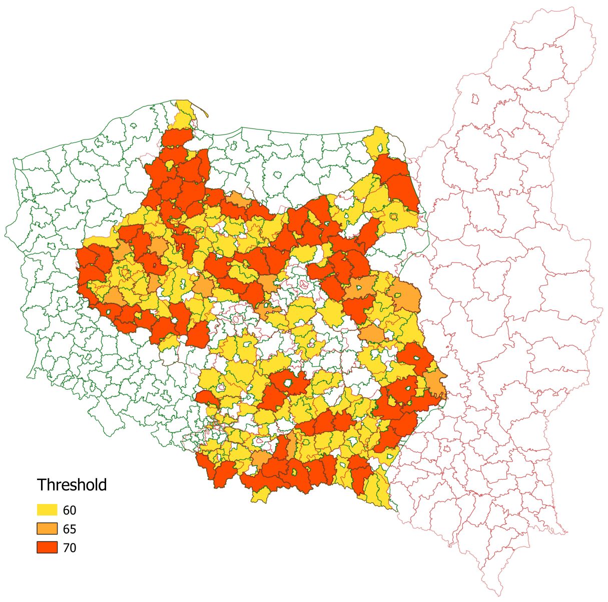


Figure 5: Comparison of the results from the 1:1 variant from the three thresholds 60%, 65% and 70% for the entire research area

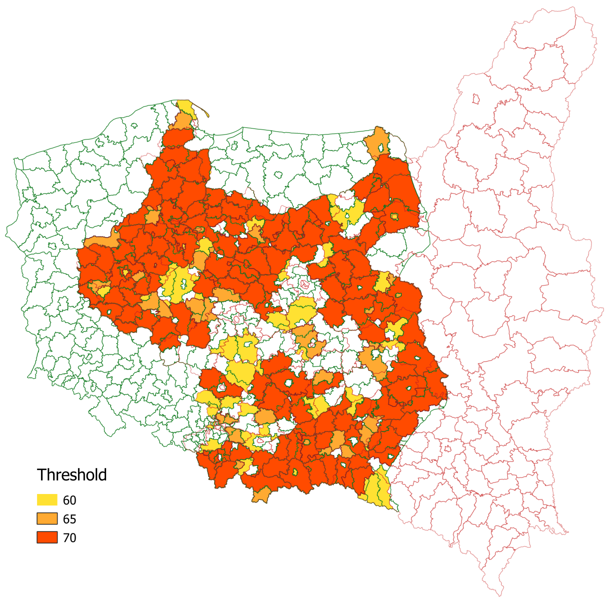


Figure 6: Comparison of the results from the 2:2 variant from the three thresholds 60%, 65% and 70% for the entire research area

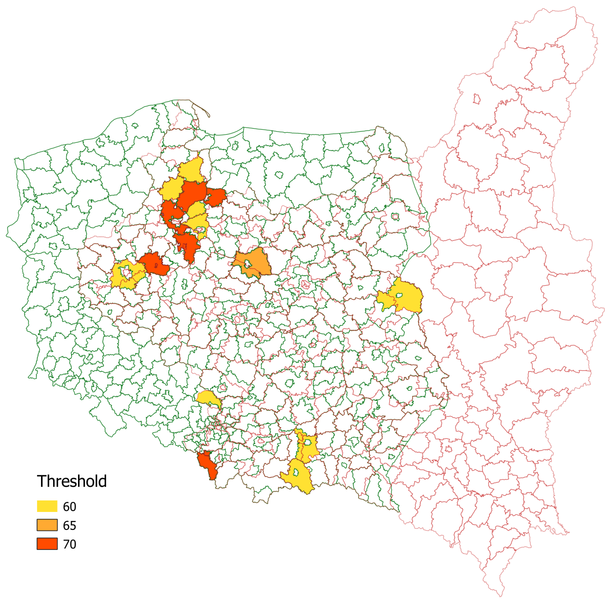


Figure 7: Comparison of the results from the 2:1 variant from the three thresholds 60%, 65% and 70% for the entire research area

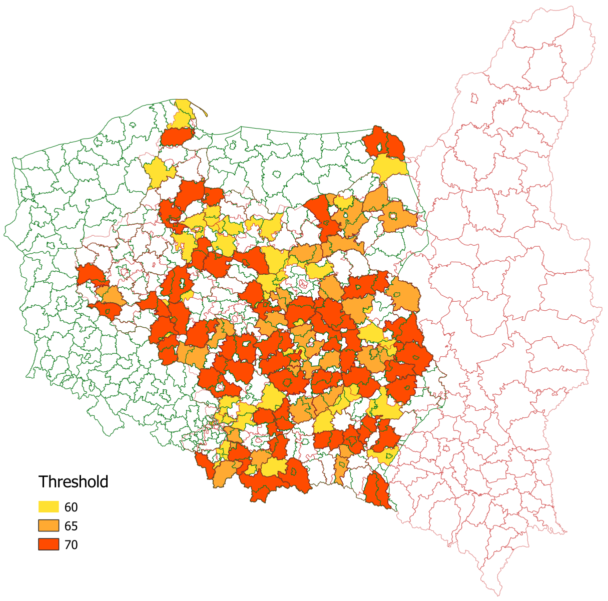


Figure 8: Comparison of the results from the 1:2 variant from the three thresholds 60%, 65% and 70% for the entire research area

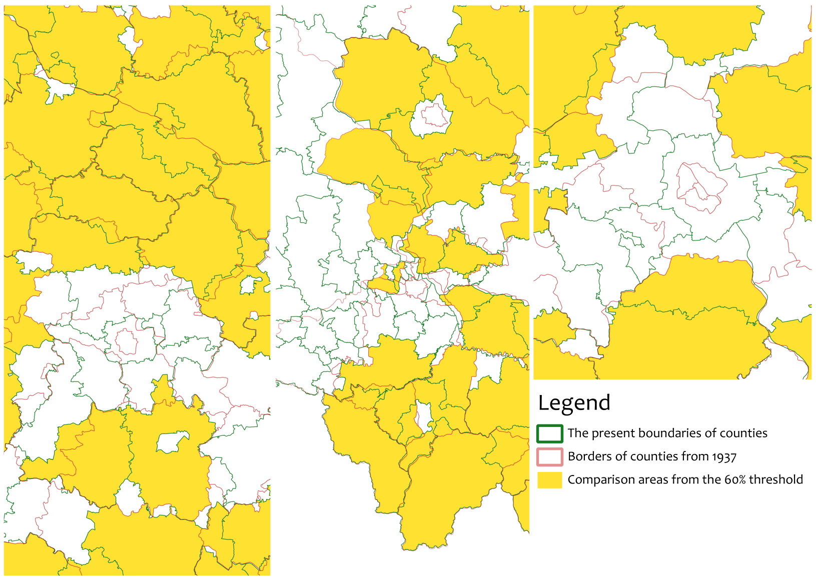

Legend
🟩 The present boundaries of counties
🟥 Borders of counties from 1937
🟨 Comparison areas from the 60% threshold

Figure 9: Three regions that do not meet the criterion of overlapping in at least 60% (From the left: Łódź and its surroundings, the Silesian Voivodeship, Warsaw and its surroundings)

$$pop_{ns} = pop_{nto} \cdot pop_n \tag{4}$$

where:

- pop_{os} — population number recorded in the "old" county in the part shared with the "new" county
- pop_{ns} — population number recorded in the "new" county in the part shared with the "old" county
- pop_{otn} — ratio of the common area to the "new" county area, calculated from the formula(1)
- pop_{nto} — ratio of the common area to the "old" county area, calculated from the formula(2)
- pop_o — population in the "old" county
- pop_n — population in the "new" county

The pattern, however, has its drawbacks. The most important assumption is that the entire county has the same population density. Administrative units consist of several smaller or larger towns and the population density is the highest there. Additionally, the population itself in the county may be burdened with a statistical error, which directly influences the increase of the margin of error in the calculated value.

It is worth looking for a different approach to the issue. The Second General Census in Poland makes this opportunity possible. You can find separate information on the urban and rural population density in the overview. Moreover, at present some cities operate with county status. The combination of these relationships partially solves the problem of uniform population density. We are able to distinguish a situation in which the appropriate size for individual counties should be used. The patterns in individual cases are as follows:

when the 'old' county is compared with the land county

$$pop_{old} = ind_{oldinpre} \cdot den_v \cdot pow_{old} \tag{5}$$

when the 'old' county is compared with the township county

$$pop_{old} = ind_{oldinpre} \cdot den_t \cdot area_o \tag{6}$$

for a 'new' county with no density distinctions

$$pop_{pre} = ind_{preinold} \cdot den \cdot area_n \tag{7}$$

where:

- pop_{old} — population of the "old" county in the compared area
- pop_{pre} — population of the "new" county in the compared area
- $ind_{preinold}$ — ratio of the common area to the "new" county area, calculated from the formula (1)
- ind_{nws} — ratio of the common area to the "old" county area, calculated from the formula (2)
- den_v — population density in the countryside in the "old" county per km^2
- den_t — population density in]cities w "old" county per km^2
- den — population density in the "new" county per km^2
- $area_o$ — area "old" county in km^2
- $area_n$ — area "new" county in km^2

Unfortunately, we achieve reliable comparison results with the 1:1 variant. The smaller the areas are juxtaposed, the greater the chance of overestimating the population size. There may also be errors caused by incorrect calculation of the area for the administrative unit (vectorization errors, use of the wrong coordinate system).

3 Tests and summary

The program was tested for all four variants of the comparison and also for the 1:1 case for both methods of calculating the population size. Shapefile results can be opened in any GIS software that supports this format. On the other hand, the results in the form of a text file in .txt format can be easily transformed into a pivot table in a spreadsheet. This file does not contain information about the geometry of objects, but you can use the software to create a relationship between the result file and one of the layers used in the program based on the identifiers of objects.

For the 1:1 variant, the indices ind_{swn} and ind_{nws} have the same values for both calculation methods (Figure 10). The present boundaries of counties more often reach the values of 90% and more. The results in the table for both methods for several pairs are identical (Figure 11). The maximum difference in the number of people between the methods is 92 957[2]. The average difference is 11 872.41. In both cases, 96 pairs of counties were found.

For 2:2 comparisons, 354 joins met the criteria. As in the previous case, the boundaries of the present counties had a greater share in the area of the common part in relation to the area of the entire counties (Figure 12). A comparison of Śremski and Poznański counties from 1937 to the present City of Poznań and Poznań counties shows the largest difference in the number of inhabitants (106 045 in 1937, 639 650 — now), which is 533 605 (Figure 13). In 169 records, we can see a greater number of inhabitants living in the common area in 1937 than today. The biggest difference can be observed when comparing Tarnogórski and Lubliniec counties with each other, it amounts to 191 424 (131 928 — now, 323 352 — in 1937). On average, for the entire area of comparisons, the number of inhabitants increased by 11062 people.

Variant 2:1 shows 23 comparisons. Most of them are located in the following voivodeships: Pomorskie (now Kujawsko-Pomorskie) and Poznań (now Wielkopolskie). Once again, the boundaries of the current counties had a greater share in the area of the common part in relation to the area of the entire counties (Figure 14, 15). On average, for the entire area of comparisons, the population increased by 4056 people.

Variant 1:2 returned 142 objects. Among them, the largest difference in the number of people was 366 746 (Figure 17). On the other hand, the largest decrease in the number of people can be observed in the present area of the Leżajsk and Łańcut counties compared to the former Łańcut county, it amounts to 106 931. On average, for the entire area of comparisons, the population dropped by 29 325 osób. Here, the boundaries of the counties from 1937 had a greater share in the area of the common part in relation to the area of the entire counties (Figure 16).

This research is a basis for the next work. Namely, the aim is to allow the machine to choose propper areas (that is, the ones which are sufficiently similar) on its own, and compare their available data for their respective periods.

[2]For the Lubliniec county in the Śląskie voivoideship method 1 — 233 939, method 2 — 140 982.

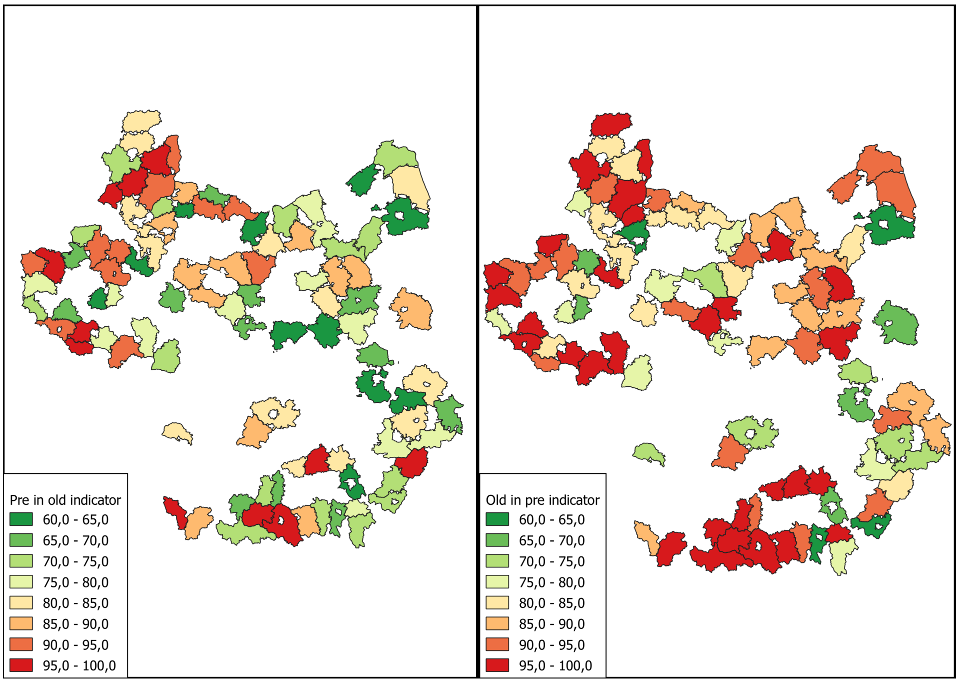
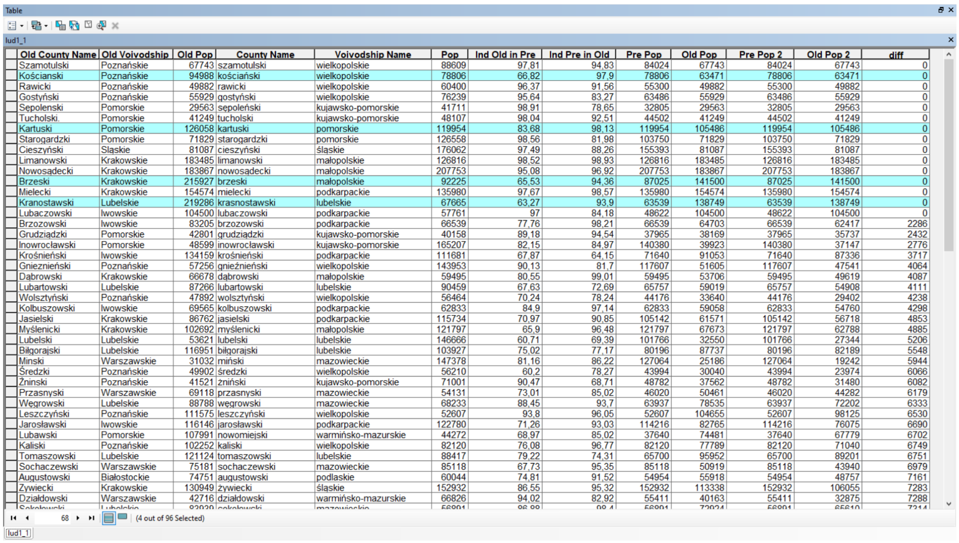


Figure 10: Common parts of counties for variant 1:1. On the left, classified by index ind_{otn}, and on the right ind_{nto}



Old County Name	Old Voivodship	Old Pop	County Name	Voivodship Name	Pop	Ind Old in Pre	Ind Pre in Old	Pre Pop	Old Pop	Pre Pop 2	Old Pop 2	diff
Szamotulski	Poznańskie	67743	szamotulski	wielkopolskie	88609	97,81	94,83	84024	67743	84024	67743	0
Kościanski	Poznańskie	94988	kościański	wielkopolskie	78806	66,82	97,9	78806	63471	78806	63471	0
Rawicki	Poznańskie	49882	rawicki	wielkopolskie	60400	96,37	91,56	55300	49882	55300	49882	0
Gostyński	Poznańskie	55929	gostyński	wielkopolskie	76239	95,64	83,27	63486	55929	63486	55929	0
Sępolenski	Pomorskie	29563	sępoleński	kujawsko-pomorskie	41711	98,91	78,65	32805	29563	32805	29563	0
Tucholski.	Pomorskie	41249	tucholski	kujawsko-pomorskie	48107	98,04	92,51	44502	41249	44502	41249	0
Kartuski	Pomorskie	126058	kartuski	pomorskie	119954	83,68	98,13	119954	105486	119954	105486	0
Starogardzki	Pomorskie	71829	starogardzki	pomorskie	126558	98,56	81,98	103750	71829	103750	71829	0
Cieszyński	Slaskie	81087	cieszyński	śląskie	176062	97,49	88,26	155393	81087	155393	81087	0
Limanowski	Krakowskie	183485	limanowski	małopolskie	126816	98,52	98,93	126816	183485	126816	183485	0
Nowosądecki	Krakowskie	183867	nowosądecki	małopolskie	207753	95,08	96,92	207753	183867	207753	183867	0
Brzeski	Krakowskie	215927	brzeski	małopolskie	92225	65,53	94,36	87025	141500	87025	141500	0
Mielecki	Krakowskie	154574	mielecki	podkarpackie	135980	97,67	98,57	135980	154574	135980	154574	0
Kranostawski	Lubelskie	219286	krasnostawski	lubelskie	67665	63,27	93,9	63539	138749	63539	138749	0
Lubaczowski	lwowskie	104500	lubaczowski	podkarpackie	57761	97	84,18	48622	104500	48622	104500	0
Brzozowski	lwowskie	83205	brzozowski	podkarpackie	66539	77,76	98,21	66539	64703	66539	62417	2286
Grudziądzki	Pomorskie	42801	grudziądzki	kujawsko-pomorskie	40158	89,18	94,54	37965	38169	37965	35737	2432
Inowrocławski	Pomorskie	48599	inowrocławski	kujawsko-pomorskie	165207	82,15	84,97	140380	39923	140380	37147	2776
Krośnieński	lwowskie	134159	krośnieński	podkarpackie	111681	67,87	64,15	71640	91053	71640	87336	3717
Gnieznieński	Poznańskie	57256	gnieźnieński	wielkopolskie	143953	90,13	81,7	117607	51605	117607	47541	4064
Dąbrowski	Krakowskie	66678	dąbrowski	małopolskie	59495	80,55	99,01	59495	53706	59495	49619	4087
Lubartowski	Lubelskie	87266	lubartowski	lubelskie	90459	67,63	72,69	65757	59019	65757	54908	4111
Wolsztyński	Poznańskie	47892	wolsztyński	wielkopolskie	56464	70,24	78,24	44176	33640	44176	29402	4238
Kolbuszowski	lwowskie	69565	kolbuszowski	podkarpackie	62833	84,9	97,14	62833	59058	62833	54760	4298
Jasielski	Krakowskie	86762	jasielski	podkarpackie	115734	70,97	90,85	105142	61571	105142	56718	4853
Myślenicki	Krakowskie	102692	myślenicki	małopolskie	121797	65,9	96,48	121797	67673	121797	62788	4885
Lubelski	Lubelskie	53621	lubelski	lubelskie	146666	60,71	69,39	101766	32550	101766	27344	5206
Biłgorajski	Lubelskie	116951	biłgorajski	lubelskie	103927	75,02	77,17	80196	87737	80196	82189	5548
Minski	Warszawskie	31032	miński	mazowieckie	147378	81,16	86,22	127064	25186	127064	19242	5944
Średzki	Poznańskie	49902	średzki	wielkopolskie	56210	60,2	78,27	43994	30040	43994	23974	6066
Żninski	Poznańskie	41521	żniński	kujawsko-pomorskie	71001	90,47	68,71	48782	37562	48782	31480	6082
Przasnyski	Warszawskie	69118	przasnyski	mazowieckie	54131	73,01	85,02	46020	50461	46020	44282	6179
Węgrowski	Lubelskie	88788	węgrowski	mazowieckie	68233	88,45	93,7	63937	78535	63937	72202	6333
Leszczyński	Poznańskie	111575	leszczyński	wielkopolskie	52607	93,8	96,05	52607	104655	52607	98125	6530
Jarosławski	lwowskie	116146	jarosławski	podkarpackie	122780	71,26	93,03	114216	82765	114216	76075	6690
Lubawski	Pomorskie	107991	nowomiejski	warmińsko-mazurskie	44272	68,97	85,02	37640	74481	37640	67779	6702
Kaliski	Poznańskie	102252	kaliski	wielkopolskie	82120	76,08	96,77	82120	77789	82120	71040	6749
Tomaszowski	Lubelskie	121124	tomaszowski	lubelskie	88417	79,22	74,31	65700	95952	65700	89201	6751
Sochaczewski	Warszawskie	75181	sochaczewski	mazowieckie	85118	67,73	95,35	85118	50919	85118	43940	6979
Augustowski	Białostockie	74751	augustowski	podlaskie	60044	74,81	91,52	54954	55918	54954	48757	7161
Zywiecki	Krakowskie	130949	żywiecki	śląskie	152932	86,55	95,32	152932	113338	152932	106055	7283
Działdowski	Warszawskie	42716	działdowski	warmińsko-mazurskie	66826	94,02	82,92	55411	40163	55411	32875	7288

Figure 11: Fragment of the result layer attributes table for the 1:1 variant. The records marked in blue are counties with the same population, calculated using two methods

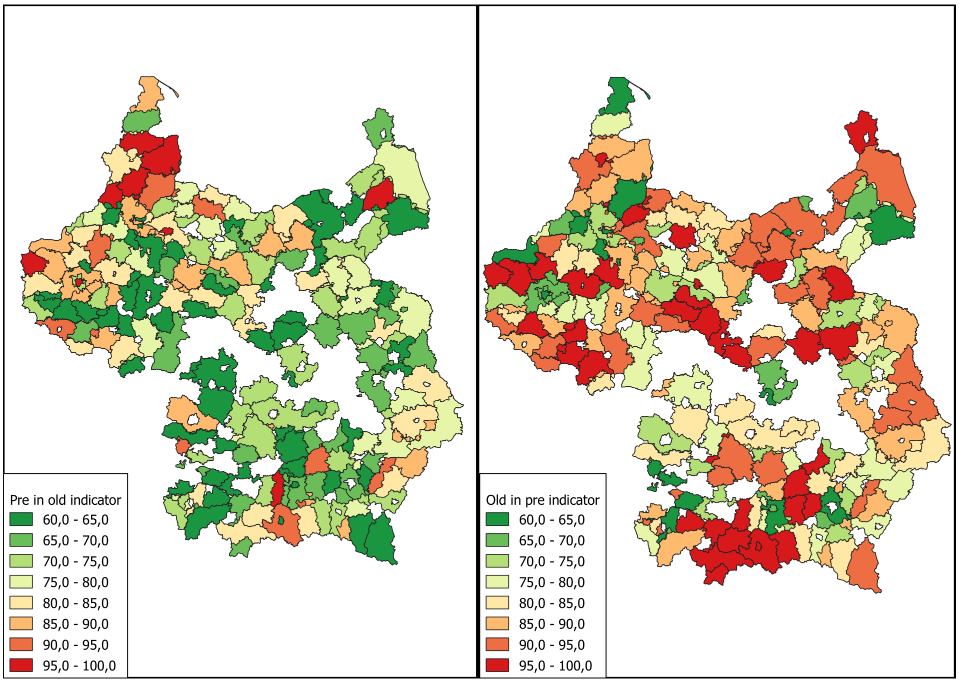
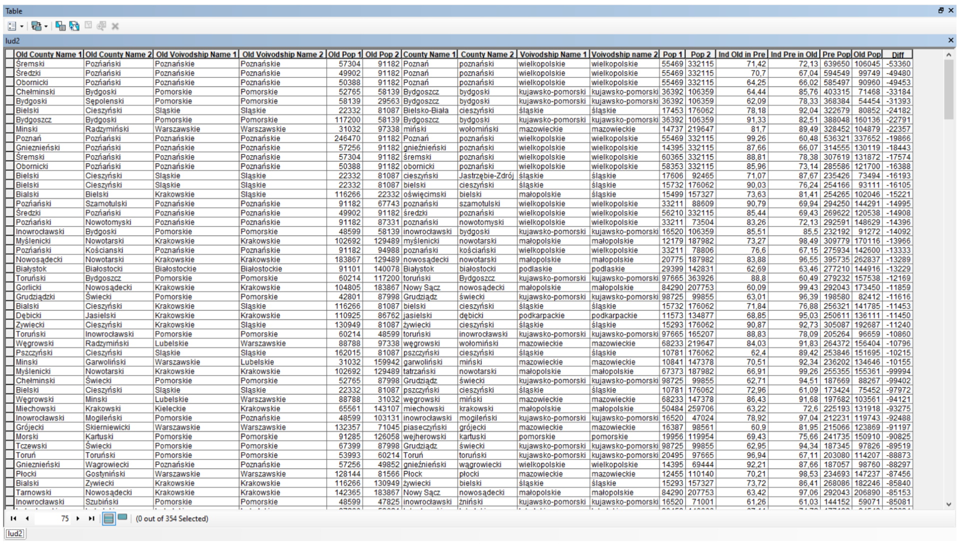


Figure 12: Common parts of counties for variant 2:2. On the left, classified by index ind_{otn}, and on the right ind_{nto}



Table

lud2

Old County Name 1	Old County Name 2	Old Voivodship Name 1	Old Voivodship Name 2	Old Pop 1	Old Pop 2	County Name 1	County Name 2	Voivodship Name 1	Voivodship name 2	Pop 1	Pop 2	Ind Old in Pre	Ind Pre in Old	Pre Pop	Old Pop	Diff
Śremski	Pozńański	Poznańskie	Poznańskie	57304	91182	Poznań	poznański	wielkopolskie	wielkopolskie	55469	332115	71,42	72,13	639650	106045	-53360
Średzki	Pozńański	Poznańskie	Poznańskie	49902	91182	Poznań	poznański	wielkopolskie	wielkopolskie	55469	332115	70,7	67,04	594549	99749	-49480
Obornicki	Pozńański	Poznańskie	Poznańskie	50388	91182	Poznań	poznański	wielkopolskie	wielkopolskie	55469	332115	64,25	66,02	585497	90960	-49453
Chełminski	Bydgoski	Pomorskie	Pomorskie	52765	58139	Bydgoszcz	bydgoski	kujawsko-pomorski	kujawsko-pomorski	36392	106359	64,44	85,76	403315	71468	-33184
Bydgoski	Sępolenski	Pomorskie	Pomorskie	58139	29563	Bydgoszcz	bydgoski	kujawsko-pomorski	kujawsko-pomorski	36392	106359	62,09	78,33	368384	54454	-31393
Bielski	Cieszyński	Sląskie	Sląskie	22332	81087	Bielsko-Biała	cieszyński	śląskie	śląskie	17453	176062	78,18	92,04	322679	80852	-24182
Bydgoszcz	Bydgoski	Pomorskie	Pomorskie	117200	58139	Bydgoszcz	bydgoski	kujawsko-pomorski	kujawsko-pomorski	36392	106359	91,33	82,51	388048	160136	-22791
Minski	Radzymiński	Warszawskie	Warszawskie	31032	97338	miński	wołomiński	mazowieckie	mazowieckie	14737	219647	81,7	89,49	328452	104879	-22357
Poznań	Pozńański	Poznańskie	Poznańskie	246470	91182	Poznań	poznański	wielkopolskie	wielkopolskie	55469	332115	99,26	60,48	536321	337652	-19866
Gnieznieński	Pozńański	Poznańskie	Poznańskie	57256	91182	gnieźnieński	poznański	wielkopolskie	wielkopolskie	14395	332115	87,66	66,07	314555	130119	-18443
Śremski	Pozńański	Poznańskie	Poznańskie	57304	91182	śremski	poznański	wielkopolskie	wielkopolskie	60365	332115	88,81	78,38	307619	131872	-17574
Obornicki	Pozńański	Poznańskie	Poznańskie	50388	91182	obornicki	poznański	wielkopolskie	wielkopolskie	58353	332115	85,96	73,14	285586	121700	-16388
Bielski	Cieszyński	Sląskie	Sląskie	22332	81087	cieszyński	Jastrzębie-Zdrój	śląskie	śląskie	17606	92465	71,07	87,67	235426	73494	-16193
Bielski	Cieszyński	Sląskie	Sląskie	22332	81087	bielski	cieszyński	śląskie	śląskie	15732	176062	90,03	76,24	254166	93111	-16105
Bialski	Bielski	Krakowskie	Sląskie	116266	22332	oświęcimski	bielski	małopolskie	śląskie	15499	157327	73,63	81,41	254265	102046	-15221
Pozńański	Szamotulski	Poznańskie	Poznańskie	91182	67743	poznański	szamotulski	wielkopolskie	wielkopolskie	33211	88609	90,79	69,94	294250	144291	-14995
Średzki	Pozńański	Poznańskie	Poznańskie	49902	91182	średzki	poznański	wielkopolskie	wielkopolskie	56210	332115	85,44	69,43	269622	120538	-14908
Pozńański	Nowotomyski	Poznańskie	Poznańskie	91182	87331	poznański	nowotomyski	wielkopolskie	wielkopolskie	33211	73504	83,26	72,13	292591	148629	-14396
Inowrocławski	Bydgoski	Pomorskie	Pomorskie	48599	58139	inowrocławski	bydgoski	kujawsko-pomorski	kujawsko-pomorski	16520	106359	85,51	85,5	232192	91272	-14092
Myślenicki	Nowotarski	Krakowskie	Krakowskie	102692	129489	myślenicki	nowotarski	małopolskie	małopolskie	12179	187982	73,27	98,49	309779	170116	-13966
Pozńański	Kościanski	Poznańskie	Poznańskie	91182	94988	poznański	kościański	wielkopolskie	wielkopolskie	33211	78806	76,6	67,15	275934	142600	-13333
Nowosądecki	Nowotarski	Krakowskie	Krakowskie	183867	129489	nowosądecki	nowotarski	małopolskie	małopolskie	20775	187982	83,88	96,55	395735	262837	-13289
Białystok	Białostocki	Białostockie	Białostockie	91101	140078	Białystok	białostocki	podlaskie	podlaskie	29399	142831	62,69	63,46	277210	144916	-13229
Toruński	Bydgoszcz	Pomorskie	Pomorskie	60214	117200	toruński	Bydgoszcz	kujawsko-pomorski	kujawsko-pomorski	97665	363926	88,8	60,49	279232	157538	-12169
Gorlicki	Nowosądecki	Krakowskie	Krakowskie	104805	183867	Nowy Sącz	nowosądecki	małopolskie	małopolskie	84290	207753	60,09	99,43	292043	173450	-11859
Grudziądzki	Świecki	Pomorskie	Pomorskie	42801	87998	Grudziądz	świecki	kujawsko-pomorski	kujawsko-pomorski	98725	99855	63,01	96,39	198580	82412	-11616
Bialski	Cieszyński	Krakowskie	Sląskie	116266	81087	bielski	cieszyński	śląskie	śląskie	15732	176062	71,84	76,88	256321	141785	-11453
Dębicki	Jasielski	Krakowskie	Krakowskie	110925	86762	jasielski	dębicki	podkarpackie	podkarpackie	11573	134877	68,85	95,03	250611	136111	-11450
Zywiecki	Cieszyński	Krakowskie	Sląskie	130949	81087	żywiecki	cieszyński	śląskie	śląskie	15293	176062	90,87	92,73	305087	192687	-11240
Toruński	Inowrocławski	Pomorskie	Pomorskie	60214	48599	toruński	inowrocławski	kujawsko-pomorski	kujawsko-pomorski	97665	165207	88,83	78,09	205264	96659	-10860
Węgrowski	Radzymiński	Lubelskie	Warszawskie	88788	97338	węgrowski	wołomiński	mazowieckie	mazowieckie	68233	219647	84,03	91,83	264372	156404	-10796
Pszczyński	Cieszyński	Sląskie	Sląskie	162015	81087	pszczyński	cieszyński	śląskie	śląskie	10781	176062	62,4	89,42	253846	151695	-10215
Minski	Garwoliński	Warszawskie	Lubelskie	31032	159942	garwoliński	miński	mazowieckie	mazowieckie	10841	147378	70,51	92,34	236202	134646	-10155
Myślenicki	Nowotarski	Krakowskie	Krakowskie	102692	129489	tatrzański	nowotarski	małopolskie	małopolskie	67373	187982	66,91	99,26	255355	155361	-99994
Chełminski	Świecki	Pomorskie	Pomorskie	52765	87998	Grudziądz	świecki	kujawsko-pomorski	kujawsko-pomorski	98725	99855	62,71	94,51	187669	88267	-99402
Bielski	Cieszyński	Sląskie	Sląskie	22332	81087	pszczyński	cieszyński	śląskie	śląskie	10781	176062	72,96	61,09	173424	75452	-97972
Węgrowski	Minski	Lubelskie	Warszawskie	88788	31032	węgrowski	miński	mazowieckie	mazowieckie	68233	147378	86,43	91,68	197682	103561	-94121
Miechowski	Krakowski	Kieleckie	Krakowskie	65561	143107	miechowski	krakowski	małopolskie	małopolskie	50484	259706	63,22	72,6	225193	131918	-93275
Inowrocławski	Mogileński	Pomorskie	Poznańskie	48599	103131	inowrocławski	mogileński	kujawsko-pomorski	kujawsko-pomorski	16520	47024	78,92	97,04	212231	119743	-92488
Grójecki	Skierniewicki	Warszawskie	Warszawskie	132357	71045	piaseczyński	grójecki	mazowieckie	mazowieckie	16387	98561	60,9	81,95	215066	123869	-91197
Morski	Kartuski	Pomorskie	Pomorskie	91285	126058	wejherowski	kartuski	pomorskie	pomorskie	19956	119954	69,43	75,66	241735	150910	-90825
Tczewski	Świecki	Pomorskie	Pomorskie	67399	87998	Grudziądz	świecki	kujawsko-pomorski	kujawsko-pomorski	98725	99855	62,95	94,34	187345	97826	-89519
Toruń	Toruński	Pomorskie	Pomorskie	53993	60214	Toruń	toruński	kujawsko-pomorski	kujawsko-pomorski	20495	97665	96,94	67,11	203080	114207	-88873
Gnieznieński	Wagrowiecki	Poznańskie	Poznańskie	57256	49852	gnieźnieński	wągrowiecki	wielkopolskie	wielkopolskie	14395	69444	92,21	87,66	187057	98760	-88297
Płocki	Gostyniński	Warszawskie	Warszawskie	128144	81566	Płock	płocki	mazowieckie	mazowieckie	12455	110140	70,21	98,53	234693	147237	-87456
Bialski	Zywiecki	Krakowskie	Krakowskie	116266	130949	żywiecki	bielski	śląskie	śląskie	15293	157327	73,72	86,41	268086	182246	-85840
Tarnowski	Nowosądecki	Krakowskie	Krakowskie	142365	183867	Nowy Sącz	nowosądecki	małopolskie	małopolskie	84290	207753	63,42	97,06	292043	206890	-85153
Inowrocławski	Szubiński	Pomorskie	Pomorskie	48599	47825	inowrocławski	żniński	kujawsko-pomorski	kujawsko-pomorski	16520	71001	61,26	61,03	144152	59071	-85081

75 (0 out of 354 Selected)

lud2

Figure 13: Fragment of the result layer attributes table for the 2:2 variant

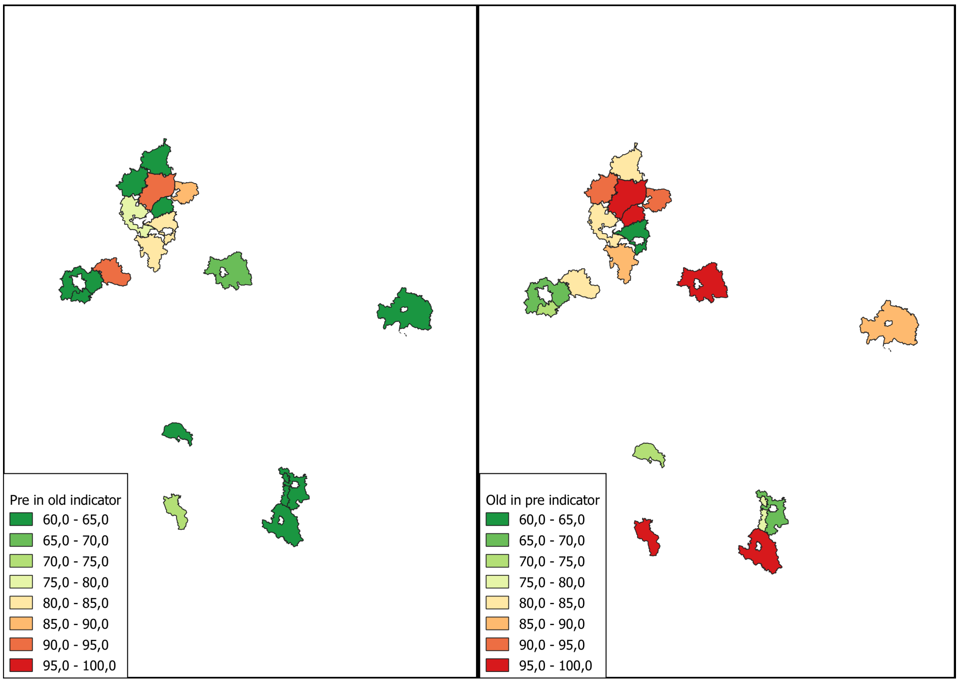
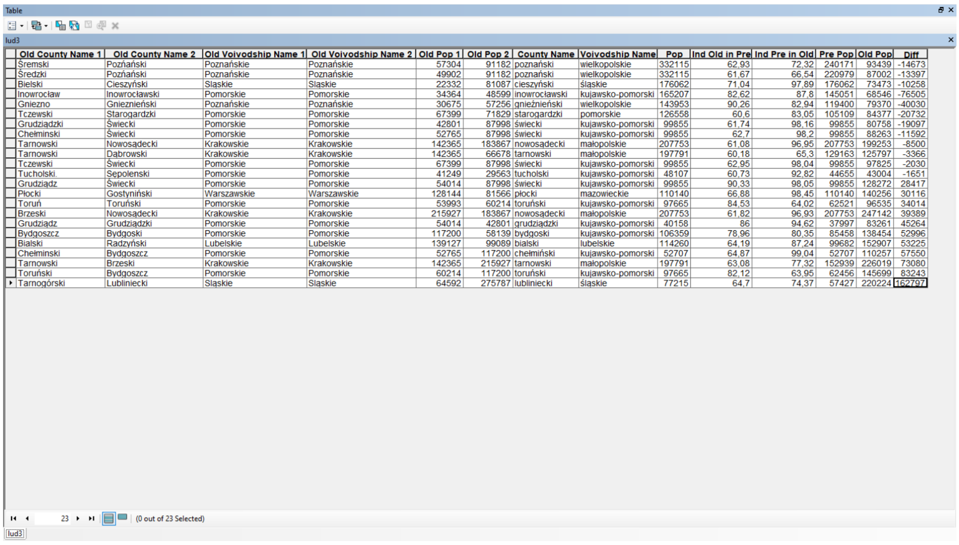


Figure 14: Common parts of counties for variant 2:1. On the left, classified by index ind_{otn}, and on the right ind_{nto}



Table

lud3

Old County Name 1	Old County Name 2	Old Voivodship Name 1	Old Voivodship Name 2	Old Pop 1	Old Pop 2	County Name	Voivodship Name	Pop	Ind Old in Pre	Ind Pre in Old	Pre Pop	Old Pop	Diff
Śremski	Pozńański	Poznańskie	Poznańskie	57304	91182	poznański	wielkopolskie	332115	62,93	72,32	240171	93439	-14673
Średzki	Pozńański	Poznańskie	Poznańskie	49902	91182	poznański	wielkopolskie	332115	61,67	66,54	220979	87002	-13397
Bielski	Cieszyński	Sląskie	Sląskie	22332	81087	cieszyński	śląskie	176062	71,04	97,89	176062	73473	-10258
Inowrocław	Inowrocławski	Pomorskie	Pomorskie	34364	48599	inowrocławski	kujawsko-pomorski	165207	82,62	87,8	145051	68546	-76505
Gniezno	Gnieznieński	Poznańskie	Poznańskie	30675	57256	gnieźnieński	wielkopolskie	143953	90,26	82,94	119400	79370	-40030
Tczewski	Starogardzki	Pomorskie	Pomorskie	67399	71829	starogardzki	pomorskie	126558	60,6	83,05	105109	84377	-20732
Grudziądzki	Świecki	Pomorskie	Pomorskie	42801	87998	świecki	kujawsko-pomorski	99855	61,74	98,16	99855	80758	-19097
Chełminski	Świecki	Pomorskie	Pomorskie	52765	87998	świecki	kujawsko-pomorski	99855	62,7	98,2	99855	88263	-11592
Tarnowski	Nowosądecki	Krakowskie	Krakowskie	142365	183867	nowosądecki	małopolskie	207753	61,08	96,95	207753	199253	-8500
Tarnowski	Dąbrowski	Krakowskie	Krakowskie	142365	66678	tarnowski	małopolskie	197791	60,18	65,3	129163	125797	-3366
Tczewski	Świecki	Pomorskie	Pomorskie	67399	87998	świecki	kujawsko-pomorski	99855	62,95	98,04	99855	97825	-2030
Tucholski.	Sępolenski	Pomorskie	Pomorskie	41249	29563	tucholski	kujawsko-pomorski	48107	60,73	92,82	44655	43004	-1651
Grudziądz	Świecki	Pomorskie	Pomorskie	54014	87998	świecki	kujawsko-pomorski	99855	90,33	98,05	99855	128272	28417
Płocki	Gostyniński	Warszawskie	Warszawskie	128144	81566	płocki	mazowieckie	110140	66,88	98,45	110140	140256	30116
Toruń	Toruński	Pomorskie	Pomorskie	53993	60214	toruński	kujawsko-pomorski	97665	84,53	64,02	62521	96535	34014
Brzeski	Nowosądecki	Krakowskie	Krakowskie	215927	183867	nowosądecki	małopolskie	207753	61,82	96,93	207753	247142	39389
Grudziądz	Grudziądzki	Pomorskie	Pomorskie	54014	42801	grudziądzki	kujawsko-pomorski	40158	86	94,62	37997	83261	45264
Bydgoszcz	Bydgoski	Pomorskie	Pomorskie	117200	58139	bydgoski	kujawsko-pomorski	106359	78,96	80,35	85458	138454	52996
Bialski	Radzyński	Lubelskie	Lubelskie	139127	99089	bialski	lubelskie	114260	64,19	87,24	99682	152907	53225
Chełminski	Bydgoszcz	Pomorskie	Pomorskie	52765	117200	chełmiński	kujawsko-pomorski	52707	64,87	99,04	52707	110257	57550
Tarnowski	Brzeski	Krakowskie	Krakowskie	142365	215927	tarnowski	małopolskie	197791	63,08	77,32	152939	226019	73080
Toruński	Bydgoszcz	Pomorskie	Pomorskie	60214	117200	toruński	kujawsko-pomorski	97665	82,12	63,95	62456	145699	83243
Tarnogórski	Lubliniecki	Sląskie	Sląskie	64592	275787	lubliniecki	śląskie	77215	64,7	74,37	57427	220224	162797

23 (0 out of 23 Selected)

lud3

Figure 15: Result layer attributes table for the 2:1 variant with an additional column with the difference in population number

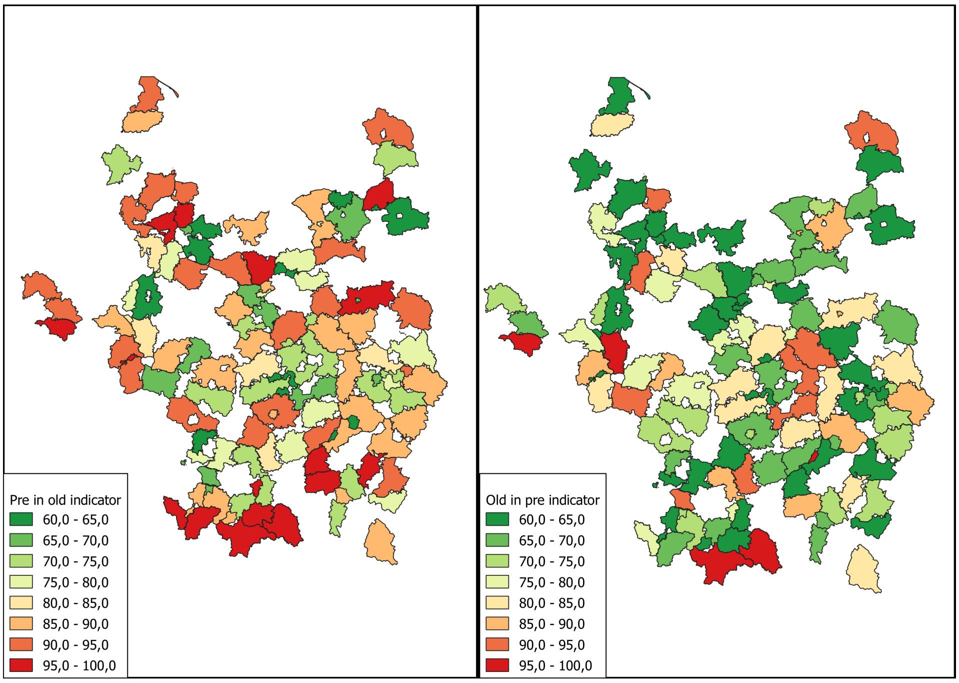
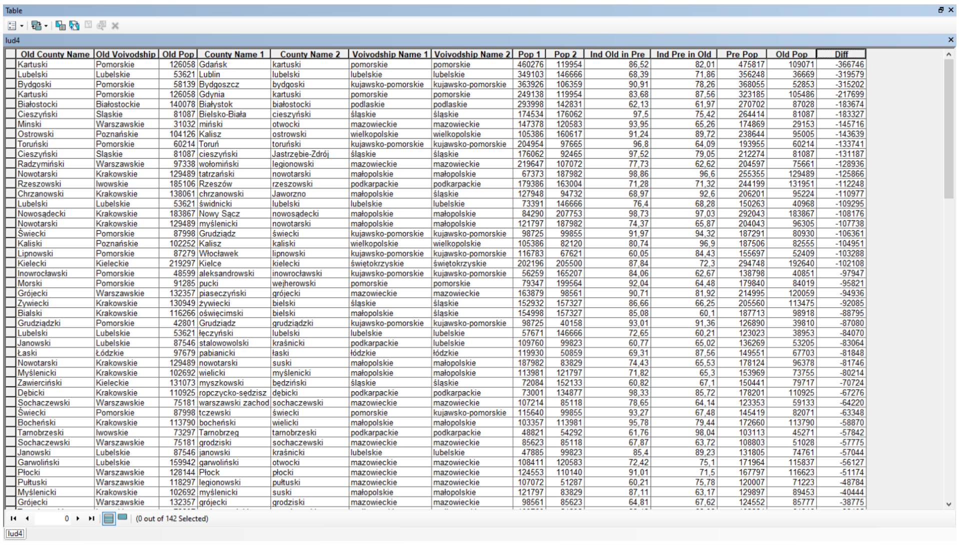


Figure 16: Common parts of counties for variant 1:2. On the left, classified by index ind_{otn}, and on the right ind_{nto}



Old County Name	Old Voivodship	Old Pop	County Name 1	County Name 2	Voivodship Name 1	Voivodship Name 2	Pop 1	Pop 2	Ind Old in Pre	Ind Pre in Old	Pre Pop	Old Pop	Diff
Kartuski	Pomorskie	126058	Gdańsk	kartuski	pomorskie	pomorskie	460276	119954	86,52	82,01	475817	109071	-366746
Lubelski	Lubelskie	53621	Lublin	lubelski	lubelskie	lubelskie	349103	146666	68,39	71,86	356248	36669	-319579
Bydgoski	Pomorskie	58139	Bydgoszcz	bydgoski	kujawsko-pomorskie	kujawsko-pomorskie	363926	106359	90,91	78,26	368055	52853	-315202
Kartuski	Pomorskie	126058	Gdynia	kartuski	pomorskie	pomorskie	249138	119954	83,68	87,56	323185	105486	-217699
Białostocki	Białostockie	140078	Białystok	białostocki	podlaskie	podlaskie	293998	142831	62,13	61,97	270702	87028	-183674
Cieszyński	Slaskie	81087	Bielsko-Biała	cieszyński	śląskie	śląskie	174534	176062	97,5	75,42	264414	81087	-183327
Minski	Warszawskie	31032	miński	otwocki	mazowieckie	mazowieckie	147378	120583	93,95	65,26	174869	29153	-145716
Ostrowski	Poznańskie	104126	Kalisz	ostrowski	wielkopolskie	wielkopolskie	105386	160617	91,24	89,72	238644	95005	-143639
Toruński	Pomorskie	60214	Toruń	toruński	kujawsko-pomorskie	kujawsko-pomorskie	204954	97665	96,8	64,09	193955	60214	-133741
Cieszyński	Slaskie	81087	cieszyński	Jastrzębie-Zdrój	śląskie	śląskie	176062	92465	97,52	79,05	212274	81087	-131187
Radzymiński	Warszawskie	97338	wołomiński	legionowski	mazowieckie	mazowieckie	219647	107072	77,73	62,62	204597	75661	-128936
Nowotarski	Krakowskie	129489	tatrzański	nowotarski	małopolskie	małopolskie	67373	187982	98,86	96,6	255355	129489	-125866
Rzeszowski	lwowskie	185106	Rzeszów	rzeszowski	podkarpackie	podkarpackie	179386	163004	71,28	71,32	244199	131951	-112248
Chrzanowski	Krakowskie	138061	chrzanowski	Jaworzno	małopolskie	śląskie	127948	94732	68,97	92,6	206201	95224	-110977
Lubelski	Lubelskie	53621	świdnicki	lubelski	lubelskie	lubelskie	73391	146666	76,4	68,28	150263	40968	-109295
Nowosądecki	Krakowskie	183867	Nowy Sącz	nowosądecki	małopolskie	małopolskie	84290	207753	98,73	97,03	292043	183867	-108176
Nowotarski	Krakowskie	129489	myślenicki	nowotarski	małopolskie	małopolskie	121797	187982	74,37	65,87	204043	96305	-107738
Świecki	Pomorskie	87998	Grudziądz	świecki	kujawsko-pomorskie	kujawsko-pomorskie	98725	99855	91,97	94,32	187291	80930	-106361
Kaliski	Poznańskie	102252	Kalisz	kaliski	wielkopolskie	wielkopolskie	105386	82120	80,74	96,9	187506	82555	-104951
Lipnowski	Pomorskie	87279	Włocławek	lipnowski	kujawsko-pomorskie	kujawsko-pomorskie	116783	67621	60,05	84,43	155697	52409	-103288
Kielecki	Kieleckie	219297	Kielce	kielecki	świętokrzyskie	świętokrzyskie	202196	205500	87,84	72,3	294748	192640	-102108
Inowrocławski	Pomorskie	48599	aleksandrowski	inowrocławski	kujawsko-pomorskie	kujawsko-pomorskie	56259	165207	84,06	62,67	138798	40851	-97947
Morski	Pomorskie	91285	pucki	wejherowski	pomorskie	pomorskie	79347	199564	92,04	64,48	179840	84019	-95821
Grójecki	Warszawskie	132357	piaseczyński	grójecki	mazowieckie	mazowieckie	163879	98561	90,71	81,92	214995	120059	-94936
Zywiecki	Krakowskie	130949	żywiecki	bielski	śląskie	śląskie	152932	157327	86,66	66,25	205560	113475	-92085
Bialski	Krakowskie	116266	oświęcimski	bielski	małopolskie	śląskie	154998	157327	85,08	60,1	187713	98918	-88795
Grudziądzki	Pomorskie	42801	Grudziądz	grudziądzki	kujawsko-pomorskie	kujawsko-pomorskie	98725	40158	93,01	91,36	126890	39810	-87080
Lubelski	Lubelskie	53621	łęczyński	lubelski	lubelskie	lubelskie	57671	146666	72,65	60,21	123023	38953	-84070
Janowski	Lubelskie	87546	stalowowolski	kraśnicki	podkarpackie	lubelskie	109760	99823	60,77	65,02	136269	53205	-83064
Łaski	Łódzkie	97679	pabianicki	łaski	łódzkie	łódzkie	119930	50859	69,31	87,56	149551	67703	-81848
Nowotarski	Krakowskie	129489	nowotarski	suski	małopolskie	małopolskie	187982	83829	74,43	65,53	178124	96378	-81746
Myślenicki	Krakowskie	102692	wielicki	myślenicki	małopolskie	małopolskie	113981	121797	71,82	65,3	153969	73755	-80214
Zawierciński	Kieleckie	131073	myszkowski	będziński	śląskie	śląskie	72084	152133	60,82	67,1	150441	79717	-70724
Dębicki	Krakowskie	110925	ropczycko-sędzisz	dębicki	podkarpackie	podkarpackie	73001	134877	98,33	85,72	178201	110925	-67276
Sochaczewski	Warszawskie	75181	warszawski zachod	sochaczewski	mazowieckie	mazowieckie	107214	85118	78,65	64,14	123353	59133	-64220
Świecki	Pomorskie	87998	tczewski	świecki	pomorskie	kujawsko-pomorskie	115640	99855	93,27	67,48	145419	82071	-63348
Bocheński	Krakowskie	113790	bocheński	wielicki	małopolskie	małopolskie	103357	113981	95,78	79,44	172660	113790	-58870
Tarnobrzeski	lwowskie	73297	Tarnobrzeg	tarnobrzeski	podkarpackie	podkarpackie	48821	54292	61,76	98,04	103113	45271	-57842
Sochaczewski	Warszawskie	75181	grodziski	sochaczewski	mazowieckie	mazowieckie	85623	85118	67,87	63,72	108803	51028	-57775
Janowski	Lubelskie	87546	janowski	kraśnicki	lubelskie	lubelskie	47885	99823	85,4	89,23	131805	74761	-57044
Garwoliński	Lubelskie	159942	garwoliński	otwocki	mazowieckie	mazowieckie	108411	120583	72,42	75,1	171964	115837	-56127
Płocki	Warszawskie	128144	Płock	płocki	mazowieckie	mazowieckie	124553	110140	91,01	71,5	167797	116623	-51174
Pułtuski	Warszawskie	118297	legionowski	pułtuski	mazowieckie	mazowieckie	107072	51287	60,21	75,78	120007	71223	-48784
Myślenicki	Krakowskie	102692	myślenicki	suski	małopolskie	małopolskie	121797	83829	87,11	63,17	129897	89453	-40444
Grójecki	Warszawskie	132357	grójecki	grodziski	mazowieckie	mazowieckie	98561	85623	64,81	67,62	124552	85777	-38775

(0 out of 142 Selected)

Figure 17: Result layer attributes table for the 1:2 variant with an additional column with the difference in population

References

[1] Westra Erik, *Python Geospatial Development*, Packt Publishing Ltd, 2016.

[2] Marian Kallas, *Województwo w Drugiej Rzeczpospolitej (z dziejów podziałów terytorialnych w latach (1919–1939)*. Acta Universitatis Nicolai Copernici. Historia 30 (322), p. 145–161, 1997.

[3] *Dz.U. z 1937, nr 46, poz. 350.*

[4] *Dz.U. z 1998 nr 96 poz. 603*

[5] *Dz.U. z 1998 nr 103 poz. 652*

[6] *Statistical Yearbook of the Republic of Poland*, Warsaw 2019, p. 68.

Some Computational Aspects of Graph-Based Cryptography

Michał Klisowski*

1 Introduction

Modern cryptography tools [6] try to provide security, including data confidentiality and integrity or the ability to verify the source of the information. The traditional tool for data confidentiality is called *symmetric-key cryptography*. In symmetric-key cryptography, both parties must have a common *secret key* that allows both encryption and decryption. A more modern solution is *public-key cryptography*. The receiver can publish information (*public key*) to enable everyone to encrypt information. However, to decrypt the message a secret *private key* is needed.

Asymmetric cryptosystems commonly used today are based on the hardness of just a few computational problems (integer factorization, the discrete logarithm problem). There exist quantum algorithms that can solve these problems once a sufficiently powerful quantum computer is built [21, 20, 31].

Post-quantum cryptography [1] is cryptography based on problems resistant to quantum attacks.

The branch of post-quantum cryptography based on the hardness of the problem of finding a solution of a system of quadratic equations with multiple variables over the finite field $\mathbb{F}_2$ is called *multivariate cryptography* [3]. Algorithms described in this work are closely related to multivariate cryptography but differ from classic multivariate cryptosystems in that they use a variety of commutative rings (not only $\mathbb{F}_2$), and use not only quadratic polynomial maps, but also maps of greater degrees.

This paper discusses some computational aspects of multivariate cryptography algorithms based on two families of algebraic graphs, namely $\mathbb{D}(n, K)$ and $\mathbb{A}(n, K)$.

The first studies on the described families were conducted by Ustimenko, Lazebnik and Woldar [14, 15, 16, 17, 13]. These works studied the $\mathbb{D}(n, K)$ graph families. They turned out to be the source of cryptographic algorithms and error correction codes [5, 7, 19]. We describe these families of graphs in Section 2.

A symmetric cryptosystem based on these families of graphs was first proposed in [26]. In papers [27] and [28] an asymmetric encryption scheme was presented, in which the public key consisted of polynomials of many variables constructed

*Corresponding author — `mklisow@hektor.umcs.lublin.pl` using polynomial equations describing the above-mentioned families of graphs. A graph-based symmetric key scheme is presented in Section 3.

Section 4 describes the basic variant of the asymmetric cryptosystem and presents some useful implementation techniques. It also presents cryptanalysis of the basic variant and informs about its modification options and safer graph-based cryptosystem families.

The paper ends with the conclusion.

2 Algebraic graphs over finite commutative rings

In this section, we describe the mathematical foundations of the algorithms described in the following subsections. These are *finite commutative rings* and algebraically defined families of graphs over these rings. We discuss some of their mathematical properties as well as some aspects of their software implementation.

2.1 Finite commutative rings

We describe commutative finite rings used in the implementation of presented cryptosystems and discuss some aspects of their software representation and arithmetic implementation. The chosen rings are: *the ring of integers modulo p* ($\mathbb{Z}_p$), *finite fields* ($\mathbb{F}_q$ — q is the number of elements of the field), and *Boolean rings* ($\mathbb{B}_n$ — Boolean ring with 2^n elements).

2.1.1 Ring of integers modulo p

The ring of integers modulo p represents modular arithmetic. It is a set of all congruence classes for a modulus p. The modulus is usually known from the context so we will denote the elements of the ring as the smallest non-negative elements of the class. E.g. for the ring of integers modulo 3, the ring would be written simply as $\mathbb{Z}_3 = \{0, 1, 2\}$. The implementation of arithmetic in this ring is simple: most programming languages provide the modulus operator that yields the remainder of an integer division, e.g. `%` operator in C and C++. The ring of integers modulo 2^n is a special, even easier case. All integer arithmetic in the processor is performed modulo 2^n where n is the number of bits of the integer.

2.1.2 Finite fields

The *field* is a commutative ring in which the neutral element of the addition is different from the neutral element of multiplication and every non-zero element is invertible (have multiplicative inverse).

The *finite field* ($\mathbb{F}_p$) is a field that contains a finite number of elements (p). It is called a *prime field* if p is a prime number. The field $\mathbb{F}_p$ is then identical to the ring $\mathbb{Z}_p$.

Let $\mathbb{F}_p$ be the prime field and $m \in \mathbb{F}_p[u]$ be an irreducible polynomial of the variable u over the $\mathbb{F}_p$ of degree n. By $\mathbb{F}_p[u]/(m)$ we denote the set of classes of residuals of polynomials from $\mathbb{F}_p[u]$ modulo m. This set, along with the operation of adding polynomials and the operation of multiplying them modulo polynomial

m, creates a field of size p^n. This field is called an *extension field* of field $\mathbb{F}_p$ of degree n. Every finite field is a prime field or an extension of a prime field of finite degree.

In the following sections, the elements of $\mathbb{F}_q$ will be written as non-negative integers less than q. For a prime field, the notation will be the same as for the ring $\mathbb{Z}_q$. For extension fields, we will write the polynomial (field element) as an integer, the digits of which in base p are the successive coefficients of the polynomial in the order from the highest power. E.g for the field $\mathbb{F}_{3^4}$ the polynomial $2\,u^3 + u$ will be written as $2010_3 = 57$.

The prime field arithmetic is modular arithmetic. Extension field arithmetic is polynomial modular arithmetic. To compute the inverse, we can use the fact that $x^{q-1} = 1$ (so $x \cdot x^{q-2} = 1$ and $x^{-1} = x^{q-2}$) for every $x \in \mathbb{F}_q$, $x \neq 0$ (see [18]) and use the fast exponentiation algorithm.

For fields of a size such that we can afford to store the array of field elements of size $\mathcal{O}\left(q\right)$ in memory, we can speed up the multiplication and division by using the fact that there exists the field element g such that we can represent each non-zero element x as $x = g^k$ for some non-negative integer $k < q - 1$. The number k is called the *discrete logarithm* of x with base g.

We can precompute every pair (x, k), where $k = \log_g(x)$, and store it in a way that allows one to quickly search for both x and k (e.g two arrays — one indexed with x and one with k).

For the fast multiplication one can use:

$$ab = g^{log_g(ab)} = g^{log_g(a)+log_g(b)},$$

and for the division:

$$a/b = g^{log_g(a/b)} = g^{log_g(a)-log_g(b)}.$$

2.1.3 Boolean rings

A Boolean ring is a ring where $x^2 = x$ for every x. We can represent the Boolean ring with 2^n elements as a sequence of n bits. The addition in the ring can be implemented as a bitwise XOR operation (`^` in C and C++) and the multiplication as a bitwise AND (`&` in C and C++).

2.2 $\mathbb{D}(n, K)$ and $\mathbb{A}(n, K)$ graph families

Algebraic graph theory [2] is a branch of graph theory in which algebraic methods are used to define graphs and derive their properties.

Let $B = K[x_1, \ldots, x_n]$ be the ring of polynomials of n variables over the commutative ring K. An *algebraic set* is the set of solutions of a system of polynomial equations over K i.e. it is a subset P of K^n such that there exists the set of polynomials F, $F \subset B$ such that

$$P = \{x \in K^n \colon f(x) = 0, \text{ for every } f \in F\}.$$

An *algebraic graph* is a graph in which the set of vertices and the set of neighbors of each vertex is an algebraic set. $\mathbb{D}(n, K)$ and $\mathbb{A}(n, K)$ graph families are examples of algebraic graphs.

First, let us define the $\mathbb{D}(n, K)$ graph family (introduced in [15] and [17]). Let P and L be two copies of the set of n-tuples of elements of K. The $\mathbb{D}(n, K)$ graph is bipartite (its vertex set consists of two disjoint sets; there is no edge between two vertices of the same part) and the sets P and L are the two parts of it. We denote the elements of the set P by (p), the elements of the set L by [l], and we will write their coordinates as follows:

$$\begin{aligned}(p) &= (p_0, p_1, p_2, \ldots, p_{n-1}, p_n)\\ [l] &= [l_0, l_1, l_2, \ldots, l_{n-1}, l_n].\end{aligned}$$

There is an edge in the graph between (p) and $[l]$ if the following equations are true:

$$\begin{aligned}
&l_1 - p_1 = l_0 p_0\\
&l_2 - p_2 = l_1 p_0\\
&\text{and for } i > 2:\\
&l_i - p_i = l_0 p_{i-2}, \quad \text{for } i \equiv 0 \vee i \equiv 3 \pmod 4\\
&l_i - p_i = l_{i-2} p_0, \quad \text{for } i \equiv 1 \vee i \equiv 2 \pmod 4.
\end{aligned} \tag{1}$$

The $\mathbb{A}(n, K)$ graph family was introduced in [23]. Its definition is similar to the definition of $\mathbb{D}(n, K)$ but the equations differ:

$$\begin{aligned}
l_i - p_i &= l_0 p_{i-1}, \quad \text{for } i \equiv 1 \pmod 2,\\
l_i - p_i &= l_{i-1} p_0, \quad \text{for } i \equiv 0 \pmod 2.
\end{aligned} \tag{2}$$

Each vertex v of the described graphs has an assigned color. We denote it with color(v). The color of the vertex is its first coordinate (an element of K). Each vertex has exactly one neighbor with the color k for any $k \in K$. It results from the equations defining the graph.

We define an operator $N_t(v)$ as a function returning the neighbor of the vertex v with the color color$(v) + t$. This operator is an essential part of the graph-based algorithm described later.

2.2.1 Software representation of algebraic graphs

The order (number of vertices) of the graphs from $\mathbb{D}(n, K)$ and $\mathbb{A}(n, K)$ families is $2|K|^n$ and their size (number of edges) is $|K|^{n+1}$. This means that we cannot use standard techniques for representing graphs in computer memory (adjacency list, adjacency matrix, or incidence matrix).

During the operation of the algorithms described in the following subsections, we usually need only two adjacent graph vertices in memory at a time. While we know the coordinates of one of them, the coordinates of the other are calculated from the first one with the N_t operator. Therefore, we do not store edge information in memory at all.

Algorithms describing the N_t operator for $\mathbb{D}(n, K)$ and $\mathbb{A}(n, K)$ graph families are presented as Algorithm 2.1 and Algorithm 2.2 respectively.

Algorithm 2.1 N_t operator — $\mathbb{D}(n, K)$ graph family

```
Input: $t \in K$, $v$ — vertex,
Output: $u = N_t(v)$
 1: $u_0 := v_0 + t$
 2: if $v \in P$ then
 3:     $u_1 := v_1 + u_0 \cdot v_0$
 4:     $u_2 := v_2 + u_1 \cdot v_0$
 5:     $r := 2$        $\{r \equiv i - 1 \pmod 4$ in the first iteration$\}$
 6:     for $i = 3, 4, \ldots, n - 1$ do
 7:         if $r < 2$        $\{i \equiv 1 \vee i \equiv 2 \pmod 4\}$ then
 8:             $u_i := v_i + v_0 \cdot u_{i-2}$
 9:         else $\{i \equiv 0 \vee i \equiv 3 \pmod 4\}$
10:             $u_i := v_i + v_{i-2} \cdot u_0$
11:         if $r = 3$ then
12:             $r := 0$
13:         else
14:             $r := r + 1$
15: else $\{v \in L\}$
16:     $u_1 := v_1 - u_0 \cdot v_0$
17:     $u_2 := v_2 - u_0 \cdot v_1$
18:     $r := 2$        $\{r \equiv i - 1 \pmod 4$ in the first iteration$\}$
19:     for $i = 3, 4, \ldots, n - 1$ do
20:         if $r < 2$        $\{i \equiv 1 \vee i \equiv 2 \pmod 4\}$ then
21:             $u_i := v_i - v_{i-2} \cdot u_0$
22:         else $\{i \equiv 0 \vee i \equiv 3 \pmod 4\}$
23:             $u_i := v_i - v_0 \cdot u_{i-2}$
24:         if $r = 3$ then
25:             $r := 0$
26:         else
27:             $r := r + 1$
```

3 Symmetric graph-based encryption scheme

Ustimenko suggested in [26] that the properties of the families of graphs of large girth can make them good candidates for a cryptographic tool. He proposed a simple symmetric-key cipher. The described scheme was called a “stream cipher” because it could be used to encrypt messages of any length regardless of key length, however, its mode of operation was completely different from the classic stream ciphers based on the pseudorandom number generator and the XOR operation.

Some aspects of the implementation of that early scheme were described in [22, 29, 25].

In this work we focus on asymmetric schemes and the symmetric cryptosystem presented here is not secure at all, but the algorithms of asymmetric schemes are derived from the symmetric ones. Therefore, it is important to describe them in detail.

Algorithm 2.2 N_t operator — $\mathbb{A}(n, K)$ graph family)

```
Input: $t \in K$, $v$ — vertex,
Output: $u = N_t(v)$
1:  $u_0 := v_0 + t$
2:  if $v \in P$ then
3:      $r := 1$      $\{r \equiv i \bmod 2$   in the first iteration$\}$
4:      for $i = 1, \ldots, n-1$ do
5:          if $r = 0$ then
6:              $u_i := v_i + v_0 \cdot u_{i-1}$
7:          else $\{r = 1\}$
8:              $u_i := v_i + v_{i-1} \cdot u_0$
9:          $r := 1 - r$
10: else $\{v \in L\}$
11:     $r := 1$      $\{r \equiv i \bmod 2$   in the first iteration$\}$
12:     for $i = 1, \ldots, n-1$ do
13:         if $r = 0$ then
14:             $u_i := v_i - v_{i-1} \cdot u_0$
15:         else $\{r = 1\}$
16:             $u_i := v_i - v_0 \cdot u_{i-1}$
17:         $r := 1 - r$
```

The idea behind graph-based encryption algorithms used in the cryptosystem described below is to use the graph vertices (n-tuples over commutative ring K) as plaintexts and ciphertexts and the graph edges as the steps of the encryption algorithm. After starting from the vertex representing the plaintext and following a certain path (depending on the encryption key), the encryption algorithm reaches the vertex representing the ciphertext. Decryption takes the same path in opposite direction. The key in this scheme is also some tuple over the ring K and it determines the edges of an encryption path.

For the key $k = (k_1, k_2, \ldots, k_s)$ the algorithm uses the path:

$$p = v_0 \longrightarrow v_1 = N_{k_1}(v_0) \longrightarrow v_2 = N_{k_2}(v_1) \longrightarrow \cdots$$
$$\cdots \longrightarrow v_s = N_{k_s}(v_{s-1}) = c$$

(p is the plaintext and c is the ciphertext) and the colors of visited vertices are:

$$\operatorname{color}(p) = x_0 \longrightarrow x_1 = x_0 + k_1 \longrightarrow x_2 = x_1 + k_2 \longrightarrow \cdots$$
$$\cdots \longrightarrow x_s = x_{s-1} + k_s = \operatorname{color}(c)$$

An N_t operator is a bijection for every t and its inverse function is $N_t^{-1} = N_{-t}$ (the color of the vertex $N_{-t}(N_t(v))$ is $\operatorname{color}(v) + t + (-t) = \operatorname{color}(v)$). Thus, the decryption procedure is the same as encryption but with different key (decryption key): $k' = (-k_s, -k_{s-1}, \ldots, -k_1)$.

An example of the encryption and decryption procedure for the graph $\mathbb{D}(7, \mathbb{Z}_{256})$ will now be given. Plain text, successive graph vertices, and ciphertext will be shown as column vectors. All addition, subtraction, and multiplication operations are corresponding operations on the ring $\mathbb{Z}_{256}$ (the results are residues modulo 256).

Let (54, 69, 244, 167) be an example password. The encryption steps are shown below (m is the plaintext and c the ciphertext):

$$m = \begin{bmatrix} 11 \\ 138 \\ 225 \\ 126 \\ 236 \\ 18 \\ 165 \end{bmatrix} \longrightarrow \begin{bmatrix} 65 = 11 + 54 \\ 85 = 138 + 65 \cdot 11 \\ 136 = 225 + 85 \cdot 11 \\ 136 = 126 + 65 \cdot 138 \\ 13 = 236 + 65 \cdot 225 \\ 234 = 18 + 136 \cdot 11 \\ 52 = 165 + 13 \cdot 11 \end{bmatrix} \longrightarrow \begin{bmatrix} 134 = 65 + 69 \\ 79 = 85 - 65 \cdot 134 \\ 10 = 136 - 85 \cdot 134 \\ 121 = 136 - 65 \cdot 79 \\ 131 = 13 - 65 \cdot 10 \\ 186 = 234 - 136 \cdot 134 \\ 102 = 52 - 13 \cdot 134 \end{bmatrix} \longrightarrow$$

$$\longrightarrow \begin{bmatrix} 122 = 134 + 244 \\ 43 = 79 + 122 \cdot 134 \\ 140 = 10 + 43 \cdot 134 \\ 31 = 121 + 122 \cdot 79 \\ 71 = 131 + 122 \cdot 10 \\ 244 = 186 + 31 \cdot 134 \\ 144 = 102 + 71 \cdot 134 \end{bmatrix} \longrightarrow \begin{bmatrix} 33 = 122 + 167 \\ 113 = 43 - 122 \cdot 33 \\ 1 = 140 - 43 \cdot 33 \\ 69 = 31 - 122 \cdot 113 \\ 205 = 71 - 122 \cdot 1 \\ 245 = 244 - 31 \cdot 33 \\ 105 = 144 - 71 \cdot 33 \end{bmatrix} = c$$

Below, it is shown how to recover the plaintext from the ciphertext calculated above. The decryption key will be $(-167, -244, -69, -54) = (89, 12, 187, 202)$.

$$c = \begin{bmatrix} 33 \\ 113 \\ 1 \\ 69 \\ 205 \\ 245 \\ 105 \end{bmatrix} \longrightarrow \begin{bmatrix} 122 = 33 + 89 \\ 43 = 113 + 122 \cdot 33 \\ 140 = 1 + 43 \cdot 33 \\ 31 = 69 + 122 \cdot 113 \\ 71 = 205 + 122 \cdot 1 \\ 244 = 245 + 31 \cdot 33 \\ 144 = 105 + 71 \cdot 33 \end{bmatrix} \longrightarrow \begin{bmatrix} 134 = 122 + 12 \\ 79 = 43 - 122 \cdot 134 \\ 10 = 140 - 43 \cdot 134 \\ 121 = 31 - 122 \cdot 79 \\ 131 = 71 - 122 \cdot 10 \\ 186 = 244 - 31 \cdot 134 \\ 102 = 144 - 71 \cdot 134 \end{bmatrix} \longrightarrow$$

$$\longrightarrow \begin{bmatrix} 65 = 134 + 187 \\ 85 = 79 + 65 \cdot 134 \\ 136 = 10 + 85 \cdot 134 \\ 136 = 121 + 65 \cdot 79 \\ 13 = 131 + 65 \cdot 10 \\ 234 = 186 + 136 \cdot 134 \\ 52 = 102 + 13 \cdot 134 \end{bmatrix} \longrightarrow \begin{bmatrix} 11 = 65 + 202 \\ 138 = 85 - 65 \cdot 11 \\ 225 = 136 - 85 \cdot 11 \\ 126 = 136 - 65 \cdot 138 \\ 236 = 13 - 65 \cdot 225 \\ 18 = 234 - 136 \cdot 11 \\ 165 = 52 - 13 \cdot 11 \end{bmatrix} = m.$$

4 Asymmetric graph-based encryption schemes

The idea of using the presented graph families to construct public-key cryptosystems was presented in [27] and [28]. Various aspects of its implementation are presented in papers [10], [9], [11] and [12].

To describe the relationship between the ciphertext $c = (c_1, c_2, \dots, c_n)$, and the plaintext $p = (p_1, p_2, \dots, p_n)$, we use a polynomial transformation:

$$\begin{cases} c_1 = f_1(p_1, p_2, \dots, p_n) \\ c_2 = f_2(p_1, p_2, \dots, p_n) \\ \vdots \\ c_n = f_n(p_1, p_2, \dots, p_n) \end{cases} \tag{3}$$

The polynomial transformation E, where $E(p) = (f_1(p), f_2(p), \dots, f_n(p))$ is a public key. Encryption is done by computing the values of the polynomials f_1, f_2, $\dots$, f_n. The private (decryption) key is of the same form as in the symmetric cryptosystem (tuple of ring elements determining the color of successive vertices on the decryption path).

The branch of cryptography dealing with cryptosystems in which the public key is a polynomial transformation of multiple variables is called multivariate cryptography [3]. Our cryptosystem differs from the classic multivariate cryptosystems in that we also consider transformations of degree greater than 2 and commutative rings other than $\mathbb{F}_2$.

4.1 Basic asymmetric schemes and their algebraic cryptanalysis

Note that the N_t described in the 3 section jest is a polynomial transformation $K^n \to K^n$ (n is the plaintext and ciphertext length). It means that the transformation F_k (where $k = (k_1, k_2, \dots, k_s)$) representing the walk in the graph, defined as

$$F_k = N_{p_s} \circ \dots \circ N_{k_2} \circ N_{k_1} \tag{4}$$

is also a polynomial transformation as a composition of polynomial transformations.

It was proved in [30] that the polynomial transformation F_k, resulting from the composition of N_{k_i} operators for the $\mathbb{D}(n, K)$ graph family has a degree not greater than 3. The proof was extended in [24] to the $\mathbb{A}(n, K)$ graph family.

An example of such a polynomial transformation will now be given. We will use $\mathbb{D}(5, \mathbb{F}_{7^2})$ as an example graph. The ring $\mathbb{F}_{7^2}$ elements will be written as integers in the range $[0, 2^7 - 1]$ as described in the 2.1 subsection, (the irreducible polynomial $w(u) = u^2 + 6u + 3$ is used). The following examples use $\alpha = (28, 20, 19, 28)$ as the password. The relationship between the plaintext $x = (x_1, x_2, x_3, x_4, x_5)$, and the ciphertext $y = (y_1, y_2, y_3, y_4, y_5)$ can then be written as equations:

$$\begin{aligned} y_1 &= x_1 + 39, \\ y_2 &= 17\,x_1 + x_2 + 7, \\ y_3 &= 17\,x_1^2 + 7\,x_1 + 17\,x_2 + x_3 + 3, \\ y_4 &= 39\,x_1^2 + 35\,x_1 + x_4 + 34, \\ y_5 &= 39\,x_1^3 + 35\,x_1^2 + 39\,x_1x_2 + 31\,x_1 + 42\,x_2 + x_5 + 30. \end{aligned} \tag{5}$$

By encrypting the plaintext $x = (2,\ 34,\ 47,\ 15,\ 0)$ using the equation (5), we get

$$y = \begin{bmatrix} 2+39 \\ 17\cdot 2+34+7 \\ 17\cdot 2^2+7\cdot 2+17\cdot 34+47+3 \\ 39\cdot 2^2+35\cdot 2+15+34 \\ 39\cdot 2^3+35\cdot 2^2+39\cdot 2\cdot 34+31\cdot 2+42\cdot 34+0+30 \end{bmatrix} = \begin{bmatrix} 41 \\ 19 \\ 42 \\ 9 \\ 36 \end{bmatrix}.$$

By decrypting with the password $(-28,\ -19,\ -20,\ -28) = (21,\ 37,\ 36,\ 21)$, we get

$$y = \begin{bmatrix} 41 \\ 19 \\ 42 \\ 9 \\ 36 \end{bmatrix} \longrightarrow \begin{bmatrix} 13 = 41+21 \\ 12 = 19+13\cdot 41 \\ 1 = 42+12\cdot 41 \\ 47 = 9+13\cdot 19 \\ 39 = 36+13\cdot 42 \end{bmatrix} \longrightarrow \begin{bmatrix} 43 = 13+37 \\ 3 = 12-13\cdot 43 \\ 35 = 1-12\cdot 43 \\ 22 = 47-13\cdot 3 \\ 40 = 39-13\cdot 35 \end{bmatrix} \longrightarrow$$

$$\longrightarrow \begin{bmatrix} 30 = 43+36 \\ 38 = 3+30\cdot 43 \\ 18 = 35+38\cdot 43 \\ 7 = 22+30\cdot 3 \\ 1 = 40+30\cdot 35 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 = 30+21 \\ 34 = 38-30\cdot 2 \\ 47 = 18-38\cdot 2 \\ 15 = 7-30\cdot 34 \\ 0 = 1-30\cdot 47 \end{bmatrix} = x.$$

It can be proved from equations 1 and 2 that the equation $y = F_k(x)$, where F_k is in the form (4), $y = (y_1,\ \dots,\ y_n)$, $x = (x_1,\ \dots,\ x_n)$, can be written as

$$\begin{cases} y_1 = f_1 + x_1 \\ y_2 = f_2(x_1) + x_2 \\ \vdots \\ y_i = f_i(x_1, x_2, \dots, x_{i-1}) + x_i \\ \vdots \\ y_n = f_n(x_1, x_2, \dots, x_{n-1}) + x_n \end{cases} \tag{6}$$

where f_i is for $2 \le i \le n$ a polynomial of $i-1$ variables, and f_1 is a constant.

This means that such a transformation cannot be directly used as a public key in our cryptosystem. It is possible, having the given ciphertext c and the publicly available f_i polynomials to easily calculate the successive elements of the plaintext p as $p_i = c_i - f_i(p_1, \dots, p_{i-1})$.

Multivariate cryptography uses two invertible affine transformations to hide such polynomial map with a clear, visible structure.

The resulting transformation is therefore of the form:

$$E_{(T_1,k,T_2)} = T_2 \circ F_k \circ T_1 \tag{7}$$

where k is the key, and T_1 and T_2 are invertible affine transformations.

The polynomial map $E_{(T_1,k,T_2)}$ has degree 3 (the composition of F_k of degree 3 with T_1 i T_2 of degree 1). The private key (used for decryption) is a triple $(T_1,\ k,\ T_2)$.

The following example shows a polynomial map resulting from composing affine transformations with a graph-based polynomial map. The graph and password used are the same as in the example above. The affine transformations used are:

$$T_1\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} 14 & 37 & 29 & 1 & 2 \\ 45 & 3 & 15 & 20 & 48 \\ 25 & 17 & 24 & 27 & 12 \\ 8 & 13 & 14 & 20 & 5 \\ 34 & 16 & 46 & 14 & 24 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 17 \\ 21 \\ 44 \\ 3 \\ 17 \end{bmatrix},$$

$$T_2\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} 24 & 5 & 32 & 2 & 33 \\ 38 & 47 & 45 & 19 & 30 \\ 47 & 26 & 18 & 25 & 3 \\ 26 & 42 & 47 & 19 & 22 \\ 27 & 43 & 28 & 23 & 18 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 43 \\ 39 \\ 31 \\ 25 \\ 1 \end{bmatrix}.$$

The resulting transformation is shown below.

$$\begin{aligned}
y_1 = {} & 45\,x_1^3 + 5\,x_1^2x_2 + 25\,x_1^2x_3 + 14\,x_1^2x_4 + 28\,x_1^2x_5 + 27\,x_1x_2^2 + 24\,x_1x_2x_3 + 25\,x_1x_2x_4 \\
& + 43\,x_1x_2x_5 + 23\,x_1x_3^2 + 9\,x_1x_3x_4 + 18\,x_1x_3x_5 + x_1x_4^2 + 4\,x_1x_4x_5 + 4\,x_1x_5^2 + 23\,x_2^3 \\
& + 39\,x_2^2x_3 + 48\,x_2^2x_4 + 40\,x_2^2x_5 + 11\,x_2x_3^2 + 23\,x_2x_3x_4 + 46\,x_2x_3x_5 + 18\,x_2x_4^2 \\
& + 9\,x_2x_4x_5 + 9\,x_2x_5^2 + 35\,x_3^3 + 17\,x_3^2x_4 + 34\,x_3^2x_5 + 8\,x_3x_4^2 + 32\,x_3x_4x_5 + 32\,x_3x_5^2 \\
& + 37\,x_4^3 + 19\,x_4^2x_5 + 31\,x_4x_5^2 + 37\,x_5^3 + 20\,x_1^2 + 47\,x_1x_2 + 31\,x_1x_3 + 6\,x_1x_4 + 24\,x_1x_5 \\
& + 31\,x_2^2 + 32\,x_2x_3 + x_2x_4 + 4\,x_2x_5 + 35\,x_3^2 + 16\,x_3x_4 + 7\,x_3x_5 + 28\,x_4^2 + 35\,x_5^2 + x_1 \\
& + 16\,x_2 + 36\,x_3 + 16\,x_4 + 15\,x_5 + 4 \\
y_2 = {} & 5\,x_1^3 + 18\,x_1^2x_2 + 8\,x_1^2x_3 + 13\,x_1^2x_4 + 19\,x_1^2x_5 + 46\,x_1x_2^2 + 17\,x_1x_2x_3 + 8\,x_1x_2x_4 \\
& + 16\,x_1x_2x_5 + 26\,x_1x_3^2 + 20\,x_1x_3x_4 + 33\,x_1x_3x_5 + 47\,x_1x_4^2 + 27\,x_1x_4x_5 + 27\,x_1x_5^2 \\
& + 26\,x_2^3 + 14\,x_2^2x_3 + 34\,x_2^2x_4 + 12\,x_2^2x_5 + 2\,x_2x_3^2 + 26\,x_2x_3x_4 + 45\,x_2x_3x_5 + 33\,x_2x_4^2 \\
& + 20\,x_2x_4x_5 + 20\,x_2x_5^2 + 43\,x_3^3 + 35\,x_3^2x_4 + 21\,x_3^2x_5 + 22\,x_3x_4^2 + 39\,x_3x_4x_5 + 39\,x_3x_5^2 \\
& + 32\,x_4^3 + 24\,x_4^2x_5 + 48\,x_4x_5^2 + 32\,x_5^3 + 26\,x_1^2 + 39\,x_2^2 + 41\,x_2x_4 + 8\,x_2x_5 + 40\,x_3^2 \\
& + 14\,x_4^2 + 35\,x_1x_2 + 29\,x_1x_3 + 16\,x_1x_4 + 31\,x_1x_5 + 31\,x_2x_3 + 13\,x_3x_4 + 11\,x_3x_5 \\
& + x_4x_5 + 44\,x_5^2 + 21\,x_1 + 7\,x_2 + 23\,x_3 + 30\,x_4 + 38\,x_5 + 10 \\
y_3 = {} & 47\,x_1^3 + 36\,x_1^2x_2 + 17\,x_1^2x_3 + 8\,x_1^2x_4 + 16\,x_1^2x_5 + 30\,x_1x_2^2 + 7\,x_1x_2x_3 + 17\,x_1x_2x_4 \\
& + 34\,x_1x_2x_5 + 4\,x_1x_3^2 + 45\,x_1x_3x_4 + 41\,x_1x_3x_5 + 10\,x_1x_4^2 + 33\,x_1x_4x_5 + 33\,x_1x_5^2 \\
& + 4\,x_2^3 + 25\,x_2^2x_3 + 14\,x_2^2x_4 + 28\,x_2^2x_5 + 9\,x_2x_3^2 + 4\,x_2x_3x_4 + x_2x_3x_5 + 41\,x_2x_4^2 \\
& + 45\,x_2x_4x_5 + 45\,x_2x_5^2 + 48\,x_3^3 + 31\,x_3^2x_4 + 13\,x_3^2x_5 + 35\,x_3x_4^2 + 42\,x_3x_4x_5 + 42\,x_3x_5^2 \\
& + 12\,x_4^3 + 44\,x_4^2x_5 + 39\,x_4x_5^2 + 12\,x_5^3 + 19\,x_1^2 + 31\,x_2^2 + 39\,x_2x_4 + 35\,x_2x_5 + 9\,x_3^2 \\
& + 22\,x_4^2 + 29\,x_1x_2 + 48\,x_1x_3 + 48\,x_1x_4 + 11\,x_1x_5 + 16\,x_2x_3 + 37\,x_3x_4 + 8\,x_3x_5 \\
& + 31\,x_4x_5 + 23\,x_5^2 + 34\,x_1 + 13\,x_2 + 44\,x_3 + 31\,x_4 + 30\,x_5 + 25
\end{aligned}$$

$$
\begin{aligned}
y_4 = {} & 42\,x_1^3 + 25\,x_1^2x_2 + 9\,x_1^2x_3 + 2\,x_1^2x_4 + 4\,x_1^2x_5 + 40\,x_1x_2^2 + 46\,x_1x_2x_3 + 9\,x_1x_2x_4 \\
& + 18\,x_1x_2x_5 + 34\,x_1x_3^2 + 32\,x_1x_3x_4 + 8\,x_1x_3x_5 + 19\,x_1x_4^2 + 13\,x_1x_4x_5 + 13\,x_1x_5^2 \\
& + 34\,x_2^3 + 11\,x_2^2x_3 + 36\,x_2^2x_4 + 23\,x_2^2x_5 + 7\,x_2x_3^2 + 34\,x_2x_3x_4 + 12\,x_2x_3x_5 + 8\,x_2x_4^2 \\
& + 32\,x_2x_4x_5 + 32\,x_2x_5^2 + 45\,x_3^2x_4 + 5\,x_3^3 + 41\,x_3^2x_5 + 20\,x_3x_4^2 + 10\,x_3x_4x_5 + 10\,x_3x_5^2 \\
& + 29\,x_4^3 + 27\,x_4^2x_5 + 47\,x_4x_5^2 + 29\,x_5^3 + 39\,x_1^2 + 24\,x_1x_2 + 47\,x_1x_3 + 37\,x_1x_4 + 46\,x_1x_5 \\
& + 47\,x_2^2 + 10\,x_2x_3 + 19\,x_2x_4 + 13\,x_2x_5 + 5\,x_3^2 + 33\,x_3x_4 + x_3x_5 + 4\,x_4^2 + 5\,x_5^2 + 13\,x_1 \\
& + 37\,x_2 + 2\,x_3 + 14\,x_4 + 24\,x_5 + 32 \\
y_5 = {} & 46\,x_1^3 + 45\,x_1^2x_2 + 42\,x_1^2x_3 + 39\,x_1^2x_4 + 22\,x_1^2x_5 + 4\,x_1x_2^2 + 19\,x_1x_2x_3 + 42\,x_1x_2x_4 \\
& + 35\,x_1x_2x_5 + 38\,x_1x_3^2 + 6\,x_1x_3x_4 + 5\,x_1x_3x_5 + 30\,x_1x_4^2 + 15\,x_1x_4x_5 + 15\,x_1x_5^2 + 38\,x_2^3 \\
& + 32\,x_2^2x_3 + 31\,x_2^2x_4 + 13\,x_2^2x_5 + 10\,x_2x_3^2 + 38\,x_2x_3x_4 + 27\,x_2x_3x_5 + 5\,x_2x_4^2 + 6\,x_2x_4x_5 \\
& + 6\,x_2x_5^2 + 17\,x_3^3 + 24\,x_3^2x_4 + 48\,x_3^2x_5 + 25\,x_3x_4^2 + 37\,x_3x_4x_5 + 37\,x_3x_5^2 + 21\,x_4^3 + 28\,x_4^2x_5 \\
& + 7\,x_4x_5^2 + 21\,x_5^3 + 29\,x_1^2 + 34\,x_1x_2 + 29\,x_1x_3 + 10\,x_1x_4 + 36\,x_1x_5 + 33\,x_2^2 + 30\,x_2x_3 \\
& + 20\,x_2x_4 + 37\,x_2x_5 + 20\,x_3^2 + 5\,x_3x_4 + 10\,x_3x_5 + 19\,x_4^2 + 23\,x_4x_5 + 40\,x_5^2 + 46\,x_1 \\
& + 20\,x_2 + 3\,x_3 + 34\,x_4 + 21\,x_5 + 35
\end{aligned}
$$

4.1.1 Public key generation

The most interesting and also time-consuming algorithm of the public key cryptosystem is the public key generation algorithm. It consists of two important steps:

- generating two invertible affine transformations over the given rings,
- generating the public polynomial map with symbolic computation.

Some efficient algorithms ($\mathcal{O}\left(n^3\right)$) for generating invertible affine transformations (it comes down to generating invertible matrices) over selected commutative rings are described in [8].

The algorithm for generating a polynomial map is the most important part of the public key generation. It can be done with an N_t operator (Algorithms 2.1 and 2.2). The main difference between this algorithm and the symmetric-key encryption algorithm is that during the symmetric encryption we perform calculations on the elements of the ring, and while we generate the public key, we perform symbolic computations — calculations on polynomials over a given ring.

In the graph $\mathbb{D}(2, \mathbb{Z}_7)$ for the password (5, 4), the encryption of the (2, 3) plaintext would go like this:

$$
\begin{bmatrix} 2 \\ 3 \end{bmatrix} \longrightarrow \begin{bmatrix} 4 = 2 + 2 \\ 4 = 3 + 2 \cdot 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 = 4 + 4 \\ 0 = 4 - 4 \cdot 1 \end{bmatrix}.
$$

Generating a polynomial transform would look like this:

$$
\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \longrightarrow \begin{bmatrix} x1 + 2 \\ x_2 + x_1(x_1 + 2) = x_1^2 + 2x_1 + x_2 \end{bmatrix} \longrightarrow
$$
$$
\longrightarrow \begin{bmatrix} x_1 + 2 + 4 = x_1 + 6 \\ (x_1^2 + 2x_1 + x_2) - (x_1 + 2)(x_1 + 6) = x_1 + x_2 + 2 \end{bmatrix}.
$$

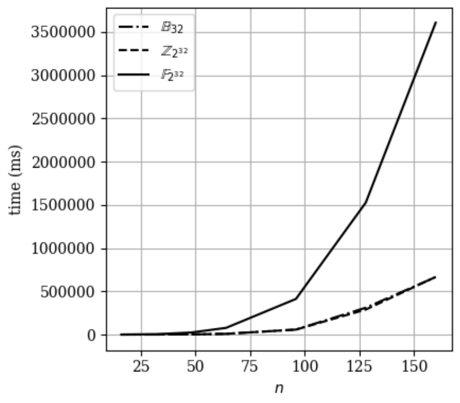
For simplicity, we have omitted affine transformations in the above examples.

A key part of these computations is the multiplication of polynomials. We tested various libraries for polynomial computations over commutative rings (e.g. Victor Shoup's NTL library) and various computer algebra systems for this, but the computation time was not satisfactory. The main problem with existing software was its generality. Although there exist fast implementations of multiplication algorithms, they all do full multiplication. In our case, because we know that the full result is of degree not bigger than 3, we don't have to compute coefficients of monomials with the bigger degree. Therefore, we implemented the library for performing the arithmetic operations on polynomials of degree bounded by 3. It gives us the multiplication algorithm linear with respect to the number of coefficients (but cubical with respect to n — the number of plaintext and ciphertext elements).

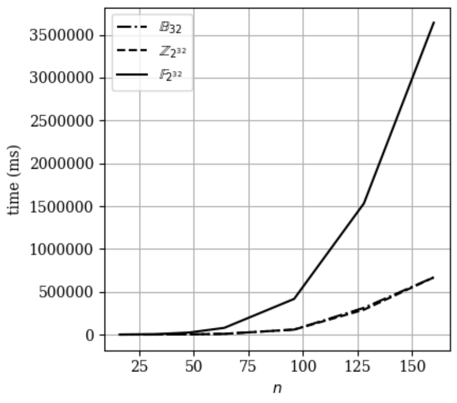
After generating a graph-based polynomial map we must compose it with affine transformation. The composition also must be done symbolically.

It was shown in [8] that the complexity of the public key generation is $\mathcal{O}\left(mn^4\right)$ where m is the length of the key and n is the length of the plaintext and ciphertext.

Figure 1 shows sample generation times for various rings and graphs. The computations were performed on a computer with AMD Athlon II X2 245, 2.9 GHz CPU.



(a) $\mathbb{D}(n, K)$ graph family



(b) $\mathbb{A}(n, K)$ graph family

Figure 1: Generation time of public polynomial map for fixed key length (64) ($K = \mathbb{B}_{32}, \mathbb{Z}_{2^{32}}, \mathbb{F}_{2^{32}}$)

4.1.2 Cryptanalysis

The asymmetric cryptosystem presented in this chapter turned out to be quite easy to break regardless of the affine transformations used. However, public-key algorithms described in this section are still useful in algebraic graph theory research and are building blocks of some newer cryptographic schemes.

Even though having a given public key in the form of a polynomial map:

$$E(x_1, x_2, \ldots, x_n) = \\ (f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n))$$

we cannot find private key in the form of the tuple of ring elements and two affine transformation, we can find its polynomial form E^{-1}, i.e. find the transformation

$$E^{-1}(x_1, x_2, \ldots, x_n) = \\ (g_1(x_1, x_2, \ldots, x_n), g_2(x_1, x_2, \ldots, x_n), \ldots, g_n(x_1, x_2, \ldots, x_n)),$$

such that

$$E^{-1}(E(x)) = x,$$

i.e.

$$g_i(f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n)) = x_i, \\ \text{dla } 1 \leq i \leq n.$$

This is because the decryption transformation in our cryptosystem is a transformation of the same type as the encryption transformation, i.e. its degree is also 3.

We can write the polynomial form of the transformation E^{-1} as n polynomials:

$$y_s = \sum_{\substack{i \leq j \leq k \\ 1 \leq i,j,k \leq n}} a^{(s)}_{ijk} x_i x_j x_k + \sum_{\substack{i \leq j \\ 1 \leq i,j \leq n}} a^{(s)}_{ij} x_i x_j + \sum_{1 \leq i \leq n} a^{(s)}_i x_i + a^{(s)}, \\ \text{dla } 1 \leq s \leq n. \tag{8}$$

Each of them has n monomials (see [8]).

Thus, in order to find the polynomial form, we need to find $nr = \frac{n(n+1)(n+2)(n+3)}{6}$ coefficients of these polynomials.

Using r random plaintexts (denoted by $x^{(1)}, \ldots, x^{(r)}$) and encrypting them with a public transformation we get r pairs $(x^{(i)}, y^{(i)})$, where $y^{(i)}$ is a ciphertext corresponding to the plaintext $x^{(i)}$. If we now substitute the coordinates of the obtained pairs into the equation (8) for the selected s, we will obtain r linear equations with r unknowns. Such a system of equations is easy to solve and by solving it, we obtain the coefficients of the component s of the decrypting transformation.

The same plaintext pairs and ciphertext pairs can be used to obtain the coefficients of each of the n polynomials that make up the decryption transform. So it is enough to generate r of them to find the entire decryption key.

Eventually finding the decryption transform is to solve the following linear equation:

$$XA = Y, \tag{9}$$

czyli

$$A = X^{-1}Y \tag{10}$$

The unknown in this equation is A — the matrix of the decrypting transformation coefficients. Matrices X and Y are matrices constructed from random plaintexts and their corresponding ciphertexts.

4.1.3 Modified public-key cryptosystems

The given cryptanalysis motivated the invention of two new families of cryptosystems with a public key based on the $\mathbb{D}(n, K)$ and $\mathbb{A}(n, K)$ graphs. They are described in detail in [8].

One of them is based on small-degree permutation polynomials of one degree whose inverse permutation has a very high degree. Such polynomials make it possible to construct a small-degree polynomial encrypting transformation that has a corresponding decrypting transformation with an arbitrarily high degree.

Another one uses the fact that the encryption map and its composition with itself form the cyclic group. This enables the use of a modified classic ElGamal cryptosystem [4].

5 Conclusion

In this paper, we presented some computational aspects of cryptography based on algebraic graphs. We proposed some algorithms and implementation techniques for these cryptosystems.

The presented techniques enable the efficient implementation of the presented cryptosystems. The presented algorithms can help in further research in graph theory and in constructing new, faster and safer cryptosystems.

A good direction for further research on the described algorithms would be their improvement with the use of HPC (high-performance computing) techniques. The arithmetic of dense polynomials (especially addition and more time-consuming multiplication) involves a lot of array processing that can be vectorized. Generating public keys involves a lot of polynomial composition, which in turn involves a lot of polynomial multiplications that could be done in parallel.

References

[1] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography.* Springer, 2009.

[2] N. Biggs. *Algebraic Graph Theory.* Cambridge University Press, drugie edition, 1993.

[3] J. Ding, J. E. Gower, and D. S. Schmidt. *Multivariate Public Key Cryptosystems.* Advances in Information Security. Springer, 2006.

[4] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[5] L. J. Guinand P. Tanner type codes arising from large girth graphs. In *Canadian Workshop on Information Theory CWIT '97*, 1997.

[6] J. Katz and Y. Lindell. *Introduction to Modern Cryptography: Third Edition.* Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press/Taylor & Francis Group, 2020.

[7] J.-L. Kim, U. Peled, I. Perepelitsa, V. Pless, and S. Friedland. Explicit construction of families of ldpc codes with no 4-cycles. *IEEE Transactions on Information Theory*, 50(10):2378–2388, 2004.

[8] M. Klisowski. *Zwiększenie bezpieczeństwa kryptograficznych algorytmów wielu zmiennych opartych na algebraicznej teorii grafów.* PhD thesis, Politechnika Częstochowska, 2015.

[9] M. Klisowski, U. Romańczuk, and V. Ustimenko. The implementation of cubic public keys based on a new family of algebraic graphs. *Annales UMCS, Informatica*, 11(2):127–141, 2011.

[10] M. Klisowski and V. Ustimenko. On the implementation of public keys algorithms based on algebraic graphs over finite commutative rings. In *IMCSIT*, pages 303–308, 2010.

[11] M. Klisowski and V. Ustimenko. On the implementation of cubic public keys based on algebraic graphs over the finite commutative rings and their symmetries. *Albanian Journal of Mathematics*, 5(3), 2011.

[12] M. Klisowski and V. Ustimenko. On the comparison of cryptographical properties of two different families of graphs with large cycle indicator. *Mathematics in Computer Science*, 6(2):181–198, 2012.

[13] F. Lazebnik, V. Ustimenko, and A. Woldar. A characterization of the components of the graphs D(k,q). *Discrete Mathematics*, 157(1–3):271–283, 1996.

[14] F. Lazebnik and V. A. Ustimenko. New examples of graphs without small cycles and of large size. *Eur. J. Comb.*, 14(5):445–460, Sept. 1993.

[15] F. Lazebnik and V. A. Ustimenko. Some algebraic constructions of dense graphs of large girth and of large size. *DIMACS Series Discrete Math. Theoret. Comput. Sci.*, 10:75–93, 1993.

[16] F. Lazebnik and V. A. Ustimenko. Explicit construction of graphs with an arbitrary large girth and of large size. *Discrete Applied Mathematics*, 60(1-3):275–284, 1995.

[17] F. Lazebnik, V. A. Ustimenko, and A. J. Woldar. A new series of dense graphs of high girth. *Bull. Amer. Math. SIC.*, 32:73–79, 1995.

[18] R. Lidl and H. Niederreiter. *Finite Fields.* Number 20 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1997.

[19] M. Polak and V. Ustimenko. Ldpc codes based on algebraic graphs. *Annales Universitatis Mariae Curie-Skłodowska. Sectio A1. Informatica*, 12, 01 2012.

[20] J. Proos and C. Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information & Computation*, 3(4):317–344, 2003.

[21] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, Oct. 1997.

[22] V. Ustimenko. Cryptim: Graphs as tools for symmetric encryption. In S. Boztaş and I. E. Shparlinski, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 2227 of *Lecture Notes in Computer Science*, pages 278–286. Springer Berlin Heidelberg, 2001.

[23] V. Ustimenko. On linguistic dynamical systems, families of graphs of large girth, and cryptography. *Journal of Mathematical Sciences*, 140(3):461–471, 2007.

[24] V. Ustimenko and U. Romańczuk. On extremal graph theory, explicit algebraic constructions of extremal graphs and corresponding turing encryption machines. In *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, pages 257–285. Springer, 2013.

[25] V. Ustimenko and A. Touzene. Cryptall: System to encrypt all types of data. *Notices of Kiev Mohyla Academy*, 2004.

[26] V. A. Ustimenko. Coordinatization of regular tree and its quotients. In P. Engel and H. Syta, editors, *Voronoï's Impact on Modern Science*, number 2 in Proceedings of the institute of mathematics of the national academy of sciences of Ukraine. Institute of Mathematics, National Academy of Sciences of Ukraine, 1998.

[27] V. A. Ustimenko. Graphs with special arcs and cryptography. *Acta Applicandae Mathematicae*, 74, 2002.

[28] V. A. Ustimenko. Maximality of affine group, and hidden graph cryptosystems. *Algebra Discrete Math.*, 2005(1):133–150, 2005.

[29] V. A. Ustimenko and Y. M. Khmelevsky. Walks on graphs as symmetric or asymmetric tools to encrypt data. *The South Pacific Journal of Natural and Applied Sciences*, 2002.

[30] A. Wróblewska. On some properties of graph based public keys. *Albanian Journal of Mathematics*, 2(3), 2008.

[31] S. Y. Yan. *Quantum Attacks on Public-Key Cryptosystems*. Springer, 2012.

DNA Based Cryptographic Key Storage System With a Simple and Automated Method of Primers Selection

Marek Miśkiewicz[*]
Adam Kuzdraliński
Damian Rusinek
Bogdan Księżopolski

1 Introduction

Nowadays, a vast amount of information and data requires protection against unauthorized access by third parties. Modern cryptographic techniques closely relate to the physical representation of data and their effectiveness, often based on limited computing power. Encryption is the basics but effective methods of data protection. All encryption algorithms require cryptographic keys. In addition, there are many secret systems where any access to resources requires password authentication. The security of data and systems in this case depends largely on how strong cryptographic keys are (strength of cryptographic keys strongly depends on their length) and how securely they are stored. Of course, it would be the best if one did not store the cryptographic keys at all, which would guarantee the highest level of security. Unfortunately, in most cases (e.g., 4096-bit RSA keys), it is not possible for the user to remember the strong password or the large key in the correct way.

Most users usually store secret keys in hard drives placed in computer devices with access secured with a simpler several-character password or on a portable flash drive. Such solutions have many disadvantages and really do not guarantee a high level of data security or even in some case's usability. There are a few important reasons the cryptographic data is not completely safe, if it is stored on portable devices (in NAND memory chips) or magnetic storage devices. A user who really cares about the security of their data cannot be sure that the data storage devices, produced by third parties, do guarantee proper security. This is because the average user does not have access to the exact device specification and cannot check if the

[*]Corresponding author — `marek.miskiewicz@mail.umcs.pl`

electronic systems controlling the memory chips do not allow easy access to data stored by unauthorized entities. Strong cryptography and ultimate cryptographic keys security require the assumption of complete distrust in the devices that are used. The continuous reduction in the size of integrated circuits leads to increased production costs. This forces the vast majority of chip design companies to trust an external third party in chip fabrication but outsourcing of chip fabrication opens-up hardware to attack. Even if the hardware manufacturers do not have bad intentions, there is always the possibility that third-party interference has occurred (E.g., via supply chain attacks and dopant-level Trojans. See [1], [2] and [3]).

Access to stored cryptographic keys needs at least computer devices and often access to wider resources e.g., Internet. With the current complexity of digital systems, we cannot fully guarantee security, which to some extent relies on trust in the integrity of digital system manufacturers and designers. Absolute security, at least theoretically, can only be guaranteed by the lack of participation of third parties in the process of managing cryptographic keys (creation, storage, use and destruction), and in particular their storage. If we consider it necessary to use electronic devices, this entails additional risks (so-called environmental threats), in particular because of the lack of access to stored data during power system failure caused, for example, by Solar Storm or High-altitude Electromagnetic Pulse [4].

In the article, we present the new cryptographic keys storage system based on DNA strands. The proposed method of key storage in terms of its security attempts to exploit the specific characteristics of DNA itself. The choice of DNA as the carrier allows to limit the threats connected with strong electromagnetic fields, to which DNA structures are not sensitive. The specificity of the carrier causes that the cryptographic keys stored in DNA will exist in a huge number of copies, occupying at the same time a very small volume, which effectively minimizes the problem of incorrect reading of the key, at the same time allowing to "hide" the keys on a molecular scale. In the presented method, it is possible to store encrypted keys (actually, data is stored as strings of bits, regardless of what those bits represent), however, this leads to the problem of providing security for the key used to decrypt the stored keys.

Presenting this innovative key storage method, we mainly focus on the procedure of preparing and selecting the main components of DNA strands, which are needed to carry any useful information in terms of DNA information storage. The contributions to key storage systems of our concept are:

- increased security and usability:
    - no third parties involved in the process of generation and access to keys,
    - lack of weaknesses and vulnerabilities associated with storing cryptographic keys on portable electronic devices,
    - enormous difficulty in accessing keys by unauthorized persons in case of control loss,
    - no DNA sequencing devices required,
    - faster and easier to use in comparing to the previously proposed methods,
    - introduced undetectable transferability in the physical form,

– effectiveness confirmed by an experiment.

Furthermore, in the case of DNA components selection procedure, our contribution is to present a simple method that allows for generating, selecting and composing DNA data strands. We provide computer program based on a presented method for easy use as well.

2 Bio-cryptography

Using DNA strands to store information and even perform simple "calculations" is not a completely new idea. Adleman in his work showed the possibility of using fragments of specially prepared DNA strands to solve the problem of Hamilton's path [5]. Gehani together with others created the basis of DNA-based cryptosystems based on the idea of OneTime Pad [6]. In their work, Y. Zhang, X. Lui and M. Sun showed a practical implementation of the problem of key distribution for the OTP method [7]. The sequence of nucleotides in a randomly selected fragment of DNA is used as the key to encrypt the message. They replaced the explicit text with a sequence of bits and using the XOR function joined with the key string. (The key, based on the DNA sequence, can be created by using one of the possible substitutions of nucleotides presented in Table 1). Next, the "DNA key" was

Table 1: DNA nucleotides substitution scheme

Nucleotide	Binary string
A	00
C	01
T	11
G	10

"glued" to the plasmid and placed in the bacterial cell. The environment inside the bacteria allows them to stably hold the information in the DNA strand, which is very sensitive to changes in the temperature and pH of the solution in which it is located. The stability of the DNA accumulated in bacterial cells carried out in the state of spore is impressive. Scientists were able to read genetic material from Subtilis bacteria, which is millions of years old [8, 9]. Modern laboratory techniques allow for stable storage of synthetic DNA in Silica for thousands of years [10]. This may be important if it is necessary to store relevant in-formation (cryptographic information) for a very long time. Traditional storage technologies, such as magnetic devices and optical discs, are not reliable for long-term data storage. Their estimated life span is equal to about 50 years [11]. Halvorsen and Wong in their paper [12] showed an interesting, simple and secure system for data encryption and decryption using DNA self-assembly and PCR-based decoded information reading method. Tanaka, Okamoto and Saito presented a system for public key distribution based on DNA as a one-way function [13]. Using the methods and algorithms described in the works of A. Leier [14, 15], one can hide the message in a DNA sequence in an encrypted or unencrypted way. Such steganography techniques require active synthesis of deoxyribonucleic acid chains. The text is encrypted directly in the series of A, C, T and G, or special groups are identified later than counterparts of binary zeros and ones. The presented methods require both synthesis and sequencing devices at almost every stage of work with data stored in DNA, which seems to be an inconvenience in a certain class of applications. It applies further some ideas presented in the last two publications. The DNA chain can also be successfully used in forensics [16] and for invisible product tagging [17].

3 The method

In this paper we use the concept presented in the work of [14] and [18], where the single bits of information are represented by groups of nucleotides. With certain restrictions, this solution allows for relatively easy generating sequences of data stored in DNA without the need to use synthesis devices. The process of preparing data containing secret information, for example password or cryptographic keys, without third parties' engagement during the synthesis process significantly increases data security. The change that we propose relates to the mentioned concept is the use of the phenomenon of DNA strand elongation. In Leier's work, the DNA fragments are joined by so-called "sticky ends", with the participation of the Ligase enzyme. As the author himself writes, this process takes a relatively long time (about 20 hours). In addition, it requires the preparation of twice as much DNA material for the construction of fragments encoding bits of information. In our method, we use another enzyme, called DNA polymerase. We can get the full thread in less than an hour. In addition, the costs associated with preparing components for the construction of a thread with recorded bits are 50% lower, which is a significant improvement relating to Leier's method. The method we use is based on the PCR reaction. The oligonucleotides used in the reaction are primers for the DNA polymerase and hybridize at 20 nt sections to form a 380 nt strand of DNA. Reading is also performed using PCR in the classic reaction with two primers, where the forward primer is common to both the bit position reading reaction "0" and "1" and amplify the entire segment. Leier's group uses ligation, which is 100 – 500 times more expensive than our method. In addition, the joining process in our method takes about an hour, while Leier's even over 1 day. Our technology only needs a thermal cycler and electrophoresis apparatus, unlike the Leier's method, where more equipment and reagents are needed.

3.1 DNA storage data structure

DNA strands with data stored within comprise a series of specially prepared components — building blocks. These building blocks in fact are shorter fragments of single-stranded DNA. The general shape of single components is presented in Figure 1. The major part named "content" identifies some sort of data stored as a series of nucleotides. These nucleotides can represent a bit or index of extracting parts. The end regions called "binding sites" are fragments of DNA strand that can easily bind to other complementary parts connected to the other building blocks to produce longer strands as a series of bits. In the simplest case to store data in DNA as a series of 0 and 1 bits, one needs at least four structurally similar fragments (see Figure 2). Two of them named Start and End start and end DNA strands

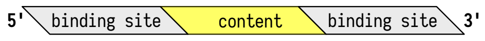


Figure 1: General structure of single component (single-handed DNA fragment)

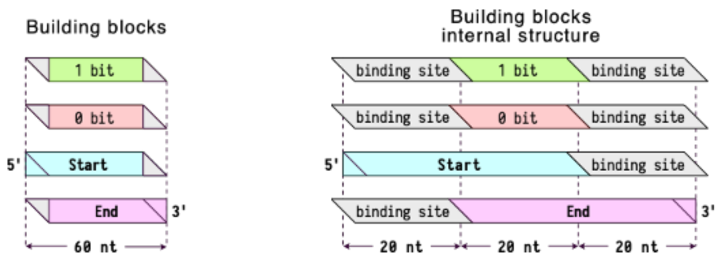


Figure 2: Structural components to build a simple DNA data strand and its detailed structure

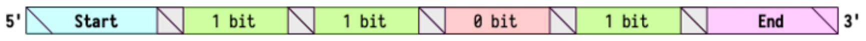


Figure 3: Example of structure of four-bit DNA strand

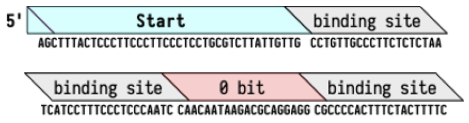

5'
Start
binding site
AGCTTTACTCCCTTCCCTTCCCTCCTGCGTCTTATTGTTG CCTGTTGCCCTTCTCTCTAA
binding site
0 bit
binding site
TCATCCTTTCCCTCCCAATC CAACAATAAGACGCAGGAGG CGCCCCACTTTCTACTTTTC

Figure 4: Example of an arrangement of nucleotides in the 5 'strand of the discussed data strand components

containing data. "0" and "1" fragments represent bits of data. An example of the general structure of a four bits single DNA strand is shown in Figure 3.

3.2 Detailed structure of data strand components

As it was mentioned earlier, individual components for building a DNA strand with data are 20 nucleotide fragments of single-stranded DNA (usually called primers). They are both shown in the pictures as "binding site" and fragments named "Start" and "End". Figure 4 shows an example of the structure of a DNA strand considering the sequence of nucleotides in the mentioned fragments. As it turns out, one of the major problems in creating the individual components is the correct selection of these 20 nucleotide DNA fragments (primers). This issue is so important that much attention was paid to it when designing data storage systems based on DNA strands. The problem is widely discussed in [22].

3.3 Binding sites

So-called "binding sites" are used to assemble a strand of DNA containing the desired bit string. Assembling DNA strands with the use of DNA polymerase requires the existence of specially selected fragments of single-stranded DNA in the reaction environment. Side by side "data" components are assembled using complementary single-stranded DNA fragments. The simplified binding process using "binding sites" and the direction of the elongation process on complementary DNA strands are shown in the Figure 5.

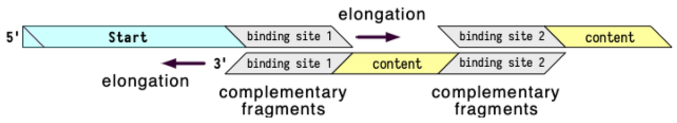


Figure 5: Connection using "binding sites" and the direction of the elongation process on complementary DNA strands

DNA data strand structure looks like this:

$$S - s_0 - B_1 - s_1 - B_2 - s_2 - \ldots - s_{(n-1)} - B_n - s_n - E,$$

where B_n denotes "0" or "1" bit components. s_n are corresponding binding sites. As one can see, n different binding sites are required. Biological limitations related to the procedure of creation data stands from DNA fragment and read them by gel electrophoresis cause that the number of bits carried by DNA strand is not enough to store, for example, a long cryptographic key (1024 to 4096 bits) in a single strand. The reasonable total length of DNA strand that can be used for data storage considered in this paper is about 1000 bp. For such strands the number of stored bits is about 32, so 1024-bit keys require 32 different DNA strands. It is therefore necessary to introduce a system of indexing individual strands or even individual keys if a multi-key system is introduced.

3.4 DNA fragments selection procedure

The procedure for preparing DNA fragments for the described method must consider many aspects related to the process of hybridization and extension of DNA strands. In this process, we are dealing with molecular mechanisms that depend on many factors and thermodynamic parameters. Many of these parameters depend on the structure of the DNA strand itself, so the arrangement of nucleotides cannot be chosen completely arbitrarily. There are many conditions that must be satisfied to prepare DNA fragments with proper nucleotides arrangement, and there are known many approaches to solve the problem of so-called DNA codes.

The procedure used in the presented method to select the proper DNA fragments comprises in general the following steps:

1. Generating and selecting a set of random strings of nucleotides.

2. Providing extended calculations and verification tests.

3. Previewing and final selection of required components.

4. DNA strands structure building.

The general workflow is presented in Figure 6. A detailed description of the presented procedure will be accompanied by a presentation of the components of the computer program in which the procedure has been implemented. Program is written in Python programming language with use of QT5 widget libraries. The main window of the program is presented in Figure 7. Program contains four tabs named: Generator, Formatter, Selector and Composer, which are related to the steps enumerated in general procedure scheme.

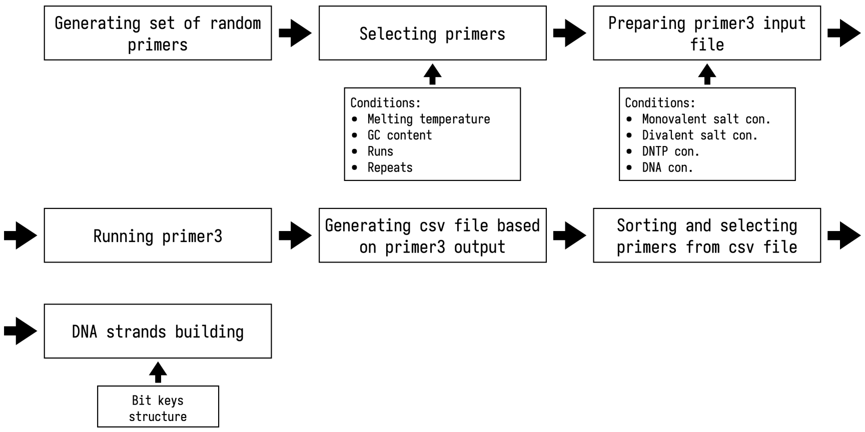


Figure 6: The general workflow for the procedure of generating and selecting primers for DNA data strands

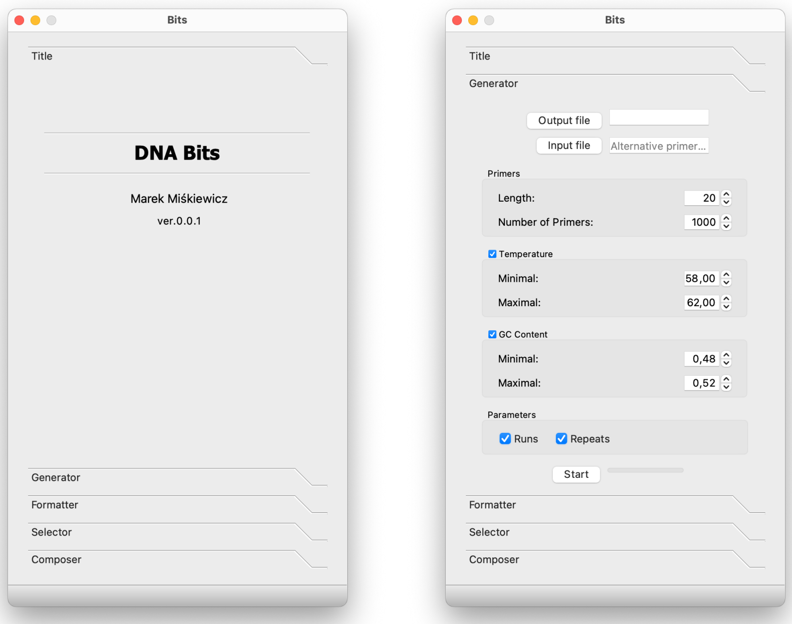
3.4.1 Primers generator

In the first step program allows to generate a set of random strings of nucleotides with a specified length. These types of strings are nothing more than a series of letters A, C, T and G arranged in a random order. The number of generated strings and its length can be easily set using The Length input field and The Number of Primers field in The Primers section (see Figure 7).

If the proper checkboxes are set, every string is tested for presence of the following structures:

- repeats — repeating over four times adjacent assembly of two nucleotides, e.g.: ...ATATATATAT...,
- runs — the same nucleotides repeated over four times next to each other,
- percentage of C and G nucleotides in the fragment.

The presence of these structures has a significant impact on the biochemical characteristics of the generated fragments in relation to their further use. It is believed that the high stability of primers desired in the PCR process is largely related to the mentioned structures. If “repeats” or “runs” occur in the processed string, the string is eliminated. The same is true for “GC content” if the percentage of G and C nucleotides is outside the selected range.



Bits

Title

DNA Bits

Marek Miśkiewicz

ver.0.0.1

Generator

Formatter

Selector

Composer

Bits

Title

Generator

Output file

Input file Alternative primer...

Primers

Length: 20

Number of Primers: 1000

☑ Temperature

Minimal: 58,00

Maximal: 62,00

☑ GC Content

Minimal: 0,48

Maximal: 0,52

Parameters

☑ Runs ☑ Repeats

Start

Formatter

Selector

Composer

Figure 7: Main window of computer program for selecting DNA components (left) and Generator tab (right)

The Temperature section allows setting and controlling the melting temperature for generated fragments (and correlated primers). The highest stability of PCR procedure is guaranteed if the temperature is between 58 and 62 degrees Celsius. For the generated strings, the melting temperature is calculated using the following formula:

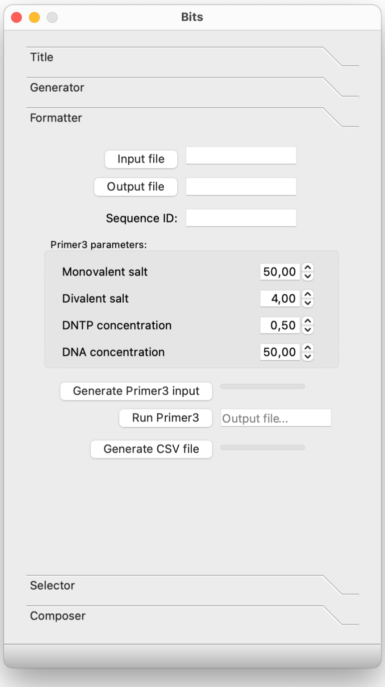
$$T_m = 4(G + C) + 2(A + T),$$

where A, C, T, G denote the number of corresponding nucleotides. If the temperature is within a specified range, the string is stored for further processing.

Output file button allows the user to select the file in which the drawn fragments will be saved. Data is saved in a simple text format where each line contains a single fragment stored as a string of nucleotides. Start button begins the process of generation. Due to the fact that the strings are random, every single "click" of the Start button generates a new, different set of strings.

3.4.2 Extended verifications and tests

The primer3 program is used to generate additional parameters for each fragment [19]. These parameters are connected with the possibility of existence of higher order structures of the DNA stands and their overall fragment stability. Because the primer3 program requires formatted input and some extra parameters to be set, Formatter tab allows the user to prepare the input file for the primer3 and pass additional parameters. Figure 8 shows the Formatter tab. The Input file field allows user to select file containing a set of primers used for primer3 input generator. Output file field is used to indicate the name of the file that finally contains the input data for the primer3 program. Sequence ID field allows for adding some additional identification information of files while working with the program (this has a minor meaning). The Generate Primer3 input button launches the process of generating primer3 input file.



Monovalent salt	50,00
Divalent salt	4,00
DNTP concentration	0,50
DNA concentration	50,00

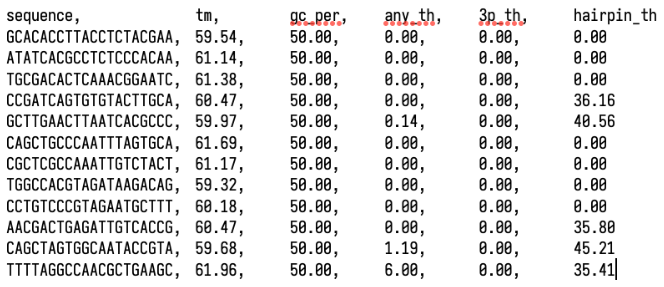
Figure 8: The Formatter tab

Primer3 parameters field allows the user to specify set of additional parameters required by primer3 program. These parameters describe PCR reaction conditions [21]:

- Monovalent Salts (mM) — the millimolar (mM) concentration of monovalent salt cations in the PCR. Primer3 uses this argument to calculate oligo and primer melting temperatures using this parameter.
- Divalent Salts ($MgCl^{2+}$, mM) — the millimolar concentration of divalent salt cations (usually $MgCl^{2+}$) in the PCR. Primer3 converts concentration of divalent cations to the concentration of the monovalent.
- DNTP Concentration (mM) — the millimolar concentration of the sum of all deoxyribonucleotide triphosphates. This argument is considered for oligo

and primer melting temperatures, for PCR product melting temperature, or for secondary structure calculations only if Divalent Salts is greater than 0.0.

- DNA Concentration (mM) — Value to use as nanomolar (nM) concentration of each annealing oligo over the course of PCR. Primer3 uses this argument to estimate oligo melting temperatures.



```
sequence,             tm,      gc_per,   any_th,   3p_th,    hairpin_th
GCACACCTTACCTCTACGAA, 59.54,   50.00,    0.00,     0.00,     0.00
ATATCACGCCTCTCCCACAA, 61.14,   50.00,    0.00,     0.00,     0.00
TGCGACACTCAAACGGAATC, 61.38,   50.00,    0.00,     0.00,     0.00
CCGATCAGTGTGTACTTGCA, 60.47,   50.00,    0.00,     0.00,     36.16
GCTTGAACTTAATCACGCCC, 59.97,   50.00,    0.14,     0.00,     40.56
CAGCTGCCCAATTTAGTGCA, 61.69,   50.00,    0.00,     0.00,     0.00
CGCTCGCCAAATTGTCTACT, 61.17,   50.00,    0.00,     0.00,     0.00
TGGCCACGTAGATAAGACAG, 59.32,   50.00,    0.00,     0.00,     0.00
CCTGTCCCGTAGAATGCTTT, 60.18,   50.00,    0.00,     0.00,     0.00
AACGACTGAGATTGTCACCG, 60.47,   50.00,    0.00,     0.00,     35.80
CAGCTAGTGGCAATACCGTA, 59.68,   50.00,    1.19,     0.00,     45.21
TTTTAGGCCAACGCTGAAGC, 61.96,   50.00,    6.00,     0.00,     35.41
```

Figure 9: The sample data from csv file

The Run Primer3 button runs the primer3 program. The Output file field located next to Run Primer3 button can be used to set the name of primer3 output file. The "generate csv" button allows you to generate a csv file based on the data contained in the output file from primer3. The csv format allows for a relatively easy processing of the results, especially the selection of appropriate primers. The sample data from the csv file is presented in Figure 9. The following columns in the csv file stand for:

- sequence — primer's nucleotide sequence,
- tm — melting temperature (described above),
- gc_per — percentage of nucleotides G and C,
- any_th — self-complementarity score of the oligo or primer, taken as a measure of its tendency to anneal to itself or form secondary structure,
- 3p_th — 3' self-complementarity of the primer or oligo, taken as a measure of its tendency to form a primer-dimer with itself,
- hairpin_th — the number that refers to the ability to create higher order structures called „hairpins".

Primer3 calculates these parameters using some thermodynamics models. They are highly correlated with so-called "higher order structures" and possible primer interactions (see Figure 10). Properly chosen set of primers prevent creation of self-dimmer, cross-dimmer and hairpins [20].

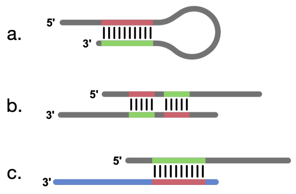
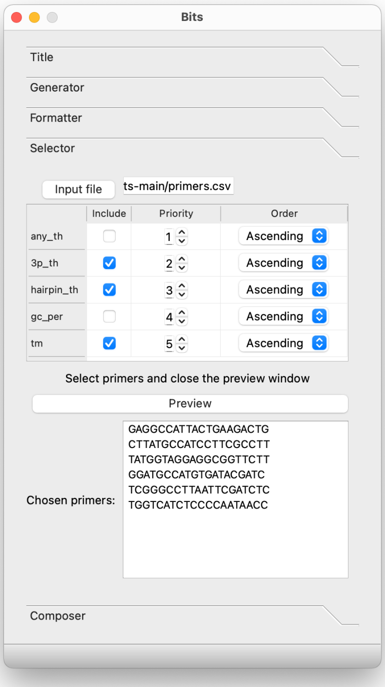


Figure 10: The higher order structures of primers: a) Hairpin, b) Self-dimmer, c) Cross-dimmer



ts-main/primers.csv

	Include	Priority	Order
any_th	☐	1	Ascending
3p_th	☑	2	Ascending
hairpin_th	☑	3	Ascending
gc_per	☐	4	Ascending
tm	☑	5	Ascending

Select primers and close the preview window

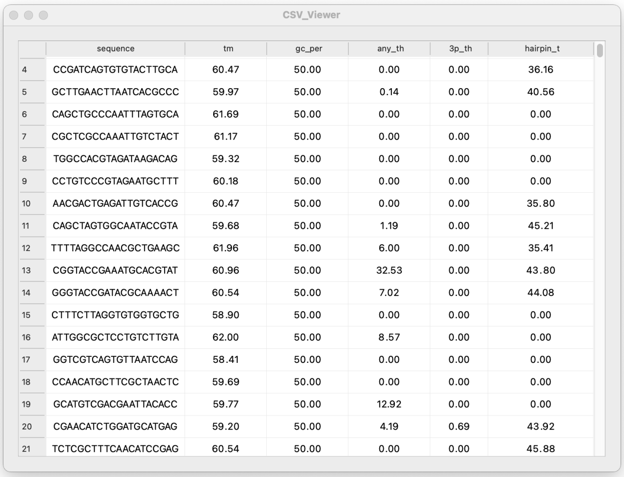
Chosen primers:

```
GAGGCCATTACTGAAGACTG
CTTATGCCATCCTTCGCCTT
TATGGTAGGAGGCGGTTCTT
GGATGCCATGTGATACGATC
TCGGGCCTTAATTCGATCTC
TGGTCATCTCCCCAATAACC
```

Figure 11: The Selector tab

3.4.3 The selection of primers

In the Selector tab (see Figure 11), the user can select primers basing on the information provided by the primer3 program. The Input file field allows the user to point to a csv file containing the results from the primer3 program. The selection panel allows to sort the displayed records (rows). Checkboxes in the "include" column allow to choose the fields to which the data will be sorted. "Spin buttons" in the "priority" column allow to set the priority of sorting. First sorted is the column for which the value "1" was set, then sorting takes place for the parameter for which the value "2" was set, etc. The "Order" column allows you to set ascending



	sequence	tm	gc_per	any_th	3p_th	hairpin_t
4	CCGATCAGTGTGTACTTGCA	60.47	50.00	0.00	0.00	36.16
5	GCTTGAACTTAATCACGCCC	59.97	50.00	0.14	0.00	40.56
6	CAGCTGCCCAATTTAGTGCA	61.69	50.00	0.00	0.00	0.00
7	CGCTCGCCAAATTGTCTACT	61.17	50.00	0.00	0.00	0.00
8	TGGCCACGTAGATAAGACAG	59.32	50.00	0.00	0.00	0.00
9	CCTGTCCCGTAGAATGCTTT	60.18	50.00	0.00	0.00	0.00
10	AACGACTGAGATTGTCACCG	60.47	50.00	0.00	0.00	35.80
11	CAGCTAGTGGCAATACCGTA	59.68	50.00	1.19	0.00	45.21
12	TTTTAGGCCAACGCTGAAGC	61.96	50.00	6.00	0.00	35.41
13	CGGTACCGAAATGCACGTAT	60.96	50.00	32.53	0.00	43.80
14	GGGTACCGATACGCAAAACT	60.54	50.00	7.02	0.00	44.08
15	CTTTCTTAGGTGTGGTGCTG	58.90	50.00	0.00	0.00	0.00
16	ATTGGCGCTCCTGTCTTGTA	62.00	50.00	8.57	0.00	0.00
17	GGTCGTCAGTGTTAATCCAG	58.41	50.00	0.00	0.00	0.00
18	CCAACATGCTTCGCTAACTC	59.69	50.00	0.00	0.00	0.00
19	GCATGTCGACGAATTACACC	59.77	50.00	12.92	0.00	0.00
20	CGAACATCTGGATGCATGAG	59.20	50.00	4.19	0.69	43.92
21	TCTCGCTTTCAACATCCGAG	60.54	50.00	0.00	0.00	45.88

Figure 12: The selector table with primers

or descending sorting order. The "Preview" button displays a window with a table containing the sorted primes (see Figure 12). Primer selections can be made in the Table window by simply clicking on them. The selected primer is highlighted in blue. Sorting allows the user to group the primes with the lowest values of parameters, such as `any_th`, `3p_th` and `hairpin_th`. These primes seem to be the most appropriate for the use in the presented storage system. The selected primes appear in the Preview box when the Table window is closed.

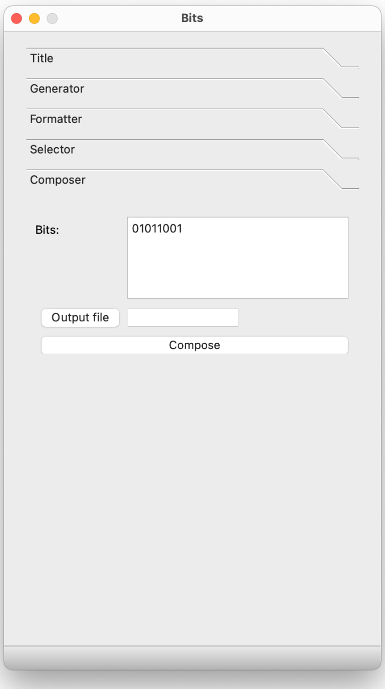
3.4.4 DNA strands building

The Composer tab, presented on Figure 13 allows you to prepare components to build a DNA strand with stored key bit information (see DNA Data structure section).

The primers selected in the previous step are assembled into longer fragments. The key bit structure is provided by the user via the "Bits" input field. The "Output file" field allows the user to indicate the file into which the strings of nucleotides forming the individual components of the DNA strand will be written. The structure of an example output file is shown in Figure 14. The labels `c_n` indicate sequentially: `c_0` — Start component; `c_1` to `c_8` — Bit component; `c_9` — End component. The nucleotide strings correspond exactly to the structural components presented in Figure 4. The information contained in the resulting file can provide the basis for synthesizing the DNA components necessary for the experiment.

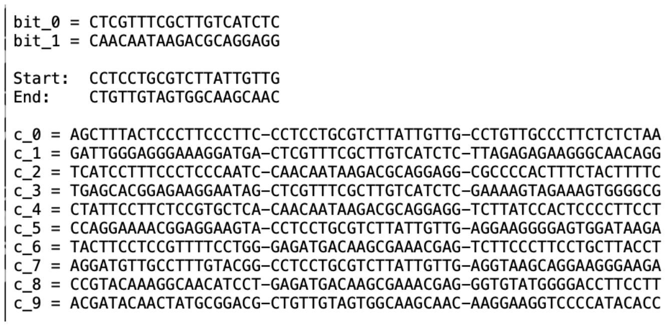
3.5 Keys reading procedure

To determine a series of bits in an extracted and isolated key, one must perform a two-step procedure used by Leier et al. in his work. First step is to carry out a PCR procedure with two types of primers. Solution with isolated and replicated



Bits: 01011001

Figure 13: The Composer tab



```
bit_0 = CTCGTTTCGCTTGTCATCTC
bit_1 = CAACAATAAGACGCAGGAGG

Start:  CCTCCTGCGTCTTATTGTTG
End:    CTGTTGTAGTGGCAAGCAAC

c_0 = AGCTTTACTCCCTTCCCTTC-CCTCCTGCGTCTTATTGTTG-CCTGTTGCCCTTCTCTCTAA
c_1 = GATTGGGAGGGAAAGGATGA-CTCGTTTCGCTTGTCATCTC-TTAGAGAGAAGGGCAACAGG
c_2 = TCATCCTTTCCCTCCCAATC-CAACAATAAGACGCAGGAGG-CGCCCCACTTTCTACTTTTC
c_3 = TGAGCACGGAGAAGGAATAG-CTCGTTTCGCTTGTCATCTC-GAAAAGTAGAAAGTGGGGCG
c_4 = CTATTCCTTCTCCGTGCTCA-CAACAATAAGACGCAGGAGG-TCTTATCCACTCCCCTTCCT
c_5 = CCAGGAAAACGGAGGAAGTA-CCTCCTGCGTCTTATTGTTG-AGGAAGGGGAGTGGATAAGA
c_6 = TACTTCCTCCGTTTTCCTGG-GAGATGACAAGCGAAACGAG-TCTTCCCTTCCTGCTTACCT
c_7 = AGGATGTTGCCTTTGTACGG-CCTCCTGCGTCTTATTGTTG-AGGTAAGCAGGAAGGGAAGA
c_8 = CCGTACAAAGGCAACATCCT-GAGATGACAAGCGAAACGAG-GGTGTATGGGGACCTTCCTT
c_9 = ACGATACAACTATGCGGACG-CTGTTGTAGTGGCAAGCAAC-AAGGAAGGTCCCCATACACC
```

Figure 14: The example of The Composer output file

key must be split into two reaction tubes. To the first tube one has to add primers corresponding to “0” bit DNA fragments, to the second reaction tube primers corresponding to “1” bit fragment must be added. Next for both tubes, PCR must be performed to elongate the primers. After PCR reaction tubes should contain shorter DNA strands with length matching to the position of “0” and “1” fragments. Figure 15 shows an example of PCR performed for strand encoding 8-bit sequence: 1 1 0 1 0 0 1 0. Second step requires implementation of gel electrophore-

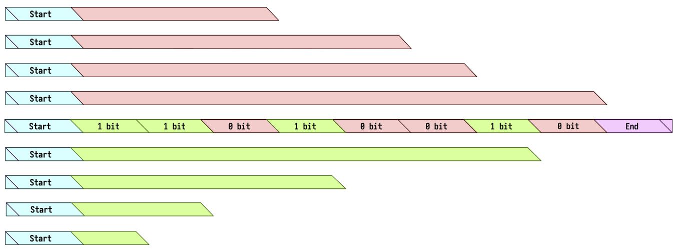


Figure 15: Expected primer length after elongation shown in example of 8-bit encoding DNA strand after PCR procedure

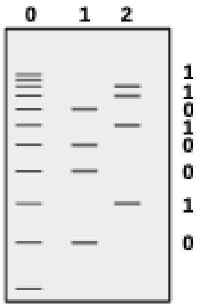


Figure 16: Example picture of gel electrophoresis performed for strands set from picture 10. Lane 0 symbolizes distribution of molecular weight marker, lane 1: distribution of strands length elongated with "0" primer, lane 2: distribution of strands length elongated with primer "1". Reading from bottom to top reveals encoded bit sequence — see right edge of picture

sis for PCR'ed mixture with the key. Contents of both reaction tubes must be put into gel separately on a different lane to visualize "0" bit bands and "1" bit bands. Positions of each band are related to DNA strand length in the analyzed sample. Because Bit fragments forming key comprise a determined number of nucleotides, some kind of "‘quantization" must occur after electrophoresis. In other way bands on the gel always should come up at the fixed positions showing positions of zeroes and ones in the analyzed key. Figure 16 shows expected bands distribution, for example, from Figure 15. To read a sequence of the entire key K every key must be read in the mentioned way.

4 The experiment

We conducted preliminary experiments to verify the method described, which concerned the possibility of easy and fast creation of cryptographic keys, which are in fact a sequence of fixed bits. In order to perform the experiments, DNA components were designed to create an eight-bit strand. Using the procedure described in the "DNA fragment selection procedure" section, we generate sixteen 20-nucleotide fragments (see Figure 17), from which components were then created to "assemble"

```
Bits primers:                 Binding sites:
CTCGTTTCGCTTGTCATCTC          AGCTTTACTCCCTTCCCTTC
CAACAATAAGACGCAGGAGG          GATTGGGAGGGAAAGGATGA
                              TCATCCTTTCCCTCCCAATC
Start component primers:      TGAGCACGGAGAAGGAATAG
AGCTTTACTCCCTTCCCTTC          CTATTCCTTCTCCGTGCTCA
CCTCCTGCGTCTTATTGTTG          CCAGGAAAACGGAGGAAGTA
CCTGTTGCCCTTCTCTCTAA          TACTTCCTCCGTTTTCCTGG
                              AGGATGTTGCCTTTGTACGG
End component primers:        CCGTACAAAGGCAACATCCT
ACGATACAACTATGCGGACG
CTGTTGTAGTGGCAAGCAAC
```

Figure 17: The set of primers selected for the experiment using a described computer program

Data Components	DNA Strand Components	Key bit
Start:	AGCTTTACTCCCTTCCCTTC-CCTCCTGCGTCTTATTGTTG-CCTGTTGCCCTTCTCTCTAA	
Bit_1:	GATTGGGAGGGAAAGGATGA-CTCGTTTCGCTTGTCATCTC-TTAGAGAGAAGGGCAACAGG	„1”
Bit_2:	TCATCCTTTCCCTCCCAATC-CAACAATAAGACGCAGGAGG-CGCCCCACTTTCTACTTTTC	„0”
Bit_3:	TGAGCACGGAGAAGGAATAG-CTCGTTTCGCTTGTCATCTC-GAAAAGTAGAAAGTGGGGCG	„1”
Bit_4:	CTATTCCTTCTCCGTGCTCA-CAACAATAAGACGCAGGAGG-TCTTATCCACTCCCCTTCCT	„0”
Bit_5:	CCAGGAAAACGGAGGAAGTA-CCTCCTGCGTCTTATTGTTG-AGGAAGGGGAGTGGATAAGA	„0”
Bit_6:	TACTTCCTCCGTTTTCCTGG-GAGATGACAAGCGAAACGAG-TCTTCCCTTCCTGCTTACCT	„1”
Bit_7:	AGGATGTTGCCTTTGTACGG-CCTCCTGCGTCTTATTGTTG-AGGTAAGCAGGAAGGGAAGA	„0”
Bit_8:	CCGTACAAAGGCAACATCCT-GAGATGACAAGCGAAACGAG-GGTGTATGGGGACCTTCCTT	„1”
End:	ACGATACAACTATGCGGACG-CTGTTGTAGTGGCAAGCAAC-AAGGAAGGTCCCCATACACC	

Figure 18: The final structure of DNA strands component built from selected primers (due to the elongation process some DNA components have to be synthesized as its complementary counterparts)

a DNA strand with a given bit pattern. We prepared the DNA components so that they could encode the string “10100101”. The exact single DNA strands used in the experiment are presented in Figure 18. Authors of this article can provide the details of the biochemical procedure and primers used in the experiment. The elongation process resulted in a DNA strand of 420 nt length. The obtained material was then subjected to a reading process using the method described in the “key reading” section. The reading process revealed the expected key bit pattern.

5 Conclusions and future work

A simple cryptographic keys creation and storing system based on DNA strands were presented. Despite the considerable complexity because of the relatively large number of necessary elements, the system does not require the participation of third parties in very important steps such as the key creation and reading. In addition, a computer program and a simple selection procedure for primers necessary in preparing DNA strands with stored keys is presented. The program was successfully used during the experiment. However, it should be stated that the experiment was successfully conducted on a small scale. Further work should therefore focus on extending the system so that keys with meaningful lengths can be operated on. The program itself, although fulfilling its task, also needs a lot of work.

References

[1] Yang, K., Hicks, M., Dong, Q., Austin, T., and Sylvester, D. (2016). *A2: Analog malicious hardware.* In 2016 IEEE Symposium on Security and Privacy (SP), pages 18–37.

[2] Becker, G. T., Regazzoni, F., Paar, C., and Burleson, W . P . (2014). *Stealthy dopant-level hardware trojans: extended version.* Journal of Cryptographic Engineering, 4(1):19–31.

[3] Kumar, R., Jovanovic, P., Burleson, W., and Polian, I. (2014). *Parametric trojans for fault-injection attacks on cryptographic hardware.* In 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, pages 18–28.

[4] Savage, E., Gilbert, J., and Radasky, W. (2010). *The early-time (e1) high-altitude electromagnetic pulse (hemp) and its impact on the U.S. power grid.* Technical report, Metatech Corporation, 358 S. Fairview Ave., Suite E Goleta, CA 93117.

[5] Adleman, L. M. (1994). *Molecular computation of solutions to combinatorial problems.* Science, 266(5187):1021–1024, DOI: 10.1039/C0CS00051E

[6] Gehani, A., LaBean, T., and Reif, J. (2004). *DNA-based Cryptography*, pages 167–188. Springer Berlin Heidelberg, Berlin, Heidelberg, DOI: 10.1007/978-3-540-24635-0_12

[7] Zhang, Y., Liu, X., and Sun, M. (2017). *Dna based random key generation and management for otp encryption.* Biosystems, 159:51–63, DOI: 10.1016/j.biosystems.2017.07.002

[8] Cano, R. and Borucki, M. (1995). *Revival and identification of bacterial spores in 25- to 40-million-year-old dominican amber.* Science, 268(5213):1060–1064, DOI: 10.1126/science.7538699

[9] Vreeland, R., Rosenzweig, W., and Powers, D. (2000). *Isolation of a 250-million-year-old halotolerant bacterium from a primary salt crystal.* Nature, 407(6806):897–900, DOI: 10.1038/35038060

[10] Grass, R. N., Heckel, R., Puddu, M., Paunescu, D., and Stark, W. J. (2015). *Robust chemical preservation of digital information on DNA in silica with error-correcting codes.* Angewandte Chemie International Edition, 54(8):2552–2555, DOI: 10.1002/anie.201411378

[11] Shah, S. B. and Elerath, J. G. (2005). *Reliability analysis of disk drive failure mechanisms.* Annual Reliability and Maintainability Symposium, 2005. Proceedings., pages 226–231.

[12] Halvorsen, K. and Wong, W. P. (2012). *Binary dna nanostructures for data encryption.* PLOS ONE, 7(9):1–4, DOI: 10.1371/journal.pone.0044212

[13] Tanaka, K., Okamoto, A., and Saito, I. (2005). *Public-key system using dna as a one-way function for key distribution.* Biosystems, 81(1):25–29, DOI: 10.1016/j.biosystems.2005.01.004

[14] Leier, A., Richter, C., Banzhaf, W., and Rauhe, H. (2000). *Cryptography with dna binary strands.* Biosystems, 57(1):13–22, DOI: 10.1016/s0303-2647(00)00083-6

[15] Shiu, H., Ng, K., Fang, J., Lee, R., and Huang, C. (2010). *Data hiding methods based upon dna sequences.* Information Sciences, 180(11):2196–2208, DOI: 10.1016/j.ins.2010.01.030

[16] Oh, J.-M., Park, D.-H., and Choy, J.-H. (2011). *Integrated bio-inorganic hybrid systems for nanoforensics.* Chem. Soc. Rev., 40:583–595, DOI: 10.1039/C0CS00051E

[17] Cormier, S., Shearman, J., and Hogan, M. (2018). *Dna in your jeans? effect of abrasion and bleaching on dna tagged denim.* AATCC Review, 18:44–48, DOI: 10.14504/ar.18.5.4

[18] M. Miśkiewicz, B. Księżopolski (2019). *Cryptographic keys management system based on DNA strands.* Federated Conference on Computer Science and Information Systems (FedCSIS), Leipzig, Germany, 2019, pp. 231–235, DOI: 10.15439/2019F313

[19] Koressaar T, Lepamets M, Kaplinski L, Raime K, Andreson R and Remm M. *Primer3_masker: integrating masking of template sequence with primer design software.* Bioinformatics 2018; 34(11): 1937–1938, DOI: 10.1093/bioinformatics/bty036

[20] Untergasser A, Cutcutache I, Koressaar T, et al. *Primer3 — new capabilities and interfaces.* Nucleic Acids Res. 2012;40(15):e115. doi:10.1093/nar/gks596

[21] https://apps.thermofisher.com/apps/help/MAN0015956/GUID-0BB0CF63-1755-4A58-B7C8-342FAE393211.html, access 28 July 2021.

[22] O. Milenkovic and N. Kashyap, *On the design of codes for DNA computing, Coding and Cryptography.* Springer, 2006, pp.100–119, DOI: 10.1007/11779360_9

Comparative Analysis of Selected Anthropomorphic Grippers Constructions

Paweł Olszewski[*]

1 Introduction

Technological progress in the field of robotic grippers is mainly caused by the demands of industrial automation and production lines. Dedicated solutions used there must fulfill specific requirements such as high resistance to environment conditions, durability, high speed or precision and be able to efficiently carry out specific tasks assigned to them. In contrast to this group of robotic grippers, where esthetic matter is less important and similarity to human hand mostly not even considered, there is also a second, smaller group of constructions, inspired by the human hand. Beside the esthetic matter and the higher level of acceptance from people working in collaboration with such devices, this second type of grippers has also an undeniable advantage of interacting with interfaces and tools originally invented with human hands in mind. This ability is crucial when it comes to prosthetic applications, but can also be very useful in developement of personal robotic assistants, whose most tasks are to be done in human oriented environment. Despite many attempts taken until now, none of them resulted with ultimate solution. All the existing constructions are more or less simplified in comparison to the original human hand. This situation is caused mainly by the human hand high level of complexity and technological limitations that engineers must somehow overcome during production process.

In this paper selected anthropomorphic grippers constructions will be introduced and compared to each other and to the original human hand. The work presents the number of degrees of freedom, that each construction offers, and their similarity to human hand. In this paper the comparative analisis concerns four robotic grippers: AR10, SVH, Shadow Dexterous Hand and Shadow Dexterous Hand Lite. These are the ones of the most popular and easy available.

[*]Corresponding author — `pawel.olszewski@mail.umcs.pl`

2 The human hand

Human hand is capable of amazing dexterity and performs many functions, from wide range of grasping to delicate and precise manipulation tasks.

Human hand consists of 14 phalanges bones, 5 metacarpal bones and 8 wrist bones, which gives total of 27 [2]. Hand movements are provided by a complex system of muscles and tendons spread along the bone structure and specialized in a way assuring high strength and dexterity, keeping low mass at the same time.

To perform a succesfull grasp of an object multiple digits must firstly spread open and then close around it. Futhermore, ability to move large number of those digits in independ way is highly relevant for precise manipulation of small objects.

Despite the fact that human hand is often treated as the ultimate solution, even its fingers movements are restricted in some way. For example, we are capable of touching the palm with every one of our tips, but we cannot do the same touching back site of the hand with them. We can also set our thumb opposite to other fingers, but we cannot set those fingers oppose one to another [6]. Because of the presence of biomechanical passive couplings between digits in human hand, there are situations when activity aiming to move only one digit will tend to move the adjacent ones as well.

2.1 The human hand kinematic model

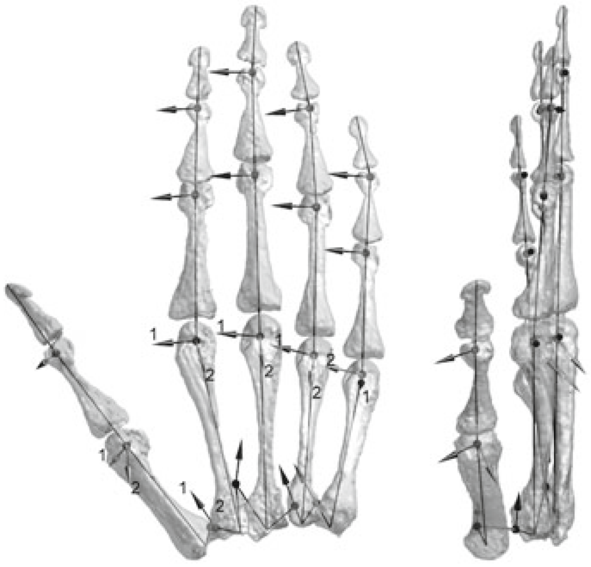


Figure 1: Kinematic model of human hand [9]

Before further analysis of robotic grippers, it is natural to choose the kinematic model of the human hand — theirs prototype. The work [9] used magnetic resonance imagining (MRI) images of a hand and its bones in various postures to determine precise orientations and positions of the axes of rotation for each finger. Because models were created with robotic systems in mind to avoid friction and control difficulties in them, only rotary joints were considered during model creation. This approach resulted in hand models of different accuracy and complexity — three of them were selected and discussed by authors. Those are 22, 24 and 33 DoF kinematic models. The choosen reference model (Fig. 1) is the middle one with 24 DoF, where arrows shows the axes of possible flexion and numbers placed near them tells us about the index of such axe in this particular joint. Some joints, like the little finger DIP joint, provides only one flexion axe, others allows movement in two axes — like the index MCP joint (Fig. 2).

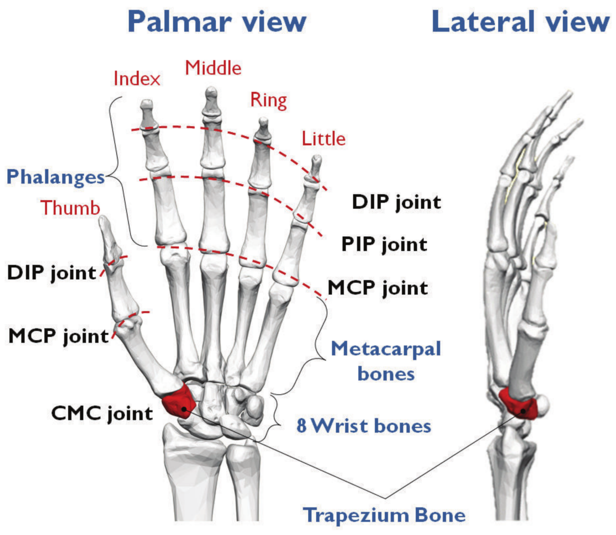


Figure 2: Bones in human hand [10]

We can see that two most distal joints of all fingers, except the thumb, are the 1-axe joints. Furthermore, third joint of every finger and second joint of the thumb are the 2-axe joints. The fourth joint of middle, ring and index fingers provide them additional single-axed movement in the wrist.

3 AR10 Humanoid Robot Hand

AR10 Humanoid Robot Hand (Fig. 3) is construction made by Active8 Robots company and offers movement in 14 joints, each consisted of only one flexion axe and limited to 10 degreees of freedom (DoF) provided by the same number of linear actuators mounted inside the hand [4]. Weight of this gripper is 0.475 kilo and when its middle finger is in straight position the gripper reaches length of 200 milimeters [11].

It was dedicated for academic purposes as low cost alternative for other constructions and can work as standalone or in a bigger system after being mounted on robotic arm such as Sawyer, Rethink Robotics or Baxter.

AR10 was designed with grasping tasks in mind and was inspired by human hand grasping capabilities and look. Finger tips can be dissassembled in a fast way and easily replaced. Provided as opensource fingertips models makes it easier to redesign them, quickly produce with 3D printer and adjust hand for own purposes.

Built in electronics has four free connections allowing to connect up to four touch sensors or two additional drives. Those additional drives can assure wrist movement — when using them that way, we must reserve two connections for signals returning from drive to electronics to provide proper movement. Communication with gripper is provided by USB or serial port and it is fully compatible with Robot Operating System (ROS). The gripper is available in left and righthanded version.

3.1 AR10 kinematic model

Every finger, except the thumb, has 3 joints (Fig. 4), however only 2 independents drives. It means, that in all of them the 2 highest joints depend on the one engine and exist in constant cooperation. The thumb possesses 2 independent drives controlling 2 joints.

4 Schunk SVH

SVH (Servo-electric 5-Finger Gripping Hand) is an antropomorphic robotic gripper (Fig. 5) with length of 242 milimeters and total weight of 1.3 kilos, provided by SHUNK company [12]. It is equipped with five fingers offering movement

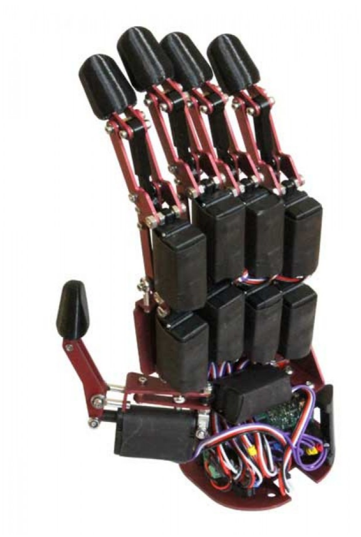


Figure 3: AR10 robotic gripper [11]

in 9 DoF assured by 9 individual drives. Gripper pose can be changed in 20 joints acompishing wide range of grips. Elastic gripping surface assures that object grasping would be more stable. Highly compact gripper construction integrates all of its electronics inside the wrist. The producer delivers dedicated driver for using the gripper with ROS. Communication with the device is possible through RS-485 serial interface.

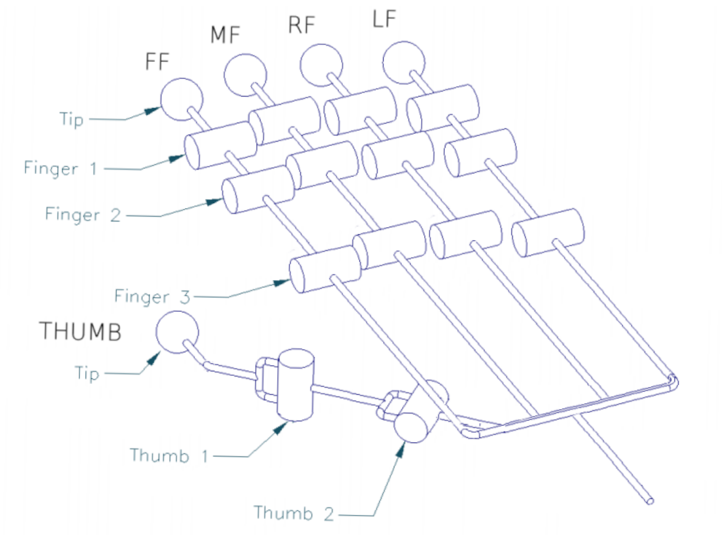


Figure 4: Kinematic model of AR10 (based on Shadow Dexterous Hand Specification)

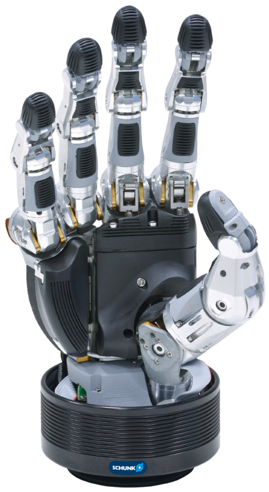


Figure 5: SHUNK SVH robotic gripper (Assembly and Operating Manual SVH)

Hand was developed with grasping and object manipulation tasks in mind, but as the result of wide range of possible poses it would be also useful in human-robot communication using gestures. The wrist movements are not provided by hand in any form. Integration with robotic arms, available on the market, is possible due to the well defined interfaces. It is offered in left- and righthanded version.

4.1 SVH kinematic model

The thumb, index finger and middle finger have 2 independent engines, while the other two have only one drive each. One servo-motor is responsible for opposite movement of the thumb in joint J1 (Fig. 6) and small finger together with ring finger in joint J5. Second servo-motor controlls thumb flexion in coupled joints J2, J3 and J4. Separate two servo-motors are independently driving flexion of index finger in joint J6a and middle finger in joint J7. Single servo-motor is responsible for spreading small, ring and index finger in joints J6b, J8b and J9b. Next servo-motor controlls index finger flexion in coupled joints J10 and J14, similarly single servo-motor controlls middle finger flexion in coupled joints J11 and J15. For flexion of ring finger in all three coupled joints (J8a, J12 and J16) is responsible only one servo-motor. All three joints (J9a, J13 and J17) of small finger are also mechanically coupled and only one servo-motor is responsible for its flexion movement [5].

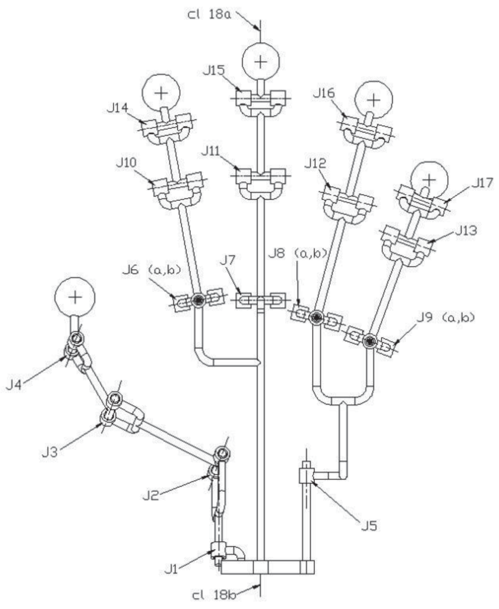


Figure 6: Kinematic diagram of the SVH [5]

5 Shadow Dexterous Hand

Shadow Dexterous Hand is an advanced humanoid construction of robotic gripper made by Shadow Robot Company (Fig. 7). It weights 4.3 kilo and has total length of 448 milimeters [13]. Integrated thumb and four fingers (with additional two movements possible in the wrist) are offering movement in 24 joints with 20 DoF. Gripper was designed to be highly similar to the human hand, reproduce as closely as possible its dexterity and kinematics and also mimic its shape and proportions.

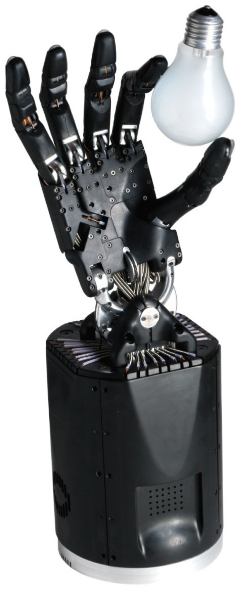


Figure 7: Shadow Dexterous Hand [7]

Producer states that gripper can be equipped with up to 129 sensors, including position, strength, pressure and touch sensors. Construction, due to full compatibility with ROS, offers quick way of beggining work. Due to gripper flexible design it can be adjusted for specific tasks or integrated as an element of bigger system cooperating with one of many robotic arms. It is available in right- and lefthanded version.

5.1 SDH kinematic model

The index, middle and ring finger in the SDH hand model possess 3 independent drives each (Fig. 8). The little finger has more drives – in the number of 4. The most numerous independent drives are located in the thumb, who has the 5 of them. Moreover, 2 drives are situated in the wrist. Flexion movements in joints FF1 and FF2 are coupled and actuated by single drive. The same situation refers to middle finger, ring finger and little finger joints 1 and 2. Two separated drives were used only for thumb flexion movement in joints TH1 and TH2. Dedicated drives were also used for assuring flexion movement each one for joints 3 and 4 of all fingers including thumb, which gives 10 drives.

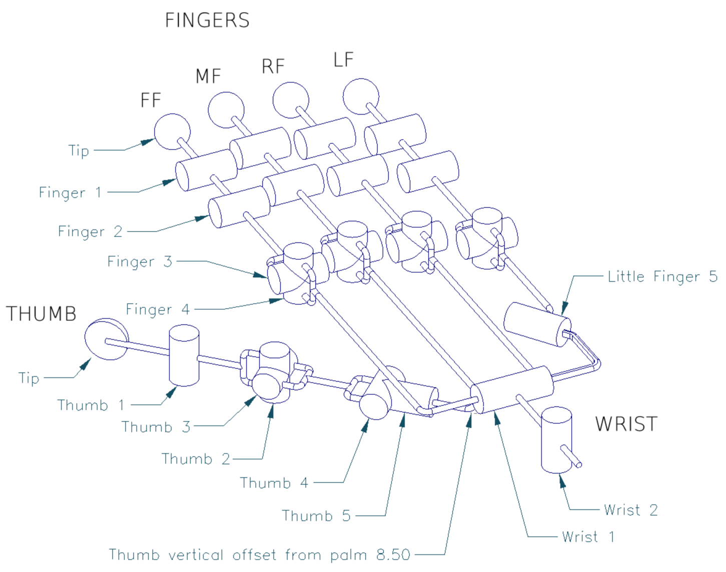


Figure 8: Kinematic model of SDH [3]

6 Shadow Dexterous Hand Lite

Shadow Dexterous Hand Lite (Fig. 9), made by Shadow Robot Company, was designed to be smaller, lighter, simpler and cheaper alternative for basic version of Shadow Dexterous Hand. It also maintains to mimic human-like kinematics as its predecessor. It offers movement in 16 joints with total of 13 DoF. Unlike SDH, in SDHL there is no movement in wrist and the little finger is missing. As the result, the integrated forearm of the hand is 121 mm smaller than the SDH, which gives total length of 327 mm. Secondly, the hand is also lighter (2.4 kg), weighting 1.9 kg less than the full version [14]. Hand was designed for tasks that needs flexibility and dexterity at the level of human hand, but it is not necessary for each of five fingers to be present. Like in the SDH, the construction offers quick way of beggining work due to full compatibility with ROS. Gripper is available in right- and lefthanded version.

6.1 SDHL kinematic model

When it comes to kinematic diagram (Fig. 10), it is partialy similar to the one of SDH. Movement in joints 1 and 2 is coupled in every finger beside the thumb and

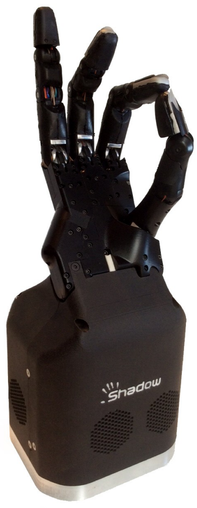


Figure 9: Shadow Dexterous Hand Lite [8]

driven by dedicated drive. Movements in joints 3 and 4 for every finger are provided by independent two drives each. Also joints 1,2,4 and 5 of thumb are independent and moved by individual drives. That gives total of 13 independent drives.

7 Comparison

The largest number of joints and DoF among all analyzed grippers is observed with SDH model, the smallest number with AR10. The similar situation is visible in the number of joints and DoF within their thumbs (Tab. 1). In all of the discussed models joints number is higher than DoF number due to presence of drives responsible for movement in coupled joints. For example in the SVH model single drive is responsible for movement in three coupled joints. The biggest difference between the number of joints and DoF is present in model SVH, in which the joints number (20) is more than two times higher than the DoF number (9). Despite small number of DoF this model still allow a high dexterity and precision.

In most cases the bigger number of DoF results in higher weight and increased size of the gripper. SDH, the model containing the biggest number of joints and DoF, is more than ten times weightier and two times bigger than the AR10 hand, which contains half of the DoF located in the first one. In the SDH gripper higher weight is also connected with the presence of augmented wrist and its drives.

Number of DoF is strictly connected with the number of individual drives among the analysed models in the fingers (Tab. 2). Among the analysed models there are from one to five individual drives responsible for particular flexion axes. In all of the models the thumb possesses the largest DoF number, or at least the number equal to the other fingers.

Table 1: Basic parameters of selected robotic hand models

	AR10	SVH	SDH	SDHL
Weight [kg]	0,475	1,3	4,3	2,4
Length [mm]	200	242	448	327
Joints	14	20	24	16
DoF	10	9	20	13
Thumb [DoF]	2	2	5	4
Thumb [joints]	2	4	5	4
Actuated wrist	NO	NO	YES	NO

8 Conclusion

The main purpose of antropomorphic robotic grippers constructions is to recreate possibly the most accurate reconstruction of human hand. The similarity of the robotic gripper dexterity to the human hand is demonstrated mainly by the number of joints and DoF — the bigger number of them results in the bigger set of various finger flexion movements. In this paper the comparative analisis concerns four robotic grippers: AR10, SVH, SDH and SDHL. These are the ones of

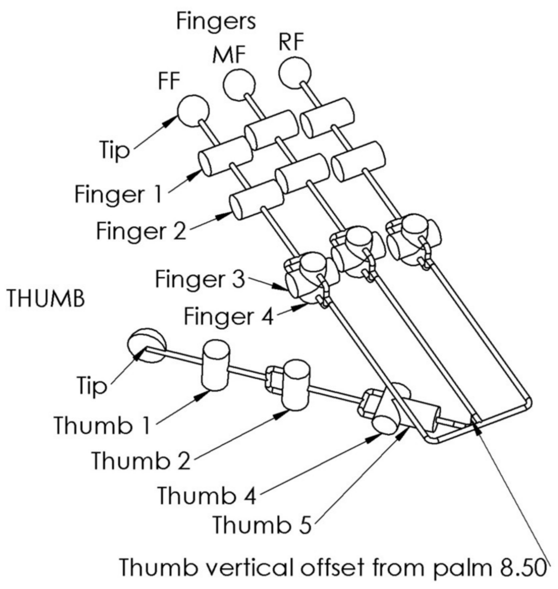


Figure 10: Kinematic model of SDHL [8]

Table 2: Independent drives in selected robotic hand models

	AR10	SVH	SDH	SDHL
LF	2	1	4	-
RF	2	1	3	3
MF	2	2	3	3
FF	2	2	3	3
TH	2	2	5	4
other	-	1 (opposite)	2 (wrist)	-
total	10	9	20	13

the most popular and easy available. Among them the higher similarity to the human hand demonstrates the SDH model having 24 joints and 20 DoF. Although it does not mean that the other models are worse — their number of joints and DoF are smaller, however they can be succesfully applied in different fields, where the simplier constructions are desirable. They are also much more smaller and lighter than prievously mentioned SDH. Actually the biggest construction problem is to integrate the proper number of drives inside the robotic gripper, because of their size.

References

[1] Assembly and operating manual SVH, Schunk Gmgh and Ca KG, 2019. Accessed on: Jun 15, 2021. [Online]. Available: https://schunk.com

[2] Frank P. J. van der Hulst F.P.J., Schatzle S., Preusche C., Schiele A., 2012, A Functional Anatomy Based Kinematic Human Hand Model with Simple Size Adaptation, 2012 IEEE International Conference on Robotics and Automation, May 14–18, Minnesota, USA.

[3] Li S., Ma X., Liang H., Gorner M., Ruppel P., Fang B., Sun F., Zhang J., 2019, Vision-based Teleoperation of Shadow Dexterous Hand using End-to-End Deep Neural Network, International Conference on Robotics and Automation (ICRA)

[4] Rao B., Li H., Krishnan K., Boldsaikhan E., He H. Knowledge-Augmented Dexterous Grasping with Incomplete Sensing, arXiv:2011.08361[preprint], november 17, 2020 [cited 2021 june 15].

[5] Ruehl S. W., Parlitz C., Heppner G., Hermann A., Roennau A., Dillmann R., 2014, Experimental Evaluation of the Schunk 5-Finger Gripping Hand for Grasping Tasks, IEEE International Conference on Robotics and Biomimetics December 5–10, Bali, Indonesia.

[6] Schieber M. H., 2014, Constraints and Flexibility in Cortical Control of the Hand [in] Balasubramanian R., Santos V.J. (ed.), The Human Hand as an Inspiration for Robot Hand Development, Springer

[7] Shadow Dexterous Hand Technical Specification, Shadow Robot Company, Feb 2019. Accessed on: Jun 15, 2021. [Online]. Available: https://shadowrobot.com

[8] Shadow Dexterous Hand Technical Specification Shadow Dexterous Hand G Series: Shadow Hand Lite, Shadow Robot Company, Sep 2015. Accessed on: Jun 15, 2021. [Online]. Available: https://shadowrobot.com

[9] Stillfried G., Hillenbrand U., Settles M. , van der Smagt P., 2014, MRI-Based Skeletal Hand Movement Model [in] Balasubramanian R., Santos V.J. (ed.), The Human Hand as an Inspiration for Robot Hand Development, Springer

[10] Xu Z., Todorov E., 2016, Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration, IEEE International Conference on Robotics and Automation (ICRA).

[11] https://active8robots.com/robots/ar10-robotic-hand/ [access: Mar 12, 2016].

[12] Assembly and operating manual SVH, Schunk Gmgh and Ca KG, 2019.
https://schunk.com [access: Jun 15, 2021].

[13] Shadow Dexterous Hand Technical Specification, Shadow Robot Company, Feb 2019.
https://shadowrobot.com [access: Jun 15, 2021].

[14] Shadow Dexterous Hand Technical Specification Shadow Dexterous Hand G Series: Shadow Hand Lite, Shadow Robot Company, Sep 2015.
https://shadowrobot.com [access: Jun 15, 2021].

Data Mining Procedures in the Oil Production Prediction for Gas Lifted Wells

Bartłomiej Bielecki
Andrzej Krajka*

1 Introduction

Full wellbore modeling during oil production is an interesting and very complicated problem. There are some collections of a huge amount of papers regarding the simulation [22]. Modern computerization allows creating sophisticated and accurate, but time-consuming models considering different approaches to the simulation. The authors used the solution described in recent papers (cf. [3], [4]). The results are comparable with other studies as (cf. [16], [19]). The standard simulation model has been extended by the authors and dedicated to a gas lift procedure which is treated as the optimization problem also appeared in the literature [18]. To increase the oil inflow, procedures such as gas lift or water injection are employed. Three phase flow appeared in a productive tubing string has many meaningful parameters (MP) responsible for the production results. The gas is provided via annulus, which is a space between tubing and casing strings. In an annulus, single phase gas flow conditions are expected. Gas is injected through Gas Lift Valves (GLV) into the tubing string. Two parameters characterizing GLV are depth and lift. The lift means amount of injected gas, presented in millions of standard cubic feet per day [MMScf/D].

2 Motivations and contributions

The full simulation process takes into the consideration input parameters such as: completion, wellbore trajectory, reservoir data. Based on these inputs, the full model of production oil (PO model) is employed. The authors wanted to check whether the data mining algorithms can meet the production results as PO model does. The relative error of PO was about 5 percent which is a satisfactory result. On the other hand, this model has two significant drawbacks:

*Corresponding author — andrzej.krajka@mail.umcs.pl

- The full PO model is time-consuming and sometimes has a discontinuity.
- A large number of parameters which has to be set prior to the simulation run.

These reasons motivated the authors to check another approach, called the MO model, to estimate production results based on GLV parameters, bottom hole pressure, and recent production results in the dedicated wellbore. We compared data mining procedures as: neural networks, different regression models, projection pursuit regression (ppr), multidimensional splines approximations (polymars), support vector machines (svm), and multiple additive regression trees (MART) approximation methods. These results may be used for:

- Pre-production wellbore simulation with the GLV lift and depth estimation.
- The real-time decisions regarding the gas injection amount
- The creation of a fuzzy controller as a future job.
- Finding the best data mining method for wellbore simulation.

To achieve the results, many runs of PO model were generated using the framework written in C# .NET. Data was saved into the database system and data mining methods were performed with the R Studio environment. Pseudocode and its R code realization have been put as the attachment of this paper, so it is available for testing. In this case, inputs are restricted to a few parameters, in comparison to the PO model. The assumption that the total number of GLV is maximum four, has been made. This study may be applied for pre-completion stage once a producer analyses the best placement of GLV in a completion. Hence we consider GLV depth and lift (injection) as the crucial input parameters of our study.

3 Data preparation

We consider the following input data:

BHP — Bottom hole pressure. This value is usually acquired from gauges at the deepest point of each completion. Usually, this value fluctuating during the production.

GLVd$_i, 1 \leq i \leq 4,$ — the depth of the i-th GLV (NA if i-th GLV is closed and the injection is stopped).

GLVl$_i, 1 \leq i \leq 4,$ — the amount of the gas injected from the annulus to the tubing string via i-th GLV, counted in million standard cubic feet per day unit [MMScf/D] (NA if i-th GLV is closed and the injection is stopped).

POIL — The value of the oil production of the PO model, identified at the surface, as the sum of the whole inflow from the production layers.

This research is based on 2810 PO model different runs. Different wellbore and reservoir conditions as the typical at an oilfield has been simulated. Records are joined into pairs on three different ways (cf. for eg. Table 1).

The technical field $AmGLV$ indicates the number of used GLV. As it is shown, the records are paired as follows:

- in type 1 only one parameter among $GLVd_i, i = 1, 2, 3, 4$, is different
- in type 2, only one parameter among $GLVl_i, i = 1, 2, 3, 4$, is different
- in type 3, one more GLV is added.

Furthermore, we created pairs that changes as small as it is possible. For two records paired one is deleted and one is passed to the next possible pair. Pairs described above will be the product the following records in the data frame *Result1*. Such that from Table 1 we may obtain results as in Table 2.

In the column $dPOIL$ we storage the value of increase of the oil production. In type 1, x and $R1$ indicate the GLV number which has been replaced and the value of change this GLV depth, respectively. GLVs are ordered by the depth ascending. In type 2, x and $R2$ contains the GLV number and this GLV lift change. In type 3, the x indicates new GLV parameter (number) and depth and lift this new GLV are placed in $R1$ and $R2$, respectively. Now we add the extra fields to our structure *Result1* according to the formulas:

$$\begin{aligned} PIL_i &= GLVd_i * (1 + GLVl_i), \\ PIE_i &= GLVd_i * exp(GLVl_i), \\ PID_i &= GLVd_i / (1 + GLVl_i), \\ PIV_i &= log(GLVd_i) / (1 + GLVl_i), \\ & i = 1, 2, 3, 4. \end{aligned} \tag{3.1}$$

Table 1: Example of input data records paired in different types

Type	*POIL*	*BHP*	$GLVd_1$	$GLVl_1$	$GLVd_2$	$GLVl_2$	$GLVd_3$	$GLVl_3$	$GLVd_4$	$GLVl_4$	*AmGLV*
1	3823.5	2000	4000	1	4100	1	4800	2	*NA*	*NA*	3
	3125.9	2000	4000	1	4200	1	4800	2	*NA*	*NA*	3
2	4229.8	2000	4000	1	4100	1	5000	1	*NA*	*NA*	3
	2287.3	2000	4000	1	4100	1	5000	2	*NA*	*NA*	3
3	2754.4	3200	4900	4	5100	1	*NA*	*NA*	*NA*	*NA*	2
	2983.9	3200	4900	4	5100	1	5800	2	*NA*	*NA*	3

Table 2: Data frame *Result1* obtained from data presented in Table 1

Type	*dPOIL*	*POIL*	*BHP*	$GLVd_1$	$GLVl_1$	$GLVd_2$	$GLVl_2$	$GLVd_3$	$GLVl_3$	$GLVd_4$	$GLVl_4$	*AmGLV*	x	$R1$	$R2$
1	−697.6	3823.5	2000	4000	1	4100	1	4800	2	*NA*	*NA*	3	2	100	*NA*
2	−1942.5	4229.8	2000	4000	1	4100	1	5000	1	*NA*	*NA*	3	2	*NA*	1
3	229.5	2754.4	3200	4900	4	5100	1	*NA*	*NA*	*NA*	*NA*	2	2	5800	2

In practical aims we consider the following three situations:

S1. We have only one GLV with $GLVd_1$ and $GLVl_1$ parameters.

S2. We have two GLV with $GLVd_i$ and $GLVl_i, i = 1, 2$, parameters.

S3. We have three GLV with $GLVd_i$ and $GLVl_i, i = 1, 2, 3$, parameters.

As we mentioned GLV points $GLVd_i$ and $GLVl_i$ are ordered ascending according to $\{GLVd_i, i \geq 1\}$. Hence we identified 17 new possible actions that appeared under these changes.

I. We changed $GLVd_i$ value, and replaced by:

[A1.] one GLV in the situation S1.

[A2.] $GLVd_1$, in the S2 situation

[A3.] $GLVd_2$, in the S2 situation

[A4.] $GLVd_1$, in the S3 situation

[A5.] $GLVd_2$, in the S3 situation

[A6.] $GLVd_3$, in the S3 situation

II. We change $GLVl_i$ value for:

[A7.] one GLV in the situation S1.

[A8.] $GLVl_1$, in the S2 situation

[A9.] $GLVl_2$, in the S2 situation

[A10.] $GLVl_1$, in the S3 situation

[A11.] $GLVl_2$, in the S3 situation

[A12.] $GLVl_3$, in the S3 situation

III. We add one more injection point:

[A13.] in situation S1 with smaller depth than $GLVd_1$

[A14.] in situation S1 with greater depth than $GLVd_1$

[A15.] in situation S2 with smaller depth than $GLVd_1$,

[A16.] in situation S2 with middle depth between $GLVd_1$ and $GLVd_2$,

[A17.] in situation S2 with greater depth than $GLVd_2, i = 1, 2$.

The general aim of this paper is to give the producer the feedback for the pre-completion process about the correct placement of GLV and consider actions $A1, A7, A13, A14$ in the situation S1, $A2, A3, A8, A9, A15, A16, A17$ in the situation S2 and $A4, A5, A6, A10, A11, A12$ in the situation S3 to obtain the desired change of the production OIL (POIL). We do not consider more than four GLV.

4 Models

Using the R language we model the following dependence types:

Type 1:

$$R1 = f(dPOIL, POIL, BHP, GLVd_i, GLVl_i, PIL_i, PID_i, PIE_i, PIV_i, i = 1, 2, 3, 4),$$

Type 2:

$$R2 = g(dPOIL, POIL, BHP, GLVd_i, GLVl_i, PIL_i, PID_i, PIE_i, PIV_i, i = 1, 2, 3, 4),$$

Type 3:

$$\begin{bmatrix} R1 \\ R2 \end{bmatrix} = \begin{bmatrix} f(dPOIL, POIL, BHP, GLVd_i, GLVl_i, PIL_i, PID_i, PIE_i, PIV_i, i = 1, 2, 3, 4), \\ g(dPOIL, POIL, BHP, GLVd_i, GLVl_i, PIL_i, PID_i, PIE_i, PIV_i, i = 1, 2, 3, 4). \end{bmatrix}$$

As the possible f and g function we consider:

lm Although the linear regression allows investigating only the linear dependence, using the PIL, PIS, PID and PIV transformation (cf. (3.1)) we investigate some nonlinear types of dependencies also.

the neural net *nnet* Described in [23] models for feed-forward neural networks with a single hidden layer and for multinomial log-linear (R library **nnet** with 9 neurons in one hidden layer and linear function as output).

the SNNS neural network The Neural Networks in R using the Stuttgart Neural Network Simulator (SNNS) described in [26] and [25] (*mlp, jordan, rbf, elman* from R library **RSNNS** with 9 neurons in hidden layer),

ppr The projection pursuit regression model developed in [10] (*ppr* in R library **stats**) consists of linear combinations of non-linear transformations of linear combinations of explanatory variables.

spline regression model The multivariate adaptive polynomial spline regression (*polymars* from library **polspline**) [13, 9, 21]. An adaptive regression procedure using piecewise linear splines to model the response.

svm Support vector machines model (*svm* from library **e1071** with $C = 1, \gamma = 0.0476, \epsilon = 0.1$ and radial kernel). The review of svm methods is presented in [1] whereas the used algorithm in the C++ *libsvm* library is described in [5].

MART . The stochastic Multiple Additive Regression Trees method (*MART* has been taken from the library **gbm**) as the implementation of the gradient tree boosting methods for predictive data mining (regression and classification). For a quick overview of the methodology and implementation cf. [8] is the main reference.

Because *svm* and *MART* methods do not work in two dimensions, they are omitted regrading the type 3. In the other methods *nnet, mlp, jordan, rbf, elman* we normalized the responses and predictors. For the applying of these methods, the renormalization procedure should be used. In the *MART* method the relative validations of used predictors is additionally computed. We summarize these results in Table 3. All models were obtained by the cross-validation test.

As it is shown, it is not possible to make the "universal" model for all situations A1-A12, because the differences are too big.

Comparizing the computed fitted values $\{y_i, 1 \le i \le n\}$ of some method (*lm,nnet,mlp,...*) with those "exact" obtained from PO model $\{x_i, 1 \le i \le n\}$ we can compute the standard deviation error as

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n} - \Big(\frac{\sum_{i=1}^{n}(x_i - y_i)}{n}\Big)^2}, \qquad (4.2)$$

and summarize these values in Table 4.

Dividing the reservoir values (BHP) on intervals $[0, 2500), [2500, 3500), [3500, 4500), [4500, \infty)$ and percentage reldPOIL computed according to $reldPOIL = \frac{|dPOIL|}{POIL} \times 100\%$ on intervals $(0, 0.05\%], (0.05\%, 0.1\%], (0.1\%, 100\%]$ we chose the best model for each interval. This result is presented on Table 5 (in parentheses there are standard errors in appropriate intervals).

Table 5 shows the results for different actions related BHP of the algorithm of steering of gas and oil production described in the next section.

Table 3: Feature importance from MART model for 12 types of simulations denoted as A1 to A12. For details of the simulations see section 3

Attribute	Type											
	$A1$	$A2$	$A3$	$A4$	$A5$	$A6$	$A7$	$A8$	$A9$	$A10$	$A11$	$A12$
$POIL$	36.35	0.84	0.03	1.03	0.71	0.76	21.00	3.68	2.45	2.60	2.18	3.51
PID_1	14.81	17.27	1.78	0.26	0.27	0.30	9.74	27.36	5.64	36.49	0.39	0.51
$GLVd_1$	14.41	0.35	0.00	57.50	20.96	0.57	4.15	0.37	0.66	0.07	0.36	0.14
PIL_1	13.81	4.30	0.95	0.39	0.19	0.21	23.56	29.48	3.31	27.37	0.95	0.39
$dPOIL$	11.96	1.60	88.17	1.24	1.17	3.22	19.80	1.68	4.05	1.95	2.14	1.25
PIV_1	5.08	0.00	0.05	0.00	0.00	0.00	14.83	3.01	0.02	0.00	0.00	0.00
PIE_1	3.06	2.70	0.00	0.00	0.00	0.00	6.01	2.11	0.00	0.00	0.00	0.00
$GLVl_1$	0.49	5.08	0.38	0.01	0.00	0.01	0.30	11.28	0.71	26.08	0.01	0.01
BHP	0.02	0.02	0.00	0.04	0.05	0.09	0.63	0.29	0.19	0.12	0.11	0.16
PID_2	0.00	30.79	2.64	0.34	0.21	0.34	0.00	9.74	27.55	0.37	12.94	0.46
PIL_2	0.00	19.14	1.66	0.46	0.40	0.58	0.00	3.53	17.34	0.60	52.31	0.53
PIE_2	0.00	9.58	1.16	0.04	0.01	0.25	0.00	2.08	8.85	0.18	17.97	0.04
$GLVd_2$	0.00	5.07	0.17	37.12	41.45	38.71	0.00	0.81	0.24	0.52	0.78	0.38
$GLVl_2$	0.00	3.10	0.77	0.00	0.00	0.00	0.00	0.96	5.57	0.00	0.14	0.00
PIV_2	0.00	0.17	2.24	0.02	0.01	0.03	0.00	3.63	23.43	0.08	5.03	0.08
$GLVd_3$	0.00	0.00	0.00	0.58	34.23	53.73	0.00	0.00	0.00	2.38	1.40	0.20
PID_3	0.00	0.00	0.00	0.42	0.16	0.53	0.00	0.00	0.00	0.37	1.27	3.22
PIL_3	0.00	0.00	0.00	0.36	0.14	0.67	0.00	0.00	0.00	0.51	1.11	17.61
PIV_3	0.00	0.00	0.00	0.14	0.02	0.01	0.00	0.00	0.00	0.15	0.56	4.59
PIE_3	0.00	0.00	0.00	0.06	0.02	0.01	0.00	0.00	0.00	0.15	0.32	66.91
$GLVl_3$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01

5 The algorithm

On the web page
`http://matrix.umcs.lublin.pl/~akrajka/oilprod`
the following files may be found: `elman4, elman7, elman8, elman9, elman10, elman11, elman12, elman16, MART1, MART2, MART3, MART4, MART12, mlp2, mlp4, mlp5, mlp13, mlp15, mlp17, nnet1, nnet2, nnet3, nnet4, nnet5, nnet6, nnet10, nnet11, nnet12, nnet14, nnet15, nnet17, svm1, svm4, svm5, svm6, normalise` (normalization and renormalization parameters), `tab5`(values from Table 5) and `program.r` (program in R).

The input data for our algorithm are *POIL, newProdOIL, BHP, Situation, GLVd, GLVl*:

- $POIL$ is the actual production of oil,
- $newProdOIL$ as desired oil production,
- *BHP* as the actual bottomhole pressure,
- *Situation* as integer numbers 1, 2 or 3 according to the number of connected GLV (the situations S1, S2, or S3),

Table 4: The standard deviation error (4.2) of all models

Type	N	*lm*	*nnet*	*mlp*	*jordan*	*rbf*	*elman*	*ppr*	*polymars*	*svm*	*MART*
depth evaluation											
*A*1	2688	76.01	26.84	44.95	1039.26	79.48	50.02	44.19	47.63	51.88	35.38
*A*2	18529	240.32	226.38	227.14	229.26	240.02	234.78	229.17	227.99	230.63	227.23
*A*3	19087	253.03	221.58	222.98	226.20	253.74	232.49	226.41	225.75	228.97	221.54
*A*4	16992	137.04	131.30	131.45	132.63	139.72	132.13	136.53	132.51	131.08	131.61
*A*5	17030	139.27	131.13	131.90	231.29	154.60	691.91	138.56	136.18	133.73	135.13
*A*6	23581	172.61	167.91	170.99	169.90	176.49	169.54	170.25	170.42	168.13	170.66
lift evaluation											
*A*7	4279	0.969	0.902	0.941	1.187	1.015	0.586	0.961	0.936	0.958	0.898
*A*8	8158	1.072	1.025	1.041	1.014	1.141	0.744	1.055	1.065	1.047	1.031
*A*9	8299	1.093	1.047	1.068	1.094	1.141	0.865	1.074	1.088	1.072	1.055
*A*10	12140	0.667	0.653	0.664	0.664	0.692	0.655	0.666	0.667	0.665	0.665
*A*11	12397	0.666	0.655	0.664	0.676	0.679	0.663	0.665	0.666	0.672	0.671
*A*12	12077	0.665	0.654	0.663	0.670	0.753	0.658	0.665	0.665	0.674	0.671
depth evaluation											
*A*13	3332	201.59	201.00	201.56	143.36	201.97	256.15	201.36	201.62	−−	−−
*A*14	28615	658.07	446.98	475.00	719.87	653.76	702.95	507.54	517.54	−−	−−
*A*15	17441	192.57	190.95	191.93	198.92	193.16	194.66	192.54	192.63	−−	−−
*A*16	17421	191.89	189.64	190.21	174.81	196.79	140.28	191.87	191.41	−−	−−
*A*17	39437	308.29	287.26	290.64	306.08	307.42	18407.88	302.17	291.26	−−	−−
lift evaluation											
*A*13	3332	1.448	1.395	1.411	2.489	1.506	1.483	1.473	1.473	−−	−−
*A*14	28615	1.184	1.182	1.180	1.219	1.186	1.212	1.186	1.184	−−	−−
*A*15	17441	1.127	1.127	1.127	1.232	1.127	1.226	1.127	1.127	−−	−−
*A*16	17421	1.124	1.121	1.124	1.498	1.125	1.132	1.125	1.125	−−	−−
*A*17	39437	1.128	1.128	1.128	1.129	1.129	6.695	1.129	1.129	−−	−−

Table 5: Set of the chosen models according to the function of BHP and related dPOIL intervals

zest	reldPOIL	reservoir BHP			
		[0, 2500)	[2500, 3500)	[3500, 4500)	[4500, ∞)
A1	(0%, 0.05%]	nnet(16.2)	nnet(44.89)	MART(34.)	nnet(23.66)
A1	(0.05%, 0.1%]	MART(7.98)	MART(39.86)	svm(10.5)	nnet(19.66)
A1	(0.1%, 100%]	nnet(12.88)	MART(36.16)	MART(14.51)	MART(21.12)
A2	(0%, 0.05%]	nnet(230.15)	nnet(202.39)	mlp(192.46)	nnet(181.99)
A2	(0.05%, 0.1%]	MART(198.55)	nnet(192.51)	nnet(198.55)	nnet(200.93)
A2	(0.1%, 100%]	nnet(227.08)	nnet(232.23)	nnet(251.5)	MART(251.74)
A3	(0%, 0.05%]	MART(213.1)	nnet(207.04)	nnet(178.57)	nnet(168.03)
A3	(0.05%, 0.1%]	MART(201.78)	MART(202.2)	MART(186.74)	MART(184.55)
A3	(0.1%, 100%]	nnet(225.83)	nnet(236.21)	MART(238.8)	MART(245.29)
A4	(0%, 0.05%]	nnet(134.94)	MART(139.32)	MART(133.87)	mlp(130.47)
A4	(0.05%, 0.1%]	elman(124.63)	svm(129.75)	svm(129.19)	nnet(130.58)
A4	(0.1%, 100%]	svm(130.17)	svm(132.16)	svm(127.4)	svm(129.25)
A5	(0%, 0.05%]	svm(124.19)	nnet(133.)	nnet(132.99)	nnet(129.32)
A5	(0.05%, 0.1%]	svm(131.11)	mlp(134.15)	nnet(139.43)	mlp(130.89)
A5	(0.1%, 100%]	nnet(131.02)	nnet(128.06)	nnet(129.16)	mlp(133.27)
A6	(0%, 0.05%]	svm(170.771)	svm(172.442)	svm(165.024)	nnet(168.971)
A6	(0.05%, 0.1%]	nnet(162.)	svm(171.405)	nnet(168.165)	svm(170.335)
A6	(0.1%, 100%]	nnet(163.984)	nnet(167.438)	nnet(165.892)	nnet(171.7)
A7	(0%, 0.05%]	elman(0.407)	elman(0.645)	elman(0.597)	elman(0.983)
A7	(0.05%, 0.1%]	elman(0.285)	elman(0.698)	elman(0.798)	elman(0.879)
A7	(0.1%, 100%]	elman(0.331)	elman(0.488)	elman(0.376)	elman(0.272)
A8	(0%, 0.05%]	elman(0.647)	elman(0.684)	elman(0.852)	elman(0.682)
A8	(0.05%, 0.1%]	elman(0.912)	elman(0.576)	elman(0.743)	elman(0.881)
A8	(0.1%, 100%]	elman(0.755)	elman(0.704)	elman(0.715)	elman(0.746)
A9	(0%, 0.05%]	elman(0.758)	elman(0.845)	elman(0.914)	elman(0.963)
A9	(0.05%, 0.1%]	elman(0.729)	elman(0.847)	elman(0.915)	elman(0.884)
A9	(0.1%, 100%]	elman(0.808)	elman(0.795)	elman(0.922)	elman(0.71)
A10	(0%, 0.05%]	elman(0.661)	nnet(0.631)	elman(0.665)	nnet(0.641)
A10	(0.05%, 0.1%]	nnet(0.656)	nnet(0.674)	elman(0.642)	nnet(0.63)
A10	(0.1%, 100%]	nnet(0.663)	elman(0.637)	nnet(0.678)	elman(0.628)
A11	(0%, 0.05%]	nnet(0.631)	elman(0.614)	nnet(0.658)	elman(0.645)
A11	(0.05%, 0.1%]	nnet(0.632)	elman(0.629)	nnet(0.647)	nnet(0.683)
A11	(0.1%, 100%]	nnet(0.665)	elman(0.651)	elman(0.654)	elman(0.656)
A12	(0%, 0.05%]	nnet(0.635)	nnet(0.659)	nnet(0.614)	nnet(0.629)
A12	(0.05%, 0.1%]	nnet(0.648)	MART(0.655)	nnet(0.657)	nnet(0.666)
A12	(0.1%, 100%]	nnet(0.657)	elman(0.636)	elman(0.644)	elman(0.697)
A13	(0%, 0.05%]	mlp(161.8/1.09)	jordan(107.3/1.87)	jordan(105.3/2.06)	nnet(206.5/1.45)
A13	(0.05%, 0.1%]	mlp(193.9/1.43)	nnet(203.5/1.46)	nnet(194.8/1.41)	nnet(196.4/1.3)
A13	(0.1%, 100%]	nnet(208.9/1.38)	jordan(120.7/2.1)	jordan(98.7/1.93)	nnet(185.1/1.41)
A14	(0%, 0.05%]	nnet(468.2/1.12)	nnet(444.1/1.15)	nnet(464.9/1.15)	nnet(446.4/1.17)
A14	(0.05%, 0.1%]	nnet(398.4/1.31)	nnet(401.7/1.27)	nnet(448.2/1.27)	nnet(431./1.17)
A14	(0.1%, 100%]	nnet(471.3/1.22)	nnet(408.7/1.25)	nnet(467.2/1.22)	nnet(418./1.15)
A15	(0%, 0.05%]	nnet(195.8/1.14)	nnet(190.4/1.14)	nnet(189.5/1.12)	nnet(192.4/1.12)
A15	(0.05%, 0.1%]	nnet(189./1.11)	nnet(192.5/1.14)	nnet(194.1/1.17)	nnet(179./1.09)
A15	(0.1%, 100%]	nnet(190.6/1.14)	nnet(187.6/1.12)	nnet(184.8/1.1)	mlp(200.5/1.07)
A16	(0%, 0.05%]	elman(152.3/1.13)	elman(136./1.15)	elman(131.9/1.12)	elman(134.2/1.11)
A16	(0.05%, 0.1%]	elman(146.9/1.15)	elman(131.6/1.15)	elman(146.8/1.15)	elman(132.2/1.11)
A16	(0.1%, 100%]	elman(157.4/1.13)	elman(137./1.12)	elman(136.9/1.11)	elman(136.1/1.25)
A17	(0%, 0.05%]	nnet(302.2/1.13)	nnet(292.9/1.13)	nnet(290.2/1.13)	nnet(287.2/1.12)
A17	(0.05%, 0.1%]	nnet(277.3/1.12)	nnet(271.2/1.16)	nnet(276.5/1.11)	nnet(285.2/1.16)
A17	(0.1%, 100%]	nnet(284.2/1.13)	nnet(276.3/1.14)	nnet(267.8/1.12)	mlp(274.9/1.13)

- *GLVd* and *GLVl* are five elements vectors of the depth and lift (complete by *NA* value if not exists).

The result is the data frame (named *wyn*) which contains a few fields. Very first *type* as the integer between 1 to 17 represents the number of the possible action A1-A17. In the another one, fields *GLVdx*, *GLVlx* contain new depth, length or depth, and length value according to the type in the intervals $[1, 6]$, $[7, 12]$, $[13, 17]$ respectively. Fields *errdx* and *errlx* contains information about the "predicted" error, taken from Table 5.

Parallelly, to the written in R code program *program.r*, we show below (Algorithm 1) the written in pseudocode main idea of mentioned above program.

Algorithm 1: The algorithm of prediction of change of parameters for given change of Oil Production

```
/* Set tab5 array                                                    */
$tab5 \leftarrow read.table(tab5)$;
/* Set lmx, lMx, lmy, lMy arrays                                     */
$readRDS(normalise)$;
/* Set dane data frame                                               */
$dane_1 \leftarrow newProdOIL - POIL$;
$dane_2 \leftarrow BHP$;
$dane_3 \leftarrow POIL$;
for $n = 1$ to 5 do
    $dane_{3+n} \leftarrow GLVd_n$;
    $dane_{8+n} \leftarrow GLVl_n$;
    $dane_{13+n} \leftarrow GLVd_n \times (1 + GLVl_n)$;
    $dane_{18+n} \leftarrow GLVd_n \times \exp(GLVl_n)$;
    $dane_{23+n} \leftarrow GLVd_n/(1 + GLVl_n)$;
    $dane_{28+n} \leftarrow \log(GLVd_n)/(1 + GLVl_n)$;
/* The possible actions A1-A17 depending on the Situation
   parameter we compute in wynz whereas in xpred are numbers of
   columns of data frame dane which should be used to
   prediction                                                        */
switch Simulation do
    case 1 do
        $wynz \leftarrow \{1, 7, 13, 14\}$;
        $xpred \leftarrow \{1, 2, 3, 4, 9, 14, 19, 24, 29\}$;
    case 2 do
        $wynz \leftarrow \{2, 3, 8, 9, 15, 16, 17\}$;
        $xpred \leftarrow \{1, 2, 3, 4, 9, 5, 10, 14, 15, 19, 20, 24, 25, 29, 30\}$;
    case 3 do
        $wynz \leftarrow \{4, 5, 6, 10, 11, 12\}$;
        $xpred \leftarrow$
          $\{1, 2, 3, 4, 9, 5, 10, 6, 11, 14, 15, 16, 19, 20, 21, 24, 25, 26, 29, 30, 31\}$;
/* For each type in wynz make prediction and write result            */
$twyn \leftarrow 1$;
```

```
/* Prepare pred and pred1                                  */
$pj \leftarrow 1$;
for $j \in xpred$ do
    $pred_{pj} \leftarrow dane_j$;
    $pred1_{pj} \leftarrow (pred_{pj} - lmx_{e,j})/(lMx_{e,j} - lmx_{e,j})$;
    $pj \leftarrow pj + 1$
/* Find and read method                                    */
$reldPOIL \leftarrow abs(newProdOIL/POIL - 1)$;
switch $reldPOIL$ do
    case $reldPOIL < 0.05$ do $nrel \leftarrow 1$;
    case $reldPOIL \geq 0.05$ and $reldPOIL < 0.1$ do $nrel \leftarrow 2$;
    case $reldPOIL \geq 0.1$ do $nrel \leftarrow 3$;
switch $BHP$ do
    case $BHP < 2500$ do $nres \leftarrow 1$;
    case $BHP \geq 2500$ and $BHP < 3500$ do $nres \leftarrow 2$;
    case $BHP \geq 3500$ and $BHP < 4500$ do $nres \leftarrow 3$;
    case $BHP \geq 4500$ do $nres \leftarrow 4$;
/* Set model from tab5                                     */
$model\_name \leftarrow tab5_{3*(e-1)+nrel,nres}$;
$model\_name3 \leftarrow substr(model\_name, 1, 3)$;
$err1 \leftarrow tab5_{3*(e-1)+nrel,nres+4}$;
$err2 \leftarrow tab5_{3*(e-1)+nrel,nres+8}$;
/* Read model from disk and predict                        */
$model \leftarrow readRDS(model\_name)$;
if $model\_name3 \in \{"nne", "elm", "mlp"\}$ then
    $wynx \leftarrow predict(model, pred1) * (lMy[1] - lmy[1]) + lmy[1]$
else
    $wynx \leftarrow predict(model, pred)$
$wyn_{twyn,1} \leftarrow e$;
$wyn_{twyn,4} \leftarrow err1$;
$wyn_{twyn,5} \leftarrow err2$;
switch $e$ do
    case $e < 7$ do
        $wyn_{twyn,2} \leftarrow wynx$;
        $wyn_{twyn,3} \leftarrow NA$;
    case $e \geq 7$ and $e < 13$ do
        $wyn_{twyn,2} \leftarrow NA$ ;
        $wyn_{twyn,3} \leftarrow wynx$ ;
    case $e \geq 13$ do
        $wyn_{twyn,2} \leftarrow wynx_1$;
        $wyn_{twyn,3} \leftarrow wynx_2$;
$twyn \leftarrow twyn + 1$;
```

6 The conclusions

We studied different data mining models to achieve the expected production results. The presented algorithms are so far different than the typical wellbore simulators and employed in open libraries. Based on our research we conclude:

1. GLV depths values ($GLVd_i, i = 1,2,3$) has the significant influence on the final results $PID_i, i = 1,2$ as it is expected. Different injection changes PIL_i and $PID_i, i = 1,2$ values. There are not equality validation parameters for all situations. Situations are essentially different.

2. The worst standard error was obtained in situations $A13 - A17$, because changes are more significant. Some evaluations of standard deviation errors eliminate used methods. In some cases, the difference seems to be small (on example $A6$) but if we look on the number of considered cases N there are still meaningful.

3. For the situations $A7 - A12$ the *elman* networks are essentially preferred. For cases $A1 - A6$ *nnet* and *MART* methods are promising. For situations $A13 - A17$ all type of networks returns satisfactory results. It is impossible to choose the global method that meets the criteria for all actions. It concludes to further investigations in this direction.

4. Error in Table 5 is not very small, but once we analyse the vector of error values, the analysis is failed by the rare output values. It should be analysed in the next paper.

References

[1] Bennett, K. P. and Campbell, C. (2000). Support vector machines: hype or hallelujah? *ACM SIGKDD explorations newsletter,* 2(2), 1-13. http://www.acm.org/sigs/sigkdd/ explorations/issue2-2/bennett.pdf.

[2] Biecek P. (2017). *Przewodnik po pakiecie R*, Oficyna Wydawnicza "GIS".

[3] Bielecki, B. and Krajka, A. (2015). The framework dedicated to three phase flows wellbore modelling,*Mathematical Problems in Engineering, 2015,* (http://dx.doi.org/10.1155/2015/183982)

[4] Bielecki, B., Ksieżopolski, B., Krajka, A. and Wierzbicki, A. (2014). The Concept and Security Analysis of Wireless Sensor Network for Gas Lift in Oilwells. *Annales Universitatis Mariae Curie-Skłodowska. Sectio AI, Informatica, 14(2).*

[5] Chang, C. C. and Lin, C. J. (2012). LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, detailed documentation (algorithms, formulae, ...) can be found in http: //www.csie.ntu.edu.tw/ cjlin/papers/libsvm.ps.gz

[6] Electronical textbook: `http://www.statsoft.com/textbook/`

[7] Fausett L., *Fundamentals of Neural Networks.* New York: Prentice Hall. 1994.

[8] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics,* 1189-1232.

[9] Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics,* 1-67.

[10] Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American statistical Association,* 76(376), 817-823.

[11] Haykin, S. and Network, N. (2004). A comprehensive foundation. *Neural networks,* 2(2004), 41.

[12] The home page of language R: `http://cran.r-project.org/` .

[13] Kooperberg, C., Bose, S. and Stone, C. J. (1997). Polychotomous regression. Journal of the American Statistical Association, 92(437), 117-127.

[14] Nisbet R., Elder J., and Miner, G., *Handbook of Statistical Analysis and Data Mining Applications.* Burlington, MA: Academic Press (Elsevier) 2009.

[15] Patterson D., *Artificial Neural Networks.* Singapore: Prentice Hall 1996.

[16] Pourafshary, P. (2007). *A coupled wellbore/reservoir simulator to model multiphase flow and temperature distribution.* The University of Texas at Austin.

[17] Ripley B. D., *Pattern Recognition and Neural Networks.* Cambridge University Press 1996.

[18] Saepudin, D., Sukarno, P., Soewono, E., Sidarto, K. A., Yodi, A., Gunawan, S. S. and Budicakrayana, Y. (2008). Optimization of gas injection allocation in multi gas lift wells system. In *Sl]: Proceedings of the International Conference on Engineering Optimization.*

[19] Shirdel M., *Development of a Coupled Wellbore — Reservoir Compositional Simulator for Damage Prediction and Remediation.* Dissertation Presented to the Faculty of the Graduate School of The University of Texas at Austin in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy. The University of Texas at Austin. August 2013

[20] *Statistics in R,* `http://zoonek2.free.fr/UNIX/48_R/all.html`

[21] Stone, C. J., Hansen, M., Kooperberg, C. and Truong, Y. K. (1994). The use of polynomial splines and their tensor products in multivariate function estimation (with discussion. In Ann. Statist.)

[22] Takacs G. (2005). *Gas Lift Manual,* PennWell Books.

[23] Venables, W. N. and Ripley, B. D., *Modern Applied Statistics with S.* Fourth edition. Springer 2002.

[24] Walesiak M. and Gatnar E., *Statystyczna analiza danych z wykorzystaniem programu R,* Warszawa 2009, Wydaw. Nauk. PWN SA.

[25] Zell, A. (1994). *Simulation neuronaler netze* (Vol. 1, No. 5.3). Bonn: Addison-Wesley. (in German)

[26] Zell, A. et al. (1998), SNNS Stuttgart Neural Network Simulator User Manual, Version 4.2, *Institute for Parallel and Distributed High Performance Systems, Technical Report,* 1995, 6/95. (http://www.ra.cs.uni-tuebingen.de/SNNS/)

Spatial Databases and Their Use in Spatial Web Applications Based on the Exemplary Internet Service for Same Chosen Objects in the Old City in Zamość

Joannna Potiopa
Paweł Wiśniewski
Beata Bylina*

1 Introduction

Spatial data (e.g. satellite pictures) are a valuable source of information in numerous fields. Each object on the Earth possesses spatial information which can be utilized in making different decisions [10]. Simplicity in obtaining geographical data causes the increasing demand for Internet applications rendering accessible and projecting spatial information [14]. Nowadays it is common to publish two-dimensional or three-dimensional maps on the websites. Especially development of the local infrastructure of the spatial data is not only advantageous for local communities but plays an important national role (e.g. in tourism) [9].

The spatial data saved in the file GeoJSON can be displayed on an Internet map by means of the JavaScript open-source library such as Leaflet or OpenLayers or by means of Google Maps API (Application Programming Interface). Data processing can be done on the client's computer by means of the library JavaScript Turf.js. However, if there is a need for making changes in the data or going over the changes made by others in real time, the data should be stored in the database on the server [7, 8].

At present in many systems managing data including the open-source ones, there are available spatial components that allow storing the information about the objects location e.g PostGIS in PostgreSQL, MySQL Spatial, SpatiaLite for SQLite. Development of programming languages and appearance of new technologies cause that the creators of Internet applications do not need to know the language SQL

*Corresponding author — beata.bylina@umcs.pl

(Structured Query Language) to manage the data collected in the database. This is possible due to the object-relational mapping (ORM).

The paper presents an outline of the implementation of Internet service for management and presentation of the spatial data based on the exemplary Old City in Zamość. We will focus mainly on the application back-end where Python will be used as the language for creating the server-side scripts. As a support for the web application creation, we will make use of the framework Flask which is characterized by limited functionality in itself but ensures a very flexible environment for individual creation of dynamic Internet websites. The data will be stored in the database PostgreSQL with its spatial extension PostGIS. Additionally, the mechanism ORM will be used for setting up the database and manipulating the data.

Chapter two discusses the technologies used in the service creation process. Chapter three presents a shortened description of application back-end building. The paper ends with a brief summary.

2 Methodology

2.1 Database

Since the 60s of the last century, databases have constituted an essential element of most computer science systems. For ages, technological progress has resulted in inventing new applications for database systems and creating their new types.

2.2 Relational databases

The conception of the relational database was introduced by Edgar Frank Codd [5] who considered the existing hierarchical and network data model to be primitive for the creation of effective applications. His model was based on reliable mathematical principles: conception of mathematical relation, theory of sets, first order logic, and predicates calculus. The combination of precision and simplicity of this model caused that since that time the relational model has been implemented in a huge number of database systems. At present, the most common [1] commercial database management systems (DBMS) are: Oracle[1] (Oracle firm), DB2[2] (IBM firm), SQL Server[3], and Access[4] (Microsoft firm). Moreover, there are available systems of the open access to the source code: MySQL[5], PostreSQL[6] and SQLite[7].

At first mainly text data were stored in the relational database. The basic types of data include numerical types, strings, bit sequences, logical values, date and time. At present, there can be stored many other types of data: documents, images, voice, video.

[1]https://www.oracle.com/database/
[2]https://www.ibm.com/analytics/db2
[3]https://www.microsoft.com/en-us/sql-server/
[4]https://www.microsoft.com/en-us/microsoft-365/access
[5]https://www.mysql.com/
[6]https://www.postgresql.org/
[7]https://www.sqlite.org/index.html

2.3 Spatial databases

The development of computer science technologies as well as measurement tools forced the need for storing and processing spatial data. The programs dedicated to the geographical information systems (GIS) such as ArcGIS[8], QGIS[9], Saga GIS[10] can operate independently main tasks including introduction, verification, storage, analysis, and visualization of such data. However, due to Internet development, it is necessary to store the information about the objects, which are described by the characteristics related to the spatial location in the relational databases. In order to accomplish this, there appeared the extensions such as Oracle Spatial[11], Spatial SQL Server[12], PostGIS[13] (PostgreSQL), SpatiaLite[14] (SQLite) in the DBMS. They include the dedicated types of data for spatial information storage, spatial SQL queries, and spatial indices [13].

2.4 PostreSQL/PostGIS

PostgreSQL is an object-relational database management system (ORDBMS), it works under numerous operating systems (Linux, Windows, FreeBSD, Solaris, Mac OS X). This is the open-source type software comparable with the commercial one such as Oracle or DB2 [11].

PostGIS is an extension of the system PosgreSQL which enables storage of geographical objects in the database, it renders accessible spatial types, indices, and functions. This software is consistent with the specification „OpenGIS Simple Features" given by Open Geospatial Consortium[15] (OGC) which is the international standard of contents and geospatial services organization [6]. Standard SFSQL defines the relationship between the geometries such as intersection, sum, difference, symmetric difference.

PostGIS includes the geometry type data consistent with OGC which can be seen in Fig. 1. The spatial data types are organized into the type hierarchy. Each subtype inherits the structure (attributes) and behavior (methods and function) of its supertype.

If we want to place a spatial object in the database we have to give additionally the number SRID (Spatial Referencing System Identifier) which identifies explicitly the coordinate system (Spatial Referencing System, SRS). For the system WGS84 SRID = 4326. SDRID = −1 indicates that the coordinate system is not defined.

[8]https://www.arcgis.com/index.html
[9]https://qgis.org/pl/site/
[10]http://www.saga-gis.org/en/index.html
[11]https://www.oracle.com/database/spatial/
[12]https://docs.microsoft.com/pl-pl/sql/relational-databases/spatial/spatial-data-sql-server?view=sql-server-ver15
[13]https://postgis.net/
[14]https://www.gaia-gis.it/fossil/libspatialite/index
[15]http://www.opengeospatial.org/

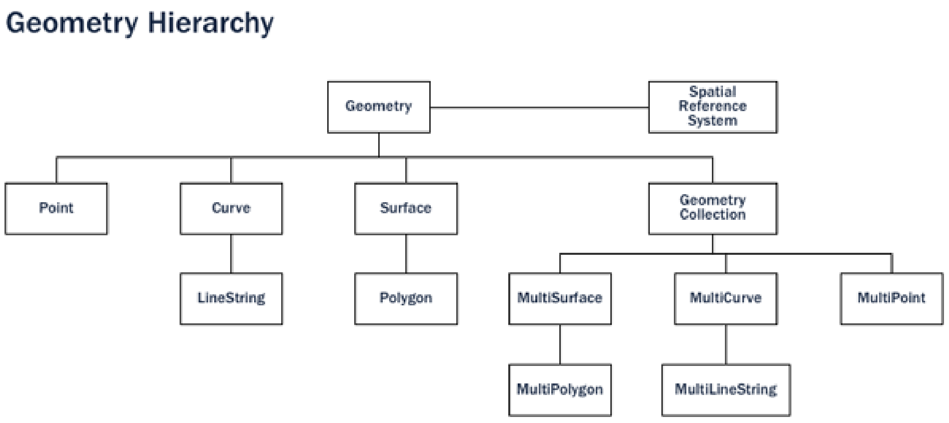


Figure 1: Spatial types of data in PostGIS [2]

2.5 Python

Python [16] is the interpreted high-level general-purpose programming language operating on various platforms (Windows, Mac, Linux, Raspberry Pi, etc.). It has a simple syntax similar to the English language. It allows writing shorter programs than using other programming languages. Its characteristic feature is the use of indentations for separating blocks of code. Python accomplishes many paradigms of programming. It can be used as a structured, object-oriented, or functional language. It is flexible as it offers many functionalities by means of modules.

2.6 Flask

Python as a universal language is not adjusted to creating websites. As it provides a few frameworks for building Internet applications [17], it can be used for writing efficient website applications operating on the server side (that is back-end) [4]. Flask [12] is one of the micro-frameworks which allows combining some chosen Python libraries owing to which it is flexible and customizable. As it is only slightly dependent on the external libraries, it is possible to decide how to build an application by choosing suitable extensions.

2.7 Peewee

For many years the traditional way of communication with the servers of relational databases was the language SQL (Structured Query Language) and its knowledge is indispensable to make use of all possibilities of the databases. However, in many projects, the databases can be operated by means of the ORM (Object-Relational Mapping) [3]. ORM is a programming technique mapping the objects of a given programming language on the structure of the relational database. Using tables in an object-oriented way is more convenient in building the application

[16] https://www.python.org/
[17] https://www.jetbrains.com/lp/python-developers-survey-2020/#FrameworksLibraries

logic. The programmer can write a code e.g. in the Python language instead of queries in SQL for creating, reading, updating, and deleting data and diagrams in the database.

Peewee[18] is one of the simple and small ORM systems written in Python. It allows operating on the database in a simple way without SQL and knowledge of its specificity.

3 Results

We will present a simple project of Internet service for the presentation of spatial data combining a few different open-source technologies. The objects in the Old City in Zamość will be used as an example of data. In the service creation, there were considered the two assumptions: rendering complex management of the geographical data possible in the form of pointwise objects from the application level as well as their presentation on the map.

In order to enable modification of the data maintaining the possibility of introduced data verification, the users' accounts system was implemented. This allows adjusting the application functionality to the user's permissions. In order to use the service, one should register in it. The registered user is able to view the data but after being given the appropriate rights by the administrator he is able to edit the data and administer the service. The view of the application home page for the user with the edition and administration rights is presented in Figure 2.

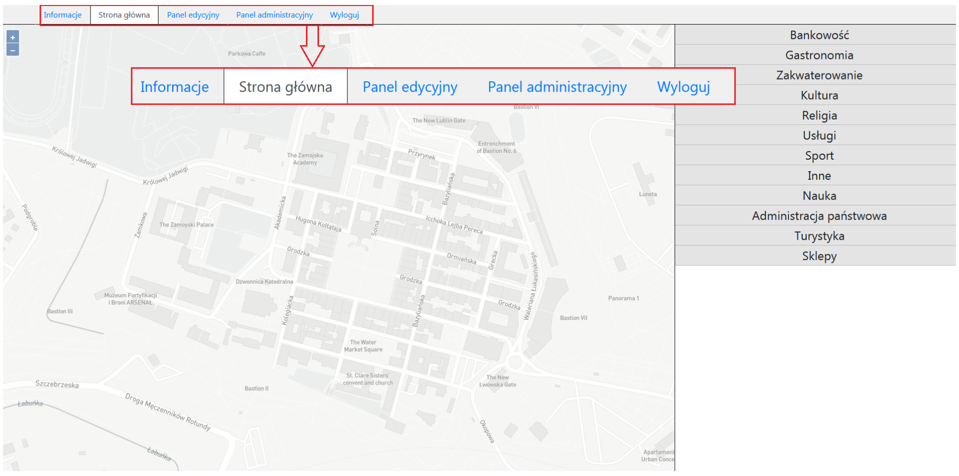


Figure 2: Application home page for the user with editing and administration rights

The main task of the objects presentations on the map is accomplished by rendering the map panel possible for the data display according to their assignment to the corresponding subcategories grouped into categories. The logged-in person can view the geographical objects in the database according to the categories (e.g. touristic objects, shops, restaurants, etc.) and subcategories (more detailed division

[18]http://docs.peewee-orm.com/en/latest/

of the objects from a given category e.g in the category shops, there are, among others, bookshops, chemist's, grocer's, confectioner's, etc.) on the home page. Each subcategory possesses its own display style on the map and the application enables display of any number of objects group at the same time as can be seen in Figure 3. The user can also display the object characteristics such as: name, description, category, subcategory as well as the author by clicking on the chosen point on the map. The user with granted suitable rights can edit these data. Some of them can also administer the users and their privileges.

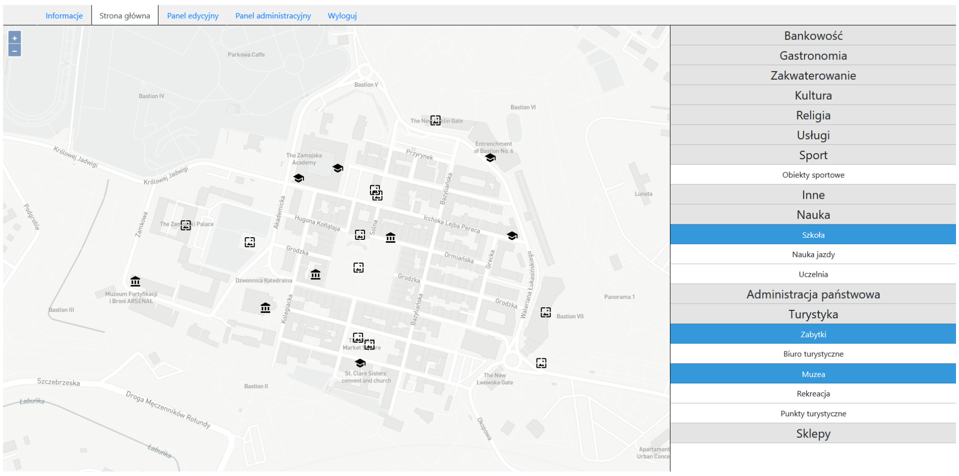


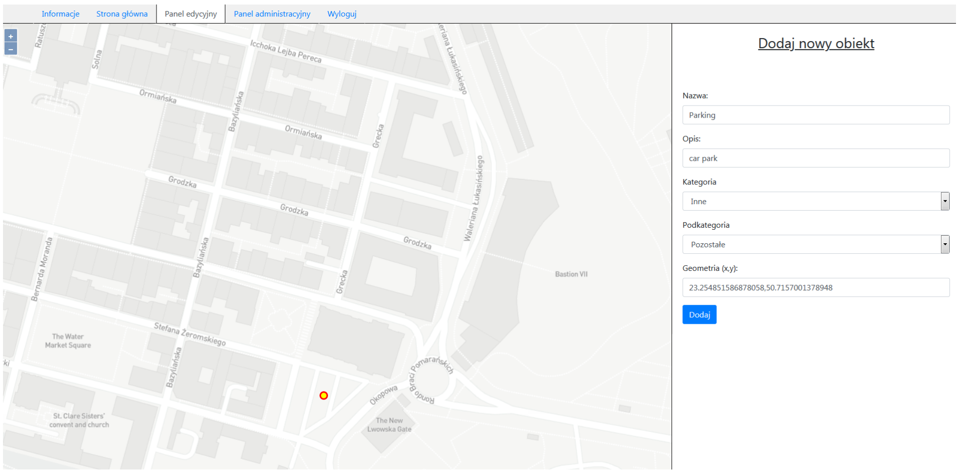
Figure 3: Map with visible objects belonging to chosen subcategories

The edition panel allows editing, deleting and creating new categories, subcategories and objects. Figure 4 shows the most extended form of new object addition. The website adding a new object is composed of a map on which one can indicate the point position and a form which can be completed with the values of individual object attributes. The object geometry can be given by entering the point coordinates in the format `x, y` in the form or indicating it on the map.

The application was assumed to store not only the data about users but mainly the information about geographical objects. Therefore the relational database PostgreSQL along with its spatial extension PostGIS was used.

The application under consideration is to make use of the database of not very complicated structure and programmed in the object-oriented language. Thus management of the database can be accomplished by means of the Peewee system — the simple and small ORM system written in the Python language. Utilization of the ORM system must be composed of three stages:

- making a connection to the database
- declaration of the model describing the base and creation of database structure
- performing the database operation



Dodaj nowy obiekt

Nazwa:
Parking

Opis:
car park

Kategoria
Inne

Podkategoria
Pozostałe

Geometria (x,y):
23.254851586878058,50.7157001378948

Dodaj

Figure 4: Addition of a new object

The conception of object-relational mapping consists in the creation of suitable tables, columns in the database taking into account their types and connections based in the classes written in the object-oriented language. Inheriting from a suitable base class, we declare the classes representing the tables in the database in the file `models.py` and the properties of the classes correspond to the columns in the tables. As an example the code of the model (taken as a class and its properties) of the class defining the table `points` is shown in Listing 1:

Code Listing 1: Exemplary class model

```
from peewee import PostgresqlDatabase, Model,
    PrimaryKeyField, TextField, DateTimeField,
    ForeignKeyField, Field, fn


class PointField(Field):
    db_field = 'geometry'

    def db_value(self, value):
        formatted_value = value.replace(",", "␣")
        return fn.St_PointFromText('POINT({})'.format(
            formatted_value), 4326)

    def python_value(self, value):
        res = database.execute_sql("select␣st_x('{}'),␣st_y
            ('{}')".format(value, value))
        return res.fetchone()

database = PostgresqlDatabase(None)
```

```
class AbstractModel(Model):
    class Meta:
        database = database

class Points(AbstractModel):
    id = PrimaryKeyField()
    name = TextField(unique=True, null=False)
    description = TextField(null=True)
    connection = ForeignKeyField(Connections)
    geometry = PointField()
    user = ForeignKeyField(Users)

    class Meta:
        db_table = 'points'
```

The class `Field()` is used for indicating the type of data to be stored in the column. Each type of field has a corresponding type of data in the database. The field `TextField()` is used for the text storage, by means of the field `PrimaryKeyField()` it is possible to determine which field will be a primary key. We can also define the additional features of field such as e.g. permission for null values (`null = False`) or unique values (`unique = True`). The relationships between tables are given making use of the field of the type `ForeignKeyField (the name of the related class)` **related-class-name**. The class `Points` possesses also the field geometry for the object location storage by means of the earlier created type `PointField()`.

The systems ORM including the exemplary Peewee create corresponding tables, columns with their types and relationships in the database based on the classes. The schema of the database in our application, made in the DBeaver[19] software is presented in Figure 5.

The database is composed of five related tables: *users*, *points*, *connections*, *categories*, *subcategories* as well as tables *logs* and *spatial_ref_sys*. The tables *users* stores the users' data together with their permissions, the tables *categories* and *subcategories* store the information about possible objects categories and subcategories, respectively, and the table *connections* connects categories and subcategories. In the table *points*, the information about the objects added to the database is stored. The name of the objects, its category, information about the user who made the object modification as the last one as well as the point location on the map given as type `point` are saved. Diagram 5 shows the table *spatial_ref_sys* — this is a special table of metadata defined in PostGIS in which the codes identifying each coordinate system specified by OGC are stored. It is created automatically by ORM. There was also made the table *logs* which stores the information about activities taking place in the Internet service.

[19]DBeaver (`https://dbeaver.io/`) — Free multi-platform database tool for developers, database administrators, analysts, and all people who need to work with databases. Supports all popular databases: MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, etc.

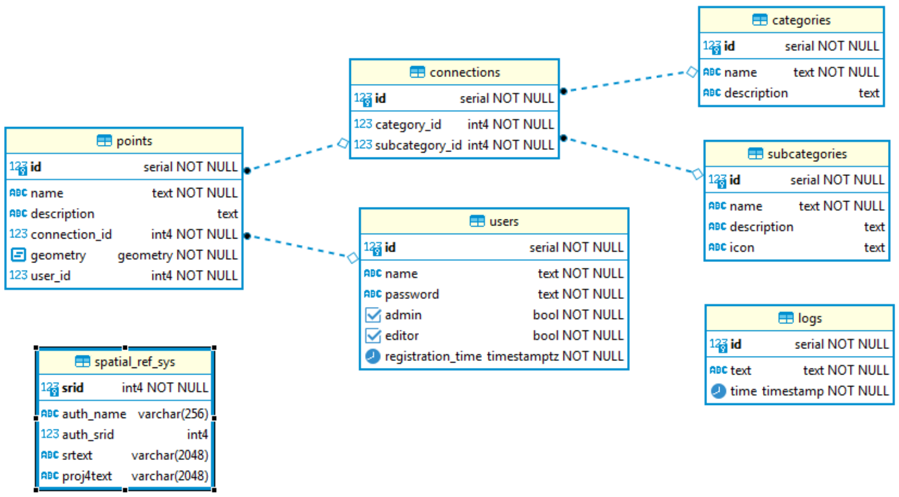


Figure 5: The database schema

It is possible to make operations CRUD (C — Create, R — Read, U — Update, D — Delete) using the created database. The basic operations made on the database, e.g. inserting or reading the data are performed in Peewee by means of the object representing the defined tables as well as their methods:

- `Model.create()` — the method allowing insertion of a new row into the database (equivalent `insert` in SQL)
- `Model.update()` — allows modifying the data in the existing row (equivalent `update` in SQL)
- `Model.delete()` — enables deletion of the row (equivalent `delete` in SQL)
- `Model.get()` — returns a single record matching the query (equivalent `select` with a condition)

Calling these methods as arguments one should give the data corresponding to the fields in the table. The example of the insertion of a new point into the database exploiting the data obtained from the form completed by the user in the application is given in Listing 2.

Code Listing 2: Insertion of a new point

```
# obtaining information about the subcategory of a point
connection_id = Connections.get(Connections.subcategory ==
    request.form['subcategory']).id
# obtaining information about the data of the user
    inserting an object
user = Users.get(Users.name == session.get('login')).id
```

```
# adding a new object to the database
point = Points.create(name=request.form['name'],
    description=request.form['description'], connection=
    connection_id, geometry=request.form['geometry'], user=
    user)
```

4 Conclusion

The paper shows that the web-GIS application does not need to use the spatial data only in a static way (e.g. displaying the location of the objects saved in the GeoJSON file). Introducing the relational database PostgreSQL with the possibility of spatial data storage thanks to PostGIS, the service renders the user the possibility of data edition. Owing to that all changes made in the databases are visible immediately for other users. It was also pointed out that Python can be an excellent choice as the application back-end as a result of the Flask framework exploitation. Additionally, it became possible to manage the spatial database not entering into details of the language SQL but in the Python language by means of ORM.

Contrary to the applications made in the dedicated GIS software, Internet application visualizing spatial data offers more advantages such as independence of the platform, user-friendliness, and common access. Moreover, owing to the use of open-source technology creation of such applications does not require any expenditure of money for software purchase. It was possible to create the application which cooperates with advanced database management systems and offers the user the dynamic Internet website for editing also spatial data. In the future, the application can be developed by the possibility of searching objects in a definite distance from the given location based on the spatial queries.

References

[1] Db-engines ranking. https://db-engines.com/en/ranking. Accessed: 2021-05-30.

[2] Introduction to PostGIS. https://postgis.net/workshops/postgis-intro/.

[3] Scott Ambler. Mapping Objects to Relational Databases: O/R Mapping In Detail, 01 2006.

[4] Fankar Aslam, Hawa Mohammed, and Prashant Lokhande. Efficient Way Of Web Development Using Python And Flask. *International Journal of Advanced Research in Computer Science*, 6, 01 2015.

[5] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387, June 1970.

[6] Open Geospatial Consortium. Simple Features SQL. https://www.ogc.org/standards/sfs.

[7] Paul Lewis, Conor Mc Elhinney, and Timothy Mccarthy. LiDAR Data Management Pipeline; from Spatial Database Population to Web-Application Visualization. 07 2012.

[8] L.M. Mamai, M. Gachari, and G. Makokha. Developing a Web-Based Water Distribution Geospatial Information System for Nairobi Northern Region. *Journal of Geographic Information System*, 9:34–46, 2017.

[9] Slaven Marasovic and Željko Hećimović. Open Source Software and Local Spatial Data Infrastructure. 09 2014.

[10] Riccardo Mari, Lorenzo Bottai, Caterina Busillo, Francesca Calastrini, Bernardo Gozzini, and Giovanni Gualtieri. A GIS-based interactive web decision support system for planning wind farms in Tuscany (Italy). *Renewable Energy*, 36(2):754–763, 2011.

[11] Pedro Martins, Paulo Tomé, Cristina Wanzeller, Filipe Sá, and Maryam Abbasi. Comparing Oracle and PostgreSQL, Performance and Optimization. In Álvaro Rocha, Hojjat Adeli, Gintautas Dzemyda, Fernando Moreira, and Ana Maria Ramalho Correia, editors, *Trends and Applications in Information Systems and Technologies*, pages 481–490, Cham, 2021. Springer International Publishing.

[12] Kunal Relan. *Beginning with Flask*, pages 1–26. 09 2019.

[13] Grace Samson, Zhongyu Lu, Mistura Usman, and Qiang Xu. *Spatial Databases: An overview*, pages 111–149. IGI Global, United States, February 2017.

[14] Sunil Singh and Preetvanti Singh. Mapping Spatial Data on the Web Using Free and Open-Source Tools: A Prototype Implementation. *Journal of Geographic Information System*, 6, 11 2013.

Nested Loop Transformations on Multi- and Many-Core Computers With Shared Memory

Beata Bylina
Jarosław Bylina[*]

1 Introduction

Nested loops are an important structure bearing a great deal of the parallelism possibilities. However, to parallelize them efficiently, the programmer has to make some decisions about applying various strategies to enable proper parallelization, vectorization, and the cache utilization. An example of such loops are matrix algorithms, like the matrix multiplication or different kinds of factorizations, widely investigated in the literature [4, 5].

In the work [2] we studied four parallelizing strategies for nested loops on multicore architectures on the example of a factorization similar to the LU factorization, namely, the WZ factorization [7, 6, 8]. The WZ factorization has some nontrivial data dependencies and the compiler is not able to efficiently optimize the algorithm. We used four parallelism strategies for nested loops (dubbed in that article: *outer*, *inner*, *nested*, and *split*). For random dense square matrices with the dominant diagonal, we reported the execution time, the performance, the speedup of the WZ factorization for these four strategies of parallelizing nested loops and we investigated the accuracy of such solutions. The *outer* and *split* approaches achieved the best speedup. In [3] we evaluated these two strategies on Intel Xeon Phi 7120P.

In this work, we want to compare the behavior of selected strategies on CPU and on MIC and investigate some other loop transformations. The loop transformations can improve the speed of the algorithm by reducing loops' costs which can lead to better utilization of contemporary architectures. They play an important role in cache efficiency improvement, better vectorization, better parallel processing. Some common examples of loop transformations are (among others) loop interchange, strip-mining, loop tiling.

The contribution of this work is: investigating and comparison of nested loops on CPU and MIC; investigating two algorithms of different granularity; studying influence of vectorization, scheduling, memory layout, loop order, loop tiling — all

[*]Corresponding author — `jaroslaw.bylina@umcs.pl`

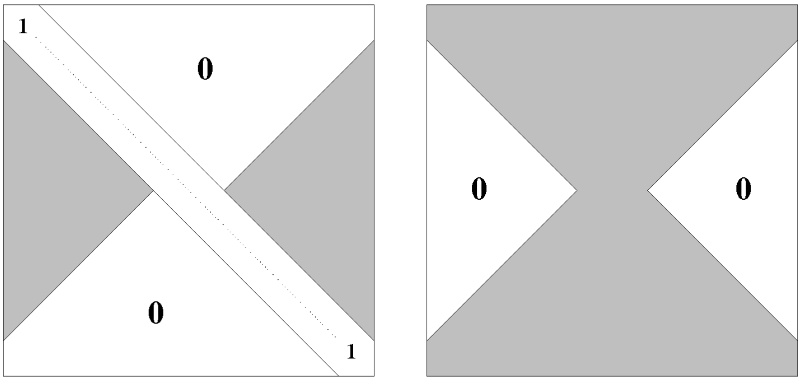


Figure 1: The output of the WZ factorization — forms of the matrices **W** (left) and **Z** (right)

of which can be significant and crucial.

The remainder of the paper is following. Section 2 presents the outline of the algorithm being a basis for tests, namely the WZ factorization. Section 3 describes variations of the algorithm which are investigated — and some analysis. In Section 4, we show how the tests have been made (environment, compiler, matrices). Section 5 describes details and results of experiments. Section 6 concludes the paper.

2 WZ factorization

We would like to present shortly the WZ factorization [7, 6, 8] which we chose here because of its quite complicated nesting of loops as well as parallel potential.

The matrix factorization is a manner to reduce a matrix to a product of two (or more) matrices of a simpler form. It is mainly used as an auxiliary operation for solving linear systems — after the factorization, we can solve more systems but their matrices are much more easy to operate than the initial one.

By the WZ factorization, we transform a square and nonsingular matrix **A** into a product of two matrices, namely **WZ**. The matrix **W** is a matrix of the form of a butterfly with units on its main diagonal, the matrix **Z** is a matrix of the form of an hourglass. Both the matrices are complements of each other in the sense of the structure of non-trivial elements (one has non-trivial elements in places where the other has zeros/units — and vice versa).

The forms of these matrices can be seen in Figure 1.

Prior to the start of the algorithm, the array `a` contains the matrix **A** which is about to be factorized. After the execution of this algorithm, we will obtain two arrays: the array `w` containing non-trivial elements of the **W** matrix (gray in Figure 1) and the array `a` containing non-trivial elements of the **Z** matrix (also gray in Figure 1).

Figure 2 presents a basic algorithm for the WZ factorization for an even size of the matrix (we only consider even sizes — without loss of generality).

```
for(k = 0; k < n/2-1; k++) {
    // the following four lines are omitted
    // in the next versions of the algorithms
    // (thay are always the same)
    p = n-k-1;
    akk = a[k][k];    akp = a[k][p];
    apk = a[p][k];    app = a[p][p];
    detinv = 1 / (apk*akp - akk*app);
    for(i = k+1; i < p; i++) {
        w[i][k] = (apk*a[i][p] - app*a[i][k])
                  * detinv;
        w[i][p] = (akp*a[i][k] - akk*a[i][p])
                  * detinv;
        for(j = k+1; j < p; j++)
            a[i][j] = a[i][j]
                      - w[i][k]*a[k][j]
                      - w[i][p]*a[p][j];
    }
}
```

Figure 2: The basic algorithm for the WZ factorization — pseudocode

We have three nested loops. The outermost loop (the `k`-loop) iterates over the number of steps. This number is half the size of the matrix minus 1 (that is, $\frac{n}{2}-1$). The kth iteration depends on the $k-1$st iteration (except for $k=0$).

The middle loop (the `i`-loop) is an inner loop (for `k`-loop) and an outer loop (for `j`-loop) at the same time. It computes elements of the matrix $\mathbf{W}$ in every iteration. Then, it executed the third loop. The innermost loop (the `j`-loop) updates the matrix $\mathbf{A}$. Both the inner loops (the `i`- and `j`-loops) have $n-2k$ iterations.

3 Algorithms and implementations

3.1 Sequential algorithms and loop interchange

The matrices $\mathbf{A}$ which we consider are dense ones. They are stored in two formats — a **column-wise** and **row-wise** manner. These are traditional schemes (layouts) for keeping dense matrices in the computer memory — known also from many standards and libraries — like BLAS [9], LAPACK [1], MKL [10], and others.

The loops' nesting order can be changed. This operation is known as **loop interchange (loop permutation)** and it can be done if it does not cause any changes in the computation results (that is, when there are no unsettled dependencies — or, in other words, all the dependencies are solved).

This is the issue with our original algorithm (Figure 2). There, we can exchange the `i`-loop with the `j`-loop with no changes to the results. However, in that algorithm, the inner loop (`j`-loop) is not the only instruction in the body of the outer loop (`i`-loop), so the interchange brings a lot of work duplication and thus is impractical — although, after the outer loop fission (see below), it becomes possible.

```
for(k = 0; k < n/2-1; k++) {
    . . .
    for(i = k+1; i < p; i++) {
        w[i][k] = (apk*a[i][p] - app*a[i][k])
                  * detinv;
        w[i][p] = (akp*a[i][k] - akk*a[i][p])
                  * detinv;
    }
    for(i = k+1; i < p; i++)
        for(j = k+1; j < p; j++)
            a[i][j] = a[i][j]
                      - w[i][k]*a[k][j]
                      - w[i][p]*a[p][j];
}
```

Figure 3: The algorithm after the fission of the `i`-loop — pseudocode. This algorithm matches the row-wise layout

To avert this duplication, we utilize an additional optimization technique on the `i`-loop — known as the **loop fission** (also called **loop distribution**).
Its result is shown in Figure 3.

3.2 Parallelization and vectorization

Next steps of optimization are vectorization and parallelization. Both the versions (the original one, Figure 2, and the fission one, Figure 3) have some potential for parallelization because they contain mainly loops.

These algorithms can be also vectorized with the use of `#pragma simd` or the `simd` clause in `#pragma omp parallel for`. The places to put them were investigated experimentally on multicore and manycore platforms in the above-mentioned papers [2, 3] and the best one was chosen. Moreover, removing `simd` clauses altogether causes a great performance drop (especially in fission algorithm).

Figure 4 shows the pseudocode of the parallelized and vectorized basic version. Figure 5 presents the pseudocode of the parallelized and vectorized fission algorithm. In the original algorithm (Figure 4), the `i`-loop is parallelized and the `j`-loop is vectorized.

3.3 Strip-mining and loop tiling

A loop in the process of strip-mining is divided into two loops, where the inner one has `BLOCK_SIZE` iterations and the outer one has `n/BLOCK_SIZE` iterations (`n` is the number of iterations in the original loop). The strip-mining alone can have some positive impact on the performance (by easing the automatic vectorization process).

In Figures 6 and 7, we use the compiler clause `__assume` which tells the compiler that a given condition is fulfilled — here, we declare that `ii` and `jj` are multiples of the `BLOCK_SIZE` — which facilitates the vectorization.

```
for(k = 0; k < n/2-1; k++) {
    . . .
#pragma omp parallel for
    for(i = k+1; i < p; i++) {
        w[i][k] = (apk*a[i][p] - app*a[i][k])
                  * detinv;
        w[i][p] = (akp*a[i][k] - akk*a[i][p])
                  * detinv;
#pragma simd
        for(j = k+1; j < p; j++)
            a[i][j] = a[i][j]
                      - w[i][k]*a[k][j]
                      - w[i][p]*a[p][j];
    }
}
```

Figure 4: The parallelized and vectorized version of the basic algorithm — pseudocode

```
for(k = 0; k < n/2-1; k++) {
    . . .
#pragma omp parallel for simd
    for(i = k+1; i < p; i++) {
        w[i][k] = (apk*a[i][p] - app*a[i][k])
                  * detinv;
        w[i][p] = (akp*a[i][k] - akk*a[i][p])
                  * detinv;
    }
#pragma omp parallel for
    for(i = k+1; i < p; i++)
#pragma simd
        for(j = k+1; j < p; j++)
            a[i][j] = a[i][j]
                      - w[i][k]*a[k][j]
                      - w[i][p]*a[p][j];
}
```

Figure 5: The parallelized and vectorized version of the fission algorithm — pseudocode

4 Testing methodology

Our testing machine consists of two CPUs and a MIC. The detailed parameters of the environment are shown in Table 1.

The language of the implementation was C++. The floating-point numbers were in double precision (the `double` type in C++). All the programs were compiled with the following compiler options:

- `-qopenmp` — it enables the compiler to generate multi-threaded code with

```
for(k = 0; k < n/2-1; k++) {
    . . .
#pragma omp parallel for
    for(i = k+1; i < p; i++) {
        w[i][k] = (apk*a[i][p] - app*a[i][k])
                  * detinv;
        w[i][p] = (akp*a[i][k] - akk*a[i][p])
                  * detinv;
        start = RDTTNM(k+1, BLOCK_SIZE);
        for(jj = start; jj < p;
                        jj += BLOCK_SIZE) {
            __assume(jj % BLOCK_SIZE == 0);
#pragma simd
            for(j = jj; j < jj+BLOCK_SIZE;
                        ++j)
                a[i][j] = a[i][j]
                          - w[i][k]*a[k][j]
                          - w[i][p]*a[p][j];
        }
    }
}
```

Figure 6: Strip-mining in the basic algorithm — pseudocode

the use of OpenMP directives;

- `-O3` — this option optimizes aggressively for maximum speed;
- `-ipo` — it enables interprocedural optimization (also between files);
- `-no-prec-div` — it improves the speed of floating-point dividing, at the cost of its precision;
- `-fp-model fast=2` — this option enables some floating point optimizations — also at the expense of its precision.

These options are quite a standard set of the optimization manners. Moreover, some of them can have (nominally) some negative impact on the precision, however, some auxiliary tests showed that this influence is almost negligible in our algorithms.

Additionally, the CPU programs were compiled with the option `-xHost` and the MIC programs — with the option `-mmic`. These two options simply allowed generating codes for the correct processors.

All the programs were tested on CPU (run directly from the operating system) and MIC (run by the `micnativeloadex` command) with a various number of threads (set with the use of the `OMP_NUM_THREADS` environment variable), with no use of the hyperthreading on CPU. Three OpenMP scheduling manners were investigated, namely `static`, `guided`, `dynamic` (the last one with two chunk sizes: 1 and 10). The scheduling techniques were tested because of their different influence on the load balancing.

```
for(k = 0; k < n/2-1; k++) {
    . . .
#pragma omp parallel for simd
    for(i = k+1; i < p; i++) {
        w[i][k] = (apk*a[i][p] - app*a[i][k])
                  * detinv;
        w[i][p] = (akp*a[i][k] - akk*a[i][p])
                  * detinv;
    }
    start = RDTTNM(k+1, BLOCK_SIZE);
#pragma omp parallel for
    for(ii = start; ii < p;
                    ii += BLOCK_SIZE) {
        for(jj = start; jj < p;
                        jj += BLOCK_SIZE) {
            __assume(ii % BLOCK_SIZE == 0);
            for(i = ii; i < ii+BLOCK_SIZE;
                        ++i) {
                __assume(
                    jj % BLOCK_SIZE == 0);
#pragma simd
                for(j = jj; j < jj+BLOCK_SIZE;
                            ++j)
                    a[i][j] =
                          a[i][j]
                          - w[i][k]*a[k][j]
                          - w[i][p]*a[p][j];
            }
        }
    }
}
```

Figure 7: Loop tiling in the fission algorithm — pseudocode

Table 1: Hardware and software used in the experiments

	CPU	MIC
	2 × Intel Xeon E5-2670 v.3 (Haswell)	Intel Xeon Phi 7120 (Knights Corner)
# cores	24 (12 per socket)	61
# threads	48 (2 per core)	244 (4 per core)
clock	2.30 GHz	1.24 Ghz
level 1 instruction cache	32 kB per core	32 kB per core
level 1 data cache	32 kB per core	32 kB per core
level 2 cache	256 kB per core	512 kB per core
level 3 cache	30 MB	—
RAM memory	128 GB	16 GB
max. mem. bandwidth	68 GB/s	352 GB/s
SIMD register size	256 b	512 b
instructions' execution	out-of-order	in-order
compiler	Intel ICC 16.0.0	
BLAS/LAPACK libraries	MKL 2016.0.109	

All the implementations were tested on random dense matrices which had the WZ factorization with no pivoting needed. The sizes of the matrices were up to 12288×12288. However, only the biggest size is shown in Section 5.

All the results present the computational performance. As its unit we use Gflops, where flops is a floating-point operation per second. The number of floating-point operations for the WZ factorization of the matrix of the size $n \times n$ is $\frac{2}{3}n^3 - \frac{7}{6}n - 3$, so to obtain the metric in Gflops ($= 10^9$ flops) we divide the number of floating-point operations by $10^9 T$ — where T is the execution time of the measured implementation. Such a metric allows comparing all implementations with the same measure. The execution time was measured with the use of a standard OpenMP function `omp_get_wtime`.

5 Numerical experiments

5.1 Previous parallel implementations

First, as a basis for our optimization attempts, we tested parallel implementations from [3]. They are `basic` and `fission` (dubbed there 'outer' and 'split' because of the place and manner of parallelization). They were implemented with the column-wise representation and the `(i,j)` sequence of the loops. Thus, we have two types of implementations:

- `basic-col`
- `fission-col-ij`

Both the algorithms were tested both as explicitly vectorized — with the `simd` clauses in `#pragmas` — and without these clauses. However, the lack of explicit vectorization in the `fission` algorithm gave the performance several times slower on both platforms. On the other hand, the performance of the `basic` version does not depend on the explicit vectorization: the compiler detects the possibility and optimizes the code appropriately, because the source is simpler. Thus, we show only vectorized versions. Figures 8 and 9 show the performance of the `basic-col` version (on CPU and MIC, respectively) and Figures 10 and 11 show the performance of the `fission-col-ij` version (also on CPU and MIC, respectively).

All these tests performed rather poorly. On CPU, the `fission` implementation (all schedulings except the dynamic one with the chunk equal to 1) is better than the `basic`, although its performance does not exceed 6 Gflops. On MIC, the `fission` implementation is also better (for the same schedulings), but performs a little worse then on CPU (less then 5 Gflops). The performance of the `basic` version on CPU depends on the scheduling; although, on MIC, the differences are very minute. On the other hand, the `fission` version is very dependent on scheduling on both platforms. On CPU, the order of the schedulings (from the best to the worst) is: guided, dynamic (with the chunk size 10), static and dynamic (with the chunk size 1). On MIC, the situation is more complicated, also because the performance declines with the growth of the number of threads (here, there is no point in using more then one thread per core).

Thus, we can see that all three aspects (algorithm, vectorization, scheduling) can have some impact on the performance and all must be taken into account.

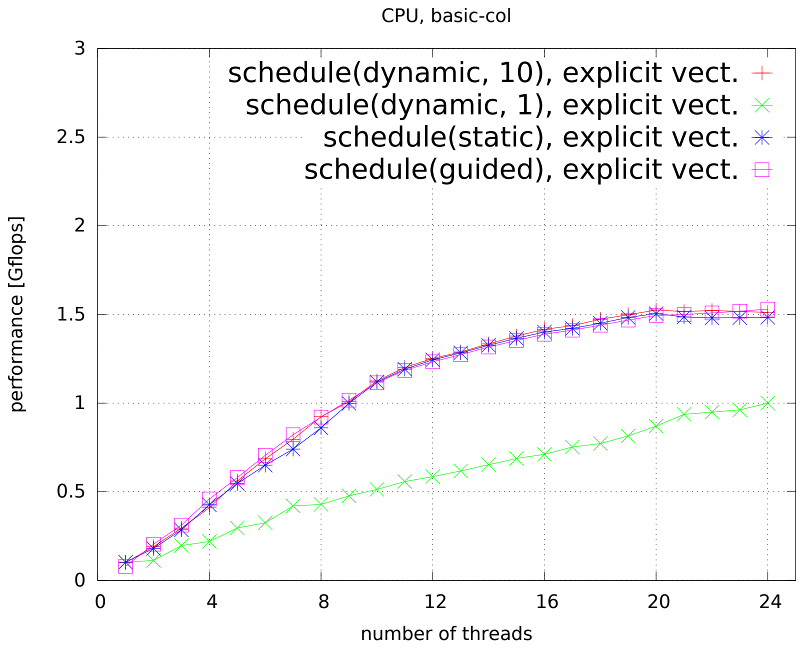


Figure 8: Manually vectorized `basic-col` on CPU

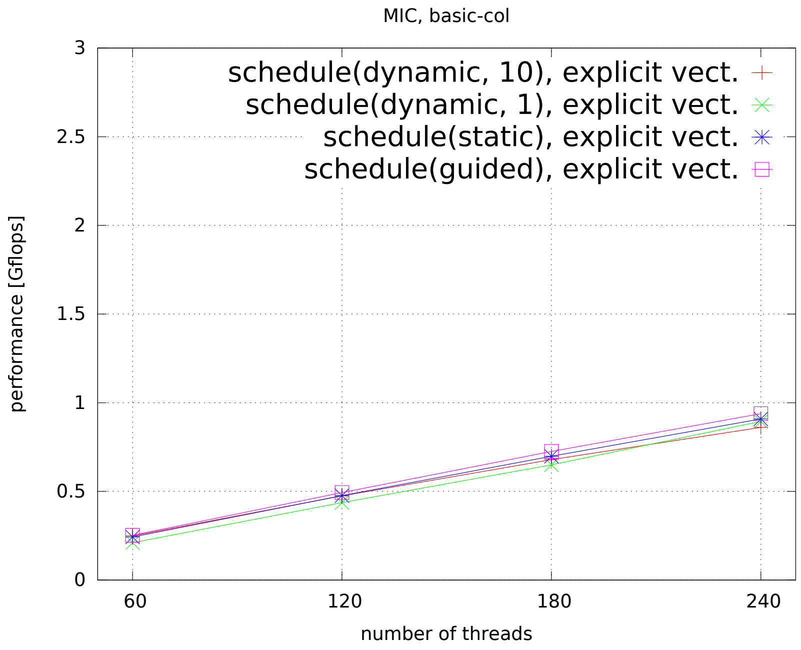


Figure 9: Manually vectorized `basic-col` on MIC

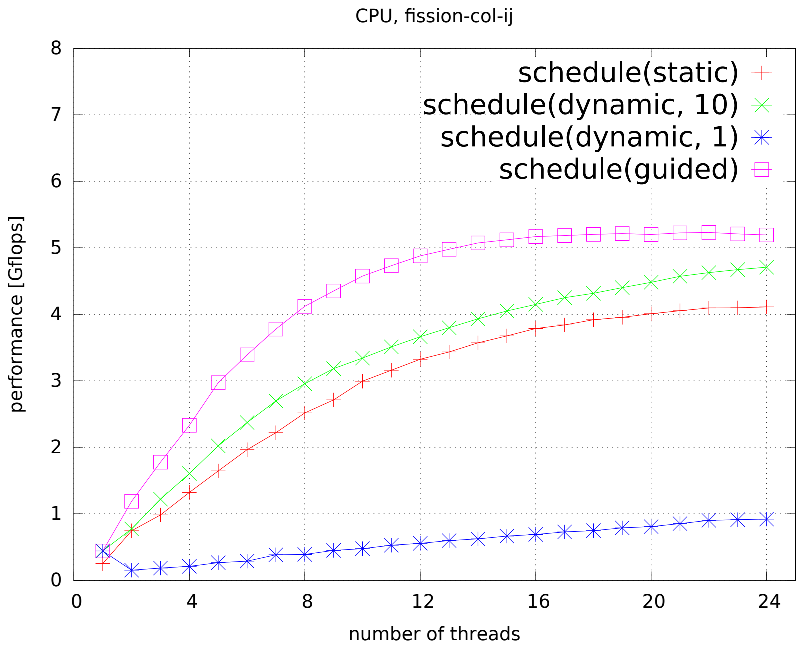


Figure 10: `fission-col-ij` on CPU

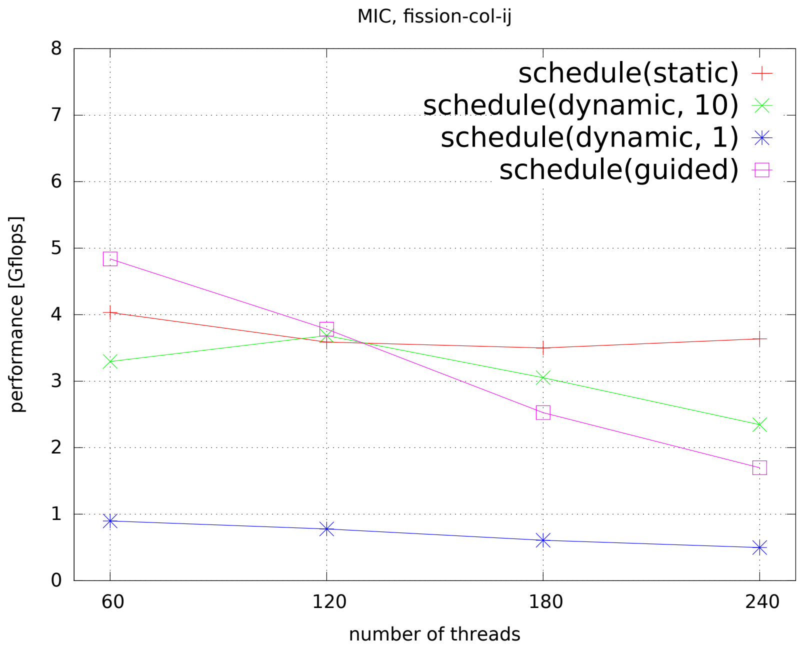


Figure 11: `fission-col-ij` on MIC

5.2 Matrix layouts and loop interchange

Next, we changed the matrix layouts (from column-wise to row-wise) and interchanged the order of the `i`-loop and the `j`-loop; however, the loop interchange is possible only in the `fission` implementation. Thus, we have the following varieties of our algorithm:

- `basic-col`
- `basic-row`
- `fission-col-ij`
- `fission-row-ij`
- `fission-col-ji`
- `fission-row-ji`

Two of them (`basic-col` and `fission-col-ij`) were tested in the previous subsection. Changing both the matrix layout and the order of the loops (from `fission-col-ij` to `fission-row-ji`) did not change the performance results, so they are not shown. The performance of the remaining versions on CPU are shown in Figures 12 (`basic-row`) and 13 (`fission-col-ji`/`fission-row-ij` — both versions with the layout compatible with the order of the loops gave the same results on CPU). The performance on MIC are shown in Figures 14 (`basic-row`), 15 (`basic-row` but without the explicit vectorization), 16 (`fission-col-ji`), 17 (`fission-row-ij`).

The best performance (independent of scheduling) is obtained by `basic-row` with the explicit vectorization. Obviously, the matrix layout and appropriate loops' order have a key impact on performance. If the layout is compatible with the order of loops (that is, `(i,j)` for row-wise, `(j,i)` for column-wise), the performance grows significantly. On CPU, the explicit vectorization in `basic` has no importance — it implies that the compiler can apply the vectorization automatically. However, on MIC, the explicit vectorization is essential, although, it only works for the proper matrix layout (that is, row-wise, because the loops must be in `(i,j)` order in `basic`). With no explicit vectorization on MIC, additional threads (that is more than one per core) enhance the performance, albeit they cannot catch up with the explicitly vectorized implementation. On the other hand, additional threads do not improve the performance; moreover, more than two threads per core even spoils it. The type of scheduling has a very little impact on the performance. On CPU, the performance grows rapidly (almost linearly) up to 6 threads, but after that it stabilizes. On CPU, the best results are achieved by all the implementations which have the matrix layout consistent with the loops' order — with no regard to the explicit vectorization. On the other hand, on MIC, the layout consistency is not the only factor — the `basic` algorithm is much better, and it must be explicitly vectorized (by `#pragma omp simd`).

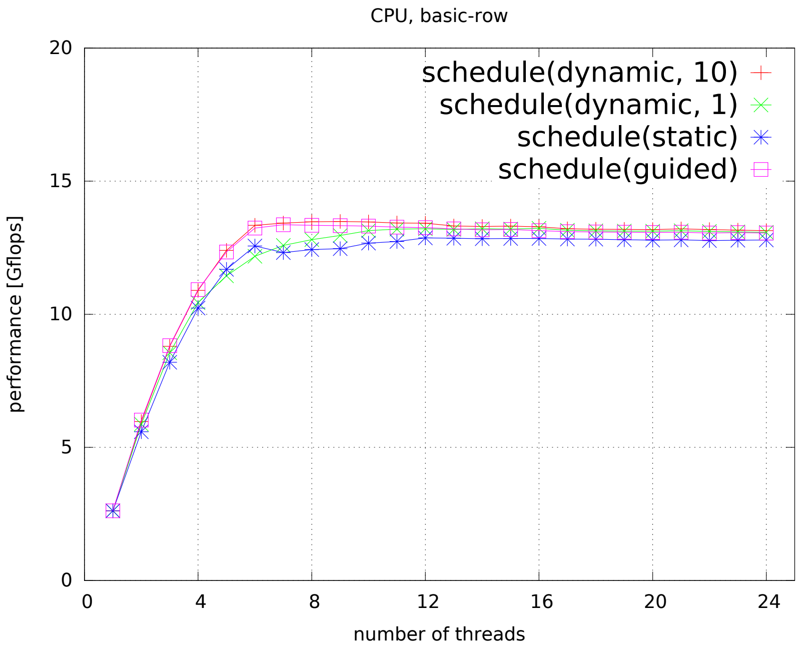


Figure 12: `basic-row` on CPU

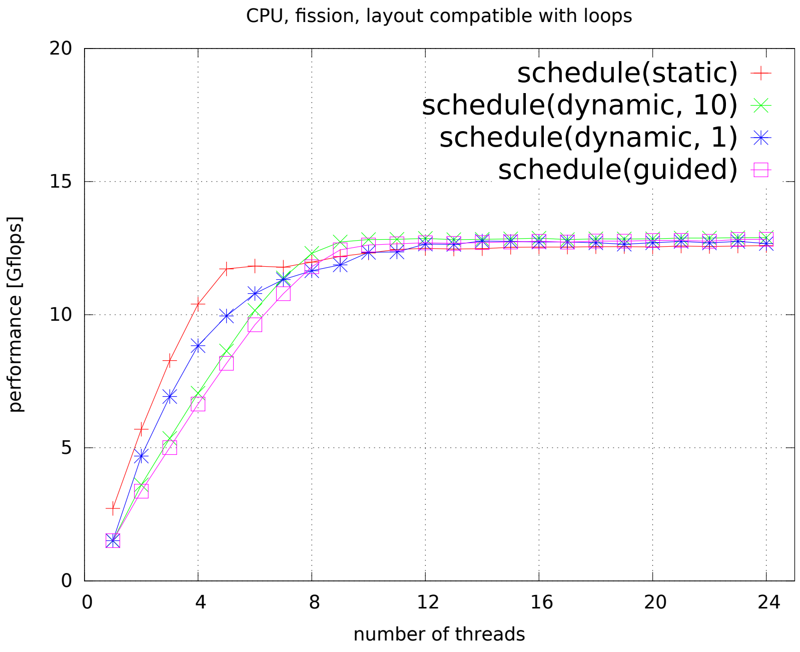


Figure 13: `fission-col-ji`/`fission-row-ij` on CPU

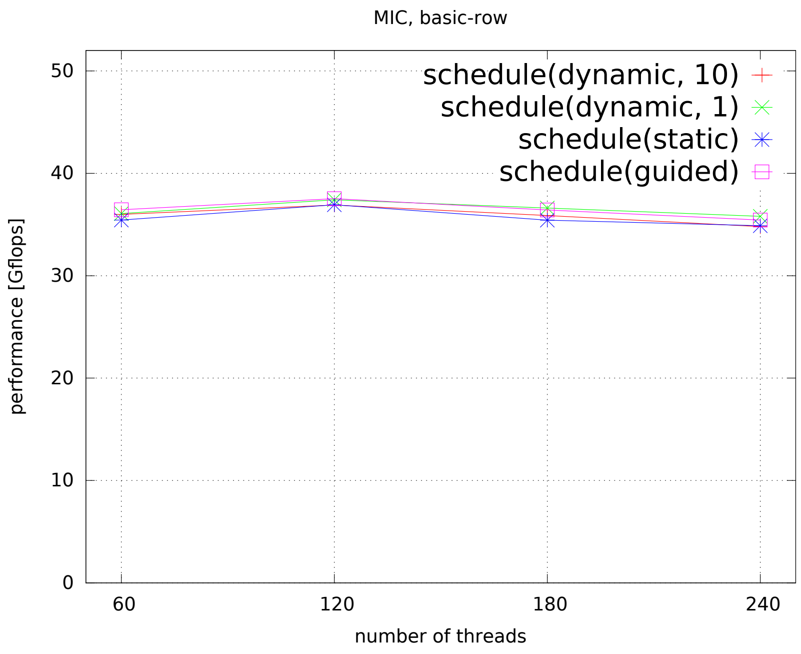


Figure 14: Manually vectorized `basic-row` on MIC

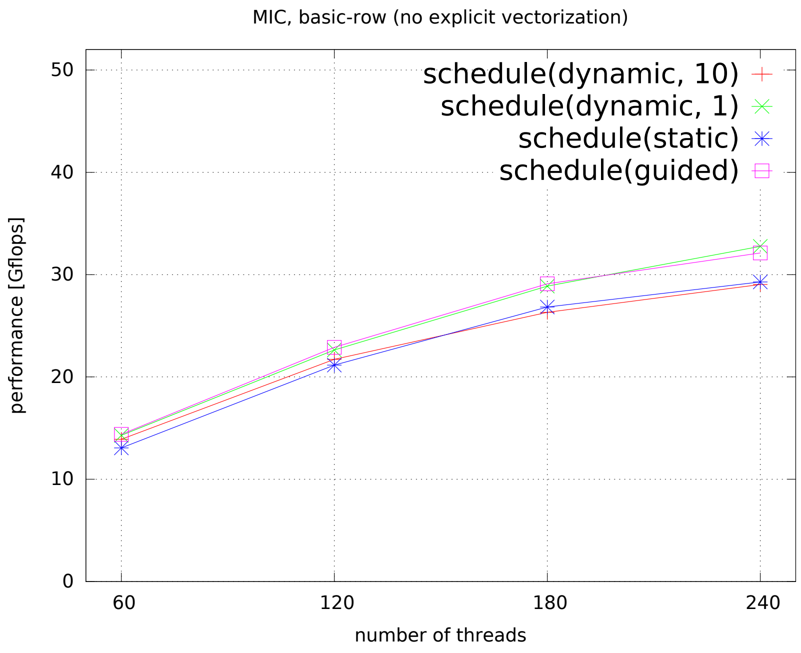


Figure 15: Non-vectorized `basic-row` on MIC

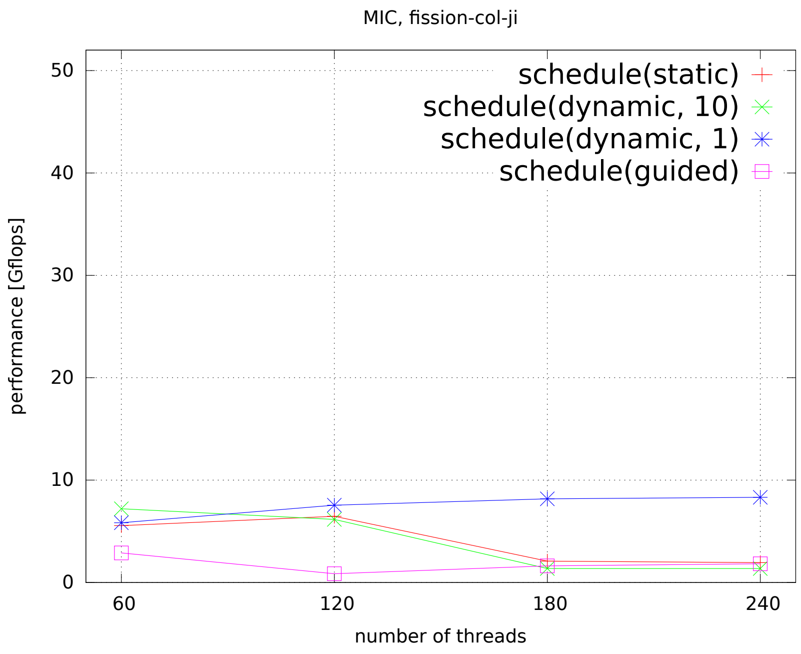


Figure 16: `fission-col-ji` on MIC

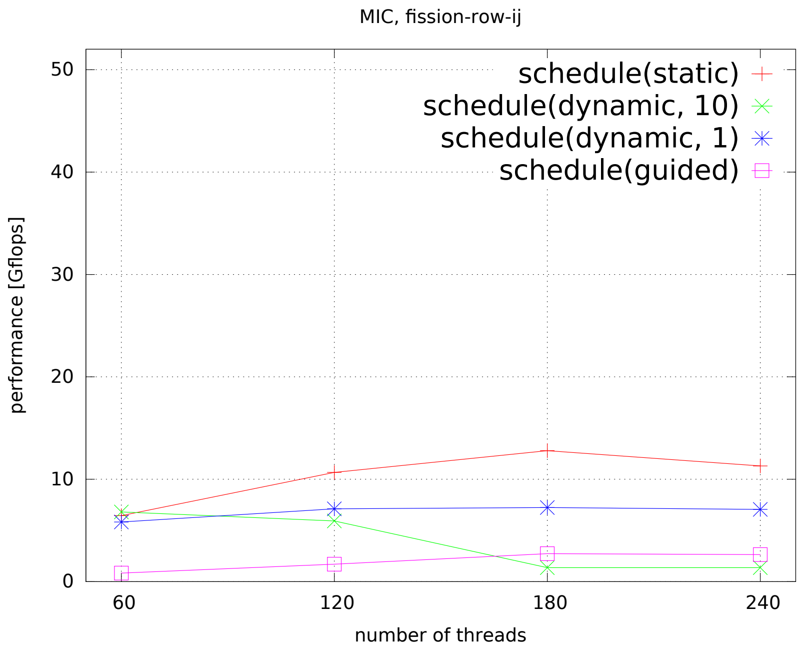


Figure 17: `fission-row-ij` on MIC

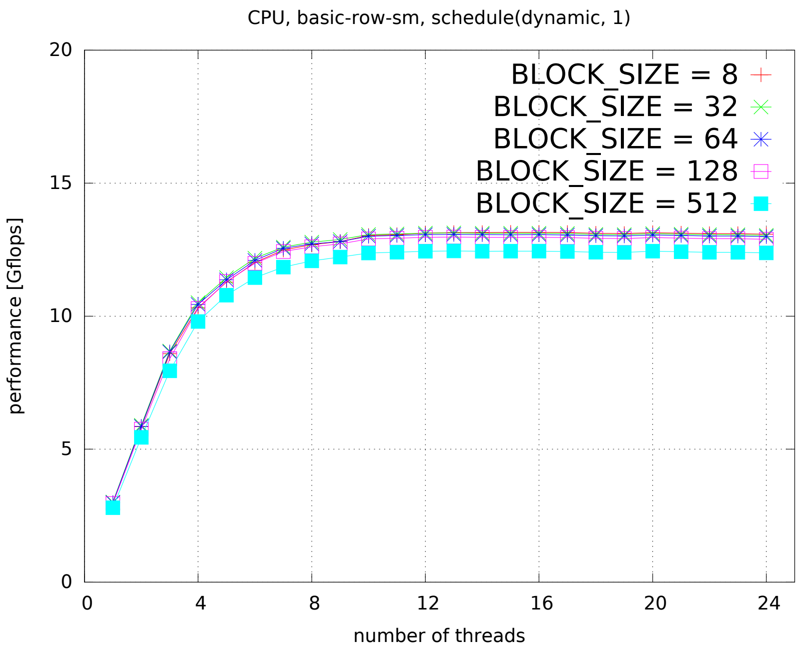


Figure 18: `basic-row-sm` on CPU with `schedule(dynamic,1)`

5.3 Strip-mining and loop tiling

Finally, we considered strip-mining (in the case of the basic version) and loop tiling (in the case of the fission implementation). We tested only implementations which gave reasonable results in the previous stage — that is, only those where the loop order was consistent with the matrix layout. Various sizes of the blocks were tested to experimentally check their influence on the efficiency. The sizes were 8, 32, 64, 128, 512. Thus, we tested the following versions:

- `basic-row-sm-`*b*
- `fission-row-ij-lt-`*b*
- `fission-col-ji-lt-`*b*

Here, `sm` stands for *strip-mining*, `lt` stands for *loop tiling* and *b* is the `BLOCK_SIZE`. The results on CPU are shown in Figures 18 (`basic-row-sm` for `schedule(dynamic,1)` — all schedulings gave the same results), 19 (`fission-row-ij-lt` for `schedule(dynamic,10)`), 20 (`fission-row-ij-lt` for `schedule(dynamic,1)`). The results on MIC are shown in Figures 21 (`basic-row-sm` for `schedule(dynamic,1)` — all schedulings gave the same results), 22 (`fission-row-ij-lt` for `schedule(dynamic,10)`), 23 (`fission-row-ij-lt` for `schedule(static)`).

The best performance (only about 13 Gflops) is obtained for the basic version with the `BLOCK_SIZE` equal to 8 or 32 — although the tiling is not full (here

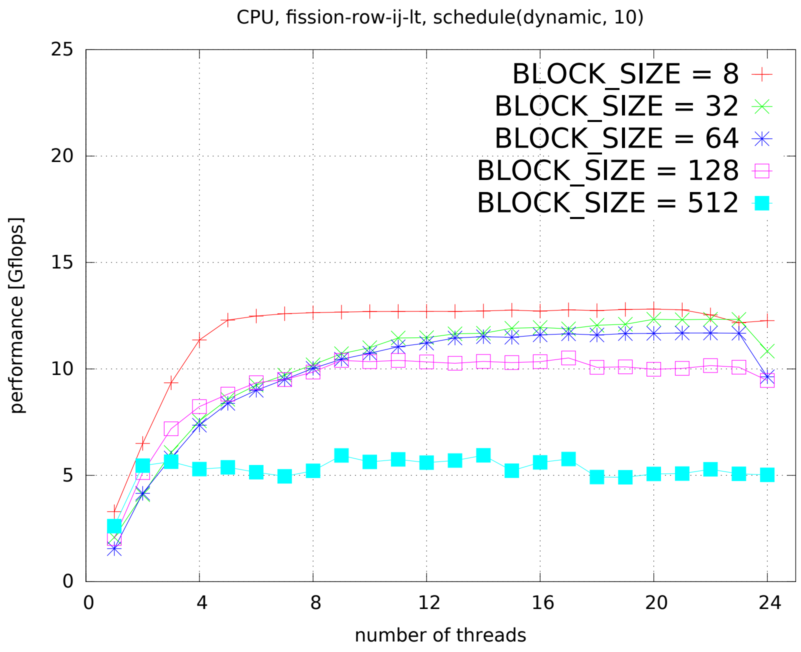


Figure 19: `fission-row-ij-lt` on CPU with `schedule(dynamic,10)`)

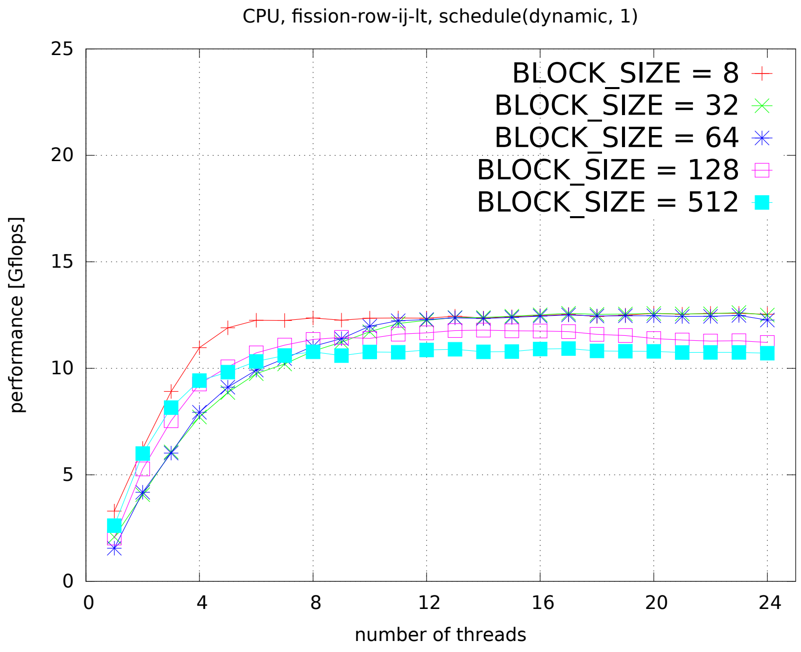


Figure 20: `fission-row-ij-lt` on CPU with `schedule(dynamic,1)`)

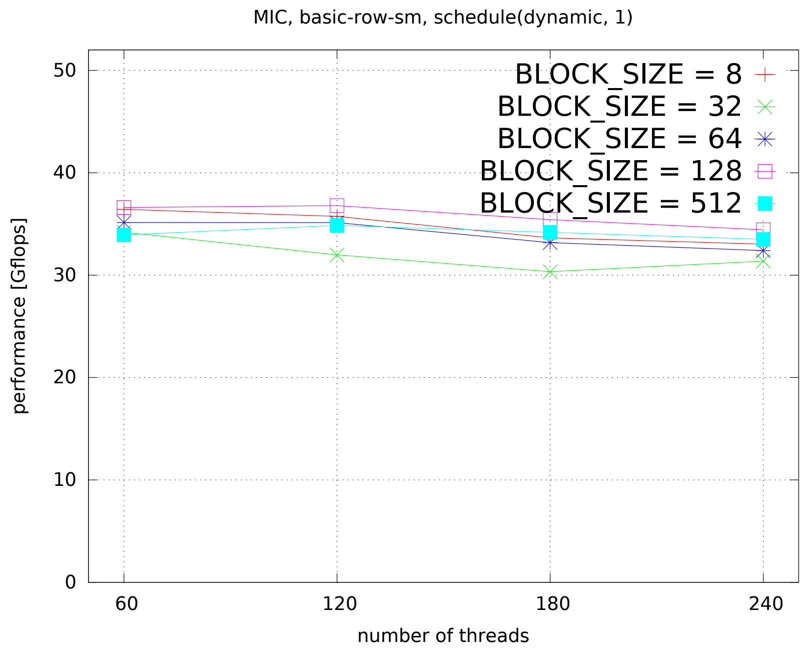


Figure 21: `basic-row-sm` on MIC with `schedule(dynamic,1)`

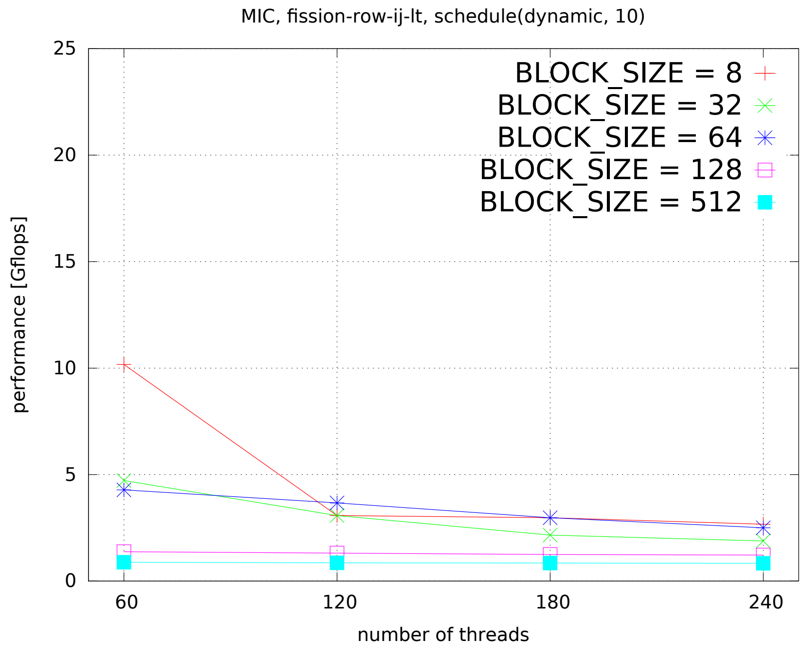


Figure 22: `fission-row-ij-lt` on MIC with `schedule(dynamic,10)`)

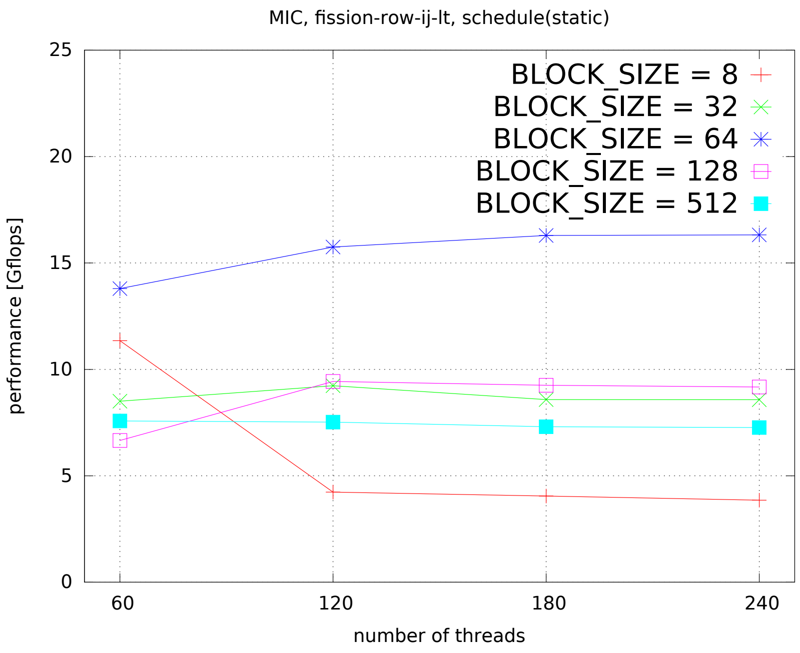


Figure 23: `fission-row-ij-lt` on MIC with `schedule(static)`)

we have only the strip-mining within the inner loop). The worst performance is obtained by the fission version with big blocks (the `BLOCK_SIZE` of 512). The best `BLOCK_SIZE` for the fission implementation is 8, and it does not depends on the matrix layout (unless it is incompatible with the loop order). However, comparing the strip-mining and loop tiling implementations to the best of previous ones, these modifications do not improve the results on CPU. Moreover, for some unlucky sets of parameters, they impair the performance.

On MIC, the best performance (after strip-mining and loop tiling) is obtained by the basic version with the `BLOCK_SIZE` equal to 128, and the worst is for `BLOCK_SIZE` equal to 32. The best `BLOCK_SIZE` for the fission implementation is also 128. However, the worst `BLOCK_SIZE` for the fission implementation is 8 (for row-wise version). The best number of threads is 2 per core — more threads decrease the performance (or do not change, at best). The basic implementation was not improved by the strip-mining, however the fission version with the loop tiling was improved — although not capped the best basic implemetations.

Both for CPU and MIC, the basic version is better. The optimal `BLOCK_SIZE` is different. The layout (for the fission version) is insignificant (provided it is consistent with loop order). In this particular case, the strip-mining and loop tiling gave no real advantage over the more traditional implementations. The dependencies between remote areas of the memory (that is, the areas not local enough to fit together in cache) in the WZ factorization are critical and they rendered the successful utilizing of these techniques impossible for a traditional matrix layouts.

6 Conclusion

Tha aim of the paper were the investigation of the influence of two aspects of the algorithm and its implementations on the performance. Those aspects were the dense matrix layout in the memory and some loop transformations (namely: loop interchange, strip-mining, loop tiling) — and the relations between them. Additionally, the influence of other factors (optimizing the running of the implementation) — namely explicit vectorization and scheduling — were studied. We considered two types of the modern multicore and manycore hierarchical shared memory architectures — like CPU and MIC. We did an analysis of the behavior of a dense linear algebra algorithm (namely, the WZ factorization) containing nested loops with strong dependencies between remote parts of the main memory.

Loop transformations with an appropriate matrix layout caused some performance growth only in specific cases.

On CPU, the correspondence between the matrix layout and the loop order has the largest influence on the algorithm optimization. The other factors (the explicit vectorization and scheduling) have a marginal significance.

The MIC architecture performance is much better in general. However, its performance depends strongly not only on the matrix layout and the loop order, but also on the explicit vectorization, scheduling and the size of the block (for loop tiling and strip-mining versions). The best performance on MIC is achieved by the `basic-row` version with explicit vectorization, guided scheduling and 120 threads (for the `basic` algorithm) and by the `fission-rot-ij-lt-64` version with explicit vectorization, static scheduling and 240 threads (for the `fission` algorithm). We show that both the `basic` version and the `fission` version of the algorithm can reach better performance on MIC than their analogs on CPU; although after some careful manipulations, especially in the case of the `fission` version.

The methodology (that is, the appropriate layout and loop order, strip-mining and loop tiling) applied here for a nested loop algorithm can be broadened to other numerical tasks containing nested loops and strong, remote and complex dependencies. It can also be utilized in other parallel languages and frameworks (like Cilk). The obtained results can be too transferred to the latest Intel (or other) CPU families. Higher performance can be expected on newer architectures that include more cores and memory and longer vector registers.

In future work we plan to extend the study to other architectures, especially GPU and the CPU-GPU hybrids — to see how the described methodology operates on those architectures. We also intend to use another dense matrix layout, namely a tiled storage scheme which can yet decrease the amount of cache misses.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide.* Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] Beata Bylina and Jarosław Bylina. Strategies of parallelizing nested loops on the multicore architectures on the example of the WZ factorization for the dense matrices. In M. Ganzha, L. Maciaszek, and M. Paprzycki, editors, *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, volume 5 of *Annals of Computer Science and Information Systems*, pages 629–639. IEEE, 2015.

[3] J. Bylina and B. Bylina. Parallelizing nested loops on the Intel Xeon Phi on the example of the dense WZ factorization. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 655–664, 2016.

[4] Simplice Donfack, Jack Dongarra, Mathieu Faverge, Mark Gates, Jakub Kurzak, Piotr Luszczek, and Ichitaro Yamazaki. A survey of recent developments in parallel implementations of Gaussian elimination. *Concurrency and Computation: Practice and Experience*, 27(5):1292–1309, 2015.

[5] Jack J Dongarra, Mathieu Faverge, Hatem Ltaief, and Piotr Luszczek. Achieving numerical accuracy and high performance using recursive tile LU factorization. *Concurrency and Computation: Practice and Experience*, 26(6):1408–1431, 2013.

[6] D.J. Evans and M. Hatzopoulos. A parallel linear system solver. *International Journal of Computer Mathematics*, 7(3):227–238, 1979.

[7] S. Chandra Sekhara Rao. Existence and uniqueness of WZ factorization. *Parallel Computing*, 23(8):1129–1139, 1997.

[8] P. Yalamov and D.J. Evans. The WZ matrix factorisation method. *Parallel Computing*, 21(7):1111–1120, 1995.

[9] Basic Linear Algebra Subprograms, 2020.

[10] Intel Math Kernel Library, 2020.

Functioning of Transnational Civil Society Organisations (TCSOs) in Cyberspace

Ewelina Panas*

1 Introduction

The aim of this article is to analyse the functioning of TCSOS in cyberspace: the determinants of the activity of these organisations in the virtual world will be analysed, the peculiarities of TCSOS that enable them to undertake activity in cyberspace will be analysed, and specific examples of TCSOs' involvement will be given. The main thesis of the considerations in the article is the statement that the peculiarities of TCSOs with regard to: firstly, the way they are organised (networked, non-hierarchical) and secondly, the issues they deal with (global, crossing border) determines the virtualisation of the activity of these entities, transferring it to cyberspace. The starting point for the analysis of the issue of TCSOs activity in cyberspace will therefore be the examination of the relationship between the properties of cyberspace and the specific nature of TCSOS (the characteristic qualities of these organisations that affect the performance of their tasks in this particular space).

One of the best known and widely quoted definitions of cyberspace is that formulated by the US Department of Defense, according to which cyberspace is: "The global domain of the information environment consisting of interdependent networks formed by information technology (IT) infrastructure and the data they contain, including the Internet, telecommunications networks, computer systems, and embedded processors and controllers" [1]. Pointing out the basic elements of cyberspace allows us to fully understand the essence of this peculiar environment: vastness (global reach), bundling of all resources into one huge database, complexity, and spatiality understood as the impossibility of relating cyberspace to the physical (including geographical) dimensions of the real world [1].

Modern information and communication technologies are now widely available and TCSOs with their solutions. The digitalisation of the life of societies also includes the functioning of civil society organisations, especially those oriented towards transnational activity, going beyond the borders of a single state. TCSOS are understood as organisations which, while functioning in a transnational space,

*Corresponding author — ewelina.panas@mail.umcs.pl

build the subjectivity of individuals and social groups, have a non-governmental character, and are not profit-oriented in their activity. TCSOS are structures representing civil society and the values and principles inherent in that community, bringing together in their ranks (both members and supporters) people from different countries, and their activity has a transnational dimension, which means that it takes place "across and beyond national borders, the control of which the states have consciously given up or limited" [2].

Undoubtedly, the flourishing of TCSOS activity in the international arena would not have been possible without the scientific and technological revolution. The transformations that took place in the international environment under the influence of this factor conditioned the development and character of the activities of civil society organisations, equipping them with the instruments necessary to undertake transnational activities. New technological developments, such as the development of commercial media and the Internet, have provided organisations with innovative tools, increasing their capacity to influence public opinion and centres of power and, facilitating the promotion of specific issues. Technological changes have also helped coordinate the geographically dispersed activities of TCSOS and made it possible to carry out tasks such as: global networking and international coalition building, exchange of experiences and ideas, mutual communication, association and the collection, processing and transfer of information [3].

Today, by using methods and means that were not available 20 years ago, transnational civil society organisations have the ability to disseminate information almost instantaneously, take action to gain mass support, initiate public campaigns on a global scale and encourage people around the world to get involved in their projects [4].

2 Determinants of the virtualisation of TCSOs' activities

The transition of TCSOs in their activity to the cyber space is mostly related to two factors. Firstly, the way these organisations operate, and secondly, the specific nature of the domains in which they are present and active.

The first factor relates to the way in which these entities are organised, managed and operate, in terms of the organisational forms/structure adopted and the methods of operation of TCSOS. One of the most common forms of TCSOS activity is the establishment and functioning within transnational coalitions. An important feature of TCSOS coalitions is their internal 'architecture' — a network form of organisation. TCSOS coalitions are a type of social network, forming "a dynamic system of communication, cooperation and partnership between groups or institutions" [5]. It is a cross-border network of connections between individuals and groups working together towards common goals. Such cooperation may be more or less formalised and the actors may be individuals, NGOs, pressure groups or interest groups. The purpose of networks is cooperation, exchange of information, experience and taking joint initiatives. An important feature of networks created by TCSOs is their transnational character. We can speak of transnational networks when: firstly, at least one of the actors forming them is of non-state character, and secondly, they come from at least two countries [5].

Three basic criteria are used to distinguish networked coalitions as new, unconventional methods of regulating the international environment: first, this type of interaction is applied to the solution of a specific, single problem. Secondly, they are characterised by mixed participation. This is because they are formed by heterogeneous groups of participants in international relations: states, international organisations, non-governmental organisations. Thirdly, such coalitions are not based on any binding international document. They do not result from any existing international agreements or treaties [6].

Thus, transnational networked coalitions of transnational civil society organisations can be regarded as flexible, non-hierarchical, self-organised problem networks, united by the pursuit of common goals and objectives, which transcend national borders. The basic bond of TCSOS network coalitions is a specific value system, acting for a common cause and initiating and engaging in an international debate on a specific phenomenon or problem.

A key element in the functioning of network coalitions of transnational civil society organisations is the use of virtual space. It seems that this space — removing all barriers and restrictions and enabling the global flow of ideas and almost immediate exchange of information — is the optimal environment for networked, non-hierarchical, cross-border structures such as TCSOS coalitions. By operating in cyberspace, TCSOS reach a wide audience with their mission and message. The virtual activity of coalitions of transnational civil society organisations results from their ability to adapt to the specific parameters of this space and use its properties to pursue their own interests. In fact, one might even venture to say that TCSOS coalitions — and transnational civil society organisations in general — are endowed with a specific kind of cyber-power, meaning “the ability to achieve desired results through the use of information resources, electronically connected in virtual space (cyberspace)”, which is based on resources related to “the creation, control and transmission of electronic information”. The spectrum of these resources includes: infrastructure, networks, software, human skills and knowledge [7].

The second factor that favours the transition of TCSOs in their activity to the cyber space is the specificity of the fields of interest and activity of these organisations. TCSOs deal with a variety of problems and are present and active in many spheres of international reality. The organisations are most interested in issues related to environmental protection, humanitarian aid, protection of human rights, problems of poverty, poverty, social and ecological responsibility of transnational corporations, marginalisation of many social groups or international justice in the sense of distribution of world wealth. The non-territorial quality of these problems — their specificity lies in the fact that they take place at the local, regional and global level. The essence of the action taken by civil society organisations, mainly through the use of solutions provided by cyberspace, lies in combining and co-shaping the local with the global. This is a decisive element in the effectiveness of these actors — TCSOS become part of the regulation of problems of a cross-border nature and those operating in a transnational social space [8].

Cyber space is therefore a source of solutions and instruments with which TC-

SOS are able to pursue their mission — to solve problems of a global, cross-border nature.

3 Functions performed by TCSOs in cyberspace

The functioning of TCSOS in cyberspace is related to the performance of specific functions by these actors. Craig Warkentin points to a number of tasks that transnational CSOs carry out through the Internet: improving internal organisation and facilitating communication with partner organisations, shaping public perceptions, disseminating information about ongoing projects, encouraging involvement in ongoing activities [9]. Moreover, the websites created by TCSOS foster the dissemination of information about the organisations themselves and their work, facilitate the recruitment of new members and communication with existing ones, and provide the possibility to post information asking for financial support [9].

Cyber space is used by TCSOs primarily for: communicating with members and supporters, multiplying messages (implementing agenda setting) and building and reinforcing a particular narrative, organising online (virtual) demonstrations, hacking attacks.

Communication with members and supporters. The specific nature of membership means that in order to maintain constant contact with members and supporters, organisations need to use tools that enable them to do this for people from different countries. Maintaining constant contact, providing information, engaging members in lobbying activities is possible by using solutions available in cyberspace.

Multiplication of messages (implementation of agenda setting), specific narrative building, Virtual space — by creating new opportunities for communication, opening new channels for the articulation of alternative visions, proposals and solutions — makes it possible not only to maintain contact with members and supporters, but also to multiply the message of TCSOS, which is another important utility of cyberspace.

The technological infrastructure that has emerged as a result of scientific and technological progress has become a 'transmission belt' for the demands, innovative solutions, ideas and projects created by TCSOS. Civil society organisations are a valuable source of information and ideas, and modern technologies favour the distribution of these “products” through traditional channels, such as the press, radio or television, but also by using new forms of communication, such as social networking sites, blogs or podcasts. Even small organisations with small budgets, thanks to the low cost and high technological quality of digital recordings, can gather and disseminate information irrespective of their location [3].

Organising online (virtual) demonstrations. Internet users can come together much more effectively, cheaply and quickly, and organising actions can be done by phone and email.

They can also work with other organisations in transnational, networked coalitions. The changes associated with the scientific and technological revolution are also conducive to the networking process: the number, density and size of networks are increasing, and they are becoming more professionalised [9]. Structures that use technological innovation to strengthen cooperation between TCSOS include the Association for Progressive Communication (APC), a network of civil society organisations that aims to develop and support organisations, social movements and individuals through the use of new information and communication technologies[1].

4 Case study

The concept of a currency transaction tax (the so-called "Tobin tax" — CTT), first appeared in 1972, when the American economist James Tobin, postulating a reform of financial markets, proposed to impose a fee (of between 0.1% and 0.5%) on foreign exchange transactions, arguing that this would limit the scale of purely speculative transactions[2] [10]. The proposed reform, which James Tobin has graphically described as "throwing sand on the wheels of the global finance machine", was intended — by raising transaction costs — to stabilise financial markets and increase the independence of states, especially with regard to the autonomy of monetary policy [11]. James Tobin referred in his conception to the views of John M. Keynes. As he pointed out, "(...) Keynes had already seen in his time that the advantages of liquidity and transactions of financial instruments meant a surge in speculation that was short-sighted and inefficient (...) he was right to suggest creating greater impediments to short-term transactions and rewarding long-term investors" [12].

James Tobin's idea has gained a wide range of supporters, but also opponents. The arguments of opponents of the CTT concept are based on the assumption that a currency transaction tax could only be effective if all, or at least a large majority of countries, introduced it, which would prevent currency transactions from migrating abroad. "Tobin tax" has also been criticised on the grounds that by increasing transaction costs, it would have a detrimental effect on the efficiency of the market mechanism [9].

The interest of civil society organisations in the "Tobin Tax" emerged in the early 1990s, as the "philanthropic" dimension of the concept was recognised, meaning that it began to appear in the context of a debate on attempts to find alternative sources of financing for the United Nations and, more generally, as an instrument for reducing global poverty. The issue of the 'Tobin tax' as a tool for combating world poverty was discussed, among others, at the World Summit for Social Development in Copenhagen in 1995 [13]. It has been argued that raising transaction costs would generate funds that could potentially go towards economic stabilisation and meeting the social needs of countries in the South and supporting UN activities.

Thus, for civil society organisations, the concept of a currency transaction tax, which was intended by its creator as a tool to stabilise financial markets, was part of a broader debate on global social justice, counteracting the unregulated and unfair distribution of wealth and its concentration in the hands of the richest. The organisations considered the introduction of CTT in terms of counteracting the negative effects of globalisation processes, such as progressing material and public

[1]See: Website of the Association for Progressive Communications, http://www.apc.org/

[2]See: D. Rosati, Co z podatkiem finansowym?, „Gazeta Wyborcza" 30.01.2012, http://wyborcza.biz/biznes/1,101562,11051865,Co_z_podatkiem_od_transakcji_finansowych_.html

polarisation and stratification of societies, which results in a growing number of the excluded, i.e. people and groups who do not participate in social life in any way due to the lack of access to resources [12].

According to estimates by organisations promoting the 'Tobin Tax', the introduction of a CTT of 0.1% on the value of international currency transactions would provide funds of around USD 150 billion per year. The revenue from the tax could be used, as advocated by the organisations, to achieve such goals as the fight against hunger, providing access to clean water, health services and universal education in the countries of the South. And, at 0.003%, the CTT would provide funding to cover the costs of all peacekeeping operations conducted under UN auspices. In this sense, the CTT would become an important instrument in the fight against inequality and social exclusion in the world[3].

The populist appeal of the slogans resulted in a rapid increase in the number of supporters of the use of foreign exchange tax as a tool to fight social injustice. Initially, the popularity of the concept was volatile, but the Asian financial crisis that began in June 1997 catalysed the development of various grassroots initiatives to find new tax solutions as a response to the financial crisis. Soon, the promotion of the "Tobin Tax" assumed the dimension of a transnational campaign for the introduction of a CTT, inspired and coordinated by ATTAC (Association pour la Taxation des Transactions Financière et l'Aide aux citoyens — Association for the Taxation of Financial Transactions and Assistance to Citizens) [14]. The association associated with the Le Monde Diplomatique newspaper was created in December 1998 in France and is now a network of organisations operating in more than 40 countries[4]. In Poland the organisation operates under the name Citizens' Initiative for Taxation of Capital Trading — ATTAC[5].

At the same time, independent of the initiative that led to the creation of ATTAC, a number of other civil society organisations have also made the imposition of a tax on currency transactions a leitmotif of their campaigns. The most important organisations lobbying for the introduction of a "Tobin tax" are: International Co-operation for Development and Solidarity, an alliance of 15 Catholic development organisations from Europe and North America, the Halifax Initiative — a coalition of faith groups, environmental, human rights, development and social justice organisations, World Women's March, Third World Network, Global South, Network Institute for Global Democratization (NIGD) — an international network of nearly 100 academics, Tobin Tax Initiative (Tobin Tax Initiative) and War or Want, an organisation fighting poverty in developing countries [10].

The activity of these organisations became an expression of contestation of the global economic order, symbolised by international financial institutions. Therefore, the activists' demands were directed against the financial sector — it was considered that it bears a significant part of the responsibility for the financial crisis, and therefore should contribute more to the costs of removing its consequences [15]. ATTAC, together with other organisations involved in the campaign for a 'Tobin tax', took part in many demonstrations against the G8, the G20 and also in the

[3]See: Website of the ATTAC, http://www.attac.org/en/whatisattac/international-platform

[4]See: Website of the ATTAC, http://www.attac.org/en/overview

[5]See: Website of the Obywatelska Inicjatywa Opodatkowania Obrotu Kapitałowego, http://www.attac.pl/

demonstrations that led to the blocking of negotiations at the 1999 WTO conference in Seattle. And in 2000, under the banner of 'anti-Davos', a parallel conference to the World Economic Forum (WEF) summit was organised. The "anti-Davos" conference started with a seminar in Zurich, followed by an activist march to Davos. ATTAC is also one of the organisations that organised the first World Social Forum in Porto Alegre in 2001, which was attended by around 10,000 people[6] [10].

This alternative initiative to the World Economic Forum aimed to create a social platform for discussion and exchange of ideas. The activists' actions were an expression of dissatisfaction with the current global socio-political order and the fact that in the forums of international organisations — in the opinion of the activists undemocratic and not representing the entire world society — decisions affecting everyone are made. These activities were also aimed at convincing public opinion that a more democratic system could be created by introducing appropriate mechanisms. One of the proposals for economic stabilisation and balancing the position of rich and poor societies was the introduction of a tax on currency transactions.

The popularity of the 'Tobin tax' concept was variable. The impetus for increased efforts, not least by civil society organisations, to adopt a mechanism to regulate the financial markets was provided by the recent financial crisis that began in 2008. The discussion on the introduction of a tax on foreign exchange trading flared up again. As a consequence of the variety of actors involved in the campaign for the adoption of a 'Tobin Tax', there are different visions regarding the introduction of a CTT. Two options are being considered with regard to how revenues generated from foreign exchange transactions would finance development needs. War on Want advocates a minimalist version of the CTT of 0.05%, with the tax levied not only on currency transactions but on banks' financial transactions in general (the so-called 'Robin Hood tax'). The revenue from the tax would make it possible to "raise USD 20 billion each year to support the Millennium Development Goals fund"[7].

In contrast, ATTAC is lobbying for the adoption of the CTT in the form of an international agreement. The organisation has developed a 'Draft Global CTT Treaty', which is also supported by other organisations associated with the World Social Summit. In addition, the first model involves the CTT being implemented unilaterally by individual states or the European Union. As proponents of this option argue, the EU is a large enough economic organism to be able to implement a financial transaction tax unilaterally on its own and safely. The ATTAC proposal, on the other hand, is for global regulatory change. In response to War Or Want, ATTAC economists propose a two-tier capital transaction tax, with two CTT rates: a lower rate for transactions within the tax zone and a higher rate for transactions between 'Tobin Tax' countries and the rest. The idea is that this would encourage other countries with links to European economies to join the CTT zone [10].

The transnational mobilisation of civil society organisations promoting the 'Tobin Tax' as a tool to address North-South disparities has helped to bring the issue to the attention of policy makers and to the attention of parliaments in many coun-

[6] See: Website of the ATTAC, `http://www.attac.org/en/overview`

[7] See: Website of the War or Want, `http://www.waronwant.org/campaigns/tax-justice-now/the-robin-hood-tax`

tries, the European Parliament and many international organisations, including the UN. Many countries supported the TCSOS initiative and the introduction of a 'Tobin Tax' — in 1999 in Canada, in 2001 in France and three years later in Belgium, proposals were adopted supporting the implementation of the CTT. In 2004, a minimalist version of the CTT [12, 10] was supported by the then presidents: France — Jacques Chirac and Brazil — Luiz Inácio Lula da Silva. The presidents of Brazil and France were soon joined by the presidents of Chile and Spain and the UN Secretary-General. The so-called Lula-Chirac Initiative was followed by the report "Action against Hunger and Poverty", which proposed the establishment of alternative sources of financing for "public welfare and development" through the creation of financial mechanisms at the international level that would contribute to reducing global poverty [16]

The idea of introducing a tax on financial transactions has recently emerged as part of the "programme" and one of the demands of the so-called "Outraged" movement, which emerged on the wave of social discontent caused by the 2008 crisis, but the escalation of the activity of the "Outraged" took place in 2011. The "Tobin Tax" has been integrated into a series of demands concerning the need to rebuild the badly organised global socio-political order [15]. The source of the global protest, which took the form of an international movement, was the Spanish Revolution of the Indignados (revolution of the "indignados"), a mass protest that began on 15 May 2011 in the Puerta del Sol square in Madrid (and almost 50 other Spanish cities). The Spanish indignados inspired multitudes of young people around the world. On 17 September 2011, with a series of demonstrations on Wall Street in New York, the Occupy Wall Street movement launched its campaign as an expression of dissatisfaction with the lack of social solidarity of the political and economic elites in the fight against the crisis[8]. Another initiative by activists demanding change, which united thousands of people around the world, was the 15 October Agreement. The creation of this movement was also inspired by the activity of the Spanish indignados. The activity of the movement is a demonstration of opposition to the current economic and political status quo. On 15 October 2011, a global demonstration began — protests took place simultaneously in more than 951 cities in 82 countries — under the slogan 'united for global change'. (United for Global Change), which was to express solidarity and identification with the goals and demands of Occupy Wall Street [15].

"Outraged" questions the international political and economic order, while the axis of criticism is based on the inept ways of fighting the global crisis, which, according to the movement's participants, comes at the expense of society[9]. As a result of the crisis of trust in the ability of international bodies to regulate the unstable global order, there are demands for the introduction of new instruments that would make it possible to reduce the disproportions between the poor South and the rich North, but also to counteract the social polarisation within individual countries. One of the instruments promoted by the Outraged to fight social and economic inequalities — besides abolishing the dictatorship of transnational

[8]See: V. Dobnik, Wall Street protesters: We‘re for the long houl, "Bloomberg Businessweek" 2.10.2011; http://www.businessweek.com/ap/financialnews/D9Q4CNR81.htm

[9]See.: T. Bielecki, Bunt Oburzonych, „Gazeta Wyborcza" 23.05.2011, http://wyborcza.pl/1,76842,9646298,Bunt_oburzonych.html

corporations, abolishing the so-called tax havens and cancelling the debts of the countries of the South — is the introduction of a tax on international trade in currencies [15].

The analysed campaign, apart from being an interesting voice in the discussion about the need for changes in the global economy and the ways of implementing possible transformations, is also an interesting material for reflection in the context of analysing the use of soft power by transnational civil society organisations. On the basis of the above considerations, the campaign for the "Tobin Tax" can be considered a model example of the ability to use the attractiveness of a target. Slogans for the enforcement of responsibility on those participants in international relations, whose actions led to the global crisis and, ultimately, to the deterioration of the living conditions of the inhabitants of poor countries of the South, met with the approval of those societies that feel victimised, in their opinion, by the unfair distribution of global wealth.

Although the Campaign for a Tax on Foreign Currency Transactions has so far failed to achieve its main objective, the mere introduction of the issue of a 'Tobin Tax' into the international discussion in various fora can be considered a major achievement. The organisations have succeeded in popularising the CTT as a tool for balancing the situation of the inhabitants of the North and the South. Undoubtedly, the campaign promoting the Tobin Tax is a demonstration of the ability of civil society organisations to shape the international public debate [14, 16] Activists have succeeded in directing the attention of global public opinion to the problem of the concentration of wealth, the deepening stratification of societies and the development and economic disparities between countries.

The fundamental weakness of the 'Tobin Tax' campaign appears to be the criticisms made by some economists of the very concept of a tax on international currency transactions. As already mentioned, the crowning argument against the CTT is the possibility of avoiding the levy by moving transactions to a country where the tax does not apply. There is also a not insignificant organisational and conceptual split between the organisations involved in promoting this instrument, resulting in two models of the tax — the minimalist CTT project, promoted by War Or Want, and the 'Draft Global CTT Treaty', developed by ATTAC. The lack of a common position and a precisely defined objective certainly weakens the campaign message and efforts to tax financial transactions.

Cyberspace has also played a huge role in another campaign organised by TC-SOs — the Campaign against the Multilateral Agreement on Investment (MAI). Of key importance throughout the campaign was the virtual collaboration between organisations. TOSO's use of the Internet was conducive to the core tasks of the campaign — disseminating information about the MAI (as well as activists' interpretations of the meaning of the agreement) and reaching out to broad audiences through Internet links. Thanks to this form of cooperation, many new organisations were mobilised and communication between organisations already involved in the campaign was improved. The use of the Internet also made it possible to directly influence political decision-makers — the websites of the organisations involved in the campaign provided e-mail addresses to members of national parliaments. Moreover, the use of technological infrastructure enabled social activism in a form that Ronald J. Deibert calls armchair activism, meaning that individuals could engage in the campaign against the MAI via the Internet and contribute to its success — joining the debate on the agreement under preparation and exerting pressure on individual governments, even without leaving their own homes[10] [19].

Among the ways in which TCSOs use cyberspace are cyber activism and hacktivism. Cyber activism is the use of the Internet to organise protests, demonstrations and lobbying, or to draw attention to and introduce a particular topic into international public debate. Online demonstrations can be cited as an example of cyber activism. One of the most spectacular virtual demonstrations was an action organised on the social networking site Facebook under the slogan "We are all Khaled Said". The page was set up on Facebook to commemorate a 28-year-old Egyptian man who was beaten to death by Egyptian police (original name Kullena Khaled Said). The page became a channel for the articulation of public discontent caused by these TCSOs covering of torture and police brutality [20].

Hacktivism, on the other hand, is the action of using IT knowledge with the intention of achieving a specific political or social goal. One type of hacktivism is hacking attacks on the websites of individual institutions whose actions are contested by activists. The events taking place in 2011-2012 with the participation of activists from the Anonymous group, which opposed the adoption of a multilateral agreement aimed at establishing international standards in the fight against infringements of intellectual property (ACTA — Anti-Counterfeiting Trade Agreement), are a manifestation of the aspirations of civil society to participate in setting the rules of control of the space that is the Internet [15]. The actions of Anonymous also demonstrate that decision-making in regulating this virtual sphere of international life requires multilateral consultation, including with the public.

5 Summary

Zygmunt Bauman pointed to the specificity (specific nature) of the functioning of transnational and virtual space and the incompatibility of state structures with the transformations taking place within it: "power, power — Macht, as Max Weber said — is already gliding in extraterritorial space, while all democratic institutions, institutions of political control over the use of power are still local. This means that this real power — Macht — is out of reach"[11]. Therefore, it can be concluded that virtual space is a potential source of power, but only for those entities that can functionally and structurally assimilate to it — "tune in" and adapt to the specific conditions present in it. Its effective use as an "infrastructure" favourable for the realisation of specific goals depends on the degree of adaptability and flexibility of individual entities. It seems that the potential of virtual space will be most effectively used by entities whose properties, method of organisation and logic of functioning correspond to the specificity of this environment. TCSOS, as flexible entities with a network structure of organisation and great ability to adapt to the changing conditions of the international environment, seem to be "programmed" to function in this space.

[10]See: M. Drohan, How the Net killed the MAI: grassroots groups used their own globalization to derail deal, "The Globe and Mail" 29.04.1998, http://www.chebucto.ns.ca/~aj382/how_net.html

[11]See: Tak zwana globalizacja, wywiad Witolda Gadomskiego z Zygmuntem Baumanem, „Gazeta Wyborcza", 09.11.2001, http://wyborcza.pl/1,76842,534465.html

It is worth referring here to the observations of Sidney Tarrow, who, while researching social movements in terms of their impact and strength, came to the conclusion that collective trust, which is the basis for the functioning of such cooperative structures, cannot develop and strengthen without the collective, direct experience of people involved in such cooperation. The source of such experiences — and, at the same time, the factor which is of paramount importance in strengthening mutual trust and developing cooperation within social networks and movements — can only be direct encounters in the real sphere. Virtual activism is devoid of such implications [17]. Sidney Tarrow's thoughts gain particular significance especially in the context of emerging arguments about the shortcomings of global civil society, the activity of which is increasingly shifting to the virtual sphere. As Harris Breslow argues, civil society in the global dimension will always be limited in some way, mainly due to the fact that functioning in the virtual sphere makes it impossible to create structures of a truly civic and communal character [18, 19].

However, it seems that these opinions are not fully justified. As a rule, the activities and strategies of global civil society organisations combine elements related to both virtual and real space. The Internet is a tool that facilitates the functioning of these entities, fosters the exchange of information, organisation of actions, mutual communication between individual organisations and between the organisation and its members. However, virtual activism does not replace real activism — it is a significant supplement to it. Examples of organisations functioning in this way include the Pirate Party and Anonymous, whose activity is based on a skilful combination of virtual and real activity. It seems that at present such a way of functioning — combining "traditional" methods with new ones based on technological innovations — is characteristic of most, if not all, global civil society organisations.

References

[1] Wasilewski, Janusz. "Zarys definicyjny cyberprzestrzeni." Przegląd Bezpieczeństwa Wewnętrznego 5.9 (2013)

[2] Dumała, Andrzej. "Uczestnicy transnarodowi–podmioty niezależne czy kontrolowane przez państwa?." Państwo we współczesnych stosunkach międzynarodowych (1995)

[3] Yaziji M., Doh J., Organizacje pozarządowe a korporacje, Warszawa 2011.

[4] Mingst K., Podstawy stosunków międzynarodowych, Warszawa 2006.

[5] Dumała, H. "Transnarodowe sieci w stosunkach międzynarodowych, w: Pietraś M." Międzynarodowe stosunki polityczne. Lublin (2006).

[6] Haliżak, Edward, and Roman Kuźniar, eds. Stosunki międzynarodowe: geneza, struktura, dynamika. Wydawn. Uniwersytetu Warszawskiego, 2000.

[7] Nye, J. S., The Future of Power, New York 2011.

[8] Pietraś, M., Piórko, K., Podmioty transnarodowe, w: M. Pietraś (red.), Międzynarodowe stosunki polityczne, Lublin (2006).

[9] Karns, M. A., Mingst ,K. A. and Kendall W. Stiles. International organizations: The politics and processes. Boulder,Colorado: Lynne Rienner Publishers, Inc, 2004.

[10] Patomäki, H. Global Tax Initiatives: The Movement for the Currency Transaction Tax, "Civil Society and Social Movements Programme Paper" 2007, no. 27, s. 1

[11] Patomäki, H. "The Tobin Tax and Global Civil Society Organisations: The Aftermath of the 2008-9 Financial Crisis." Ritsumeikan Annual Review of International Studies 8.1 (2009)

[12] Tobin, James. "On the efficiency of the financial-system." Lloyds Bank Annual Review 153 (1984)

[13] H. Patomäki, The Tobin Tax and Global Civil Society Organizations: The Aftermath of the 2008-9 Financial Crisis, „Ritsumeikan Annual Review of International Studies" 2009, vol. 8, no. 1, p. 3.

[14] Zachara, Małgorzata. Global governance: ład międzynarodowy po zakończeniu stulecia Ameryki. Kraków: Wydawnictwo Uniwersytetu Jagiellońskiego, 2012.

[15] Nakonieczna-Bartosiewicz, Justyna. "Pozasystemowe poszukiwanie sprawiedliwości w stosunkach międzynarodowych. Alterglobaliści. Ruchy oburzenia obywatelskiego. Haktywiści." Sprawy Międzynarodowe 2 (2012)

[16] Likosky B., Mueller M. L., Civil Society Intervention in the Reform of Global Public Policy, Proceedings from the IRG Ford Foundation International Seminar, Paris 17-18-19.04. 2007

[17] Tarrow S., Power in Movements: Social Movements and Contentious Politics, Cambridge 2011, p. 119-125.

[18] Breslow H., Civil Society, Political Economy, and the Internet, w: S. G. Jones (ed.), Virtual Culture: Identity and Communication in Cybersociety, London 1997, s. 236-257

[19] Deibert, R. J., International Plug'n Play? Citizen Activism, the Internet, and Global Public Policy, „International Studies Perspectives" 2000, no. 1, p. 256.

[20] Abdulla, R., Poell, T., Rieder, B., Woltering, R., & Zack, L. (2018). Facebook polls as proto-democratic instruments in the Egyptian revolution: The 'We Are All Khaled Said'Facebook page. Global Media and Communication, 14(1), 141-160.

Applying Ethics to Autonomous Agents

Tomasz Zurek[*]
Dorota Stachura-Zurek

1 Introduction

The speedy development of the field of science called artificial intelligence that we have been witnessing in recent years has triggered the equally astounding development of autonomous devices, which have evolved from objects of interest for SF writers into the tools we use every day and have already gotten quite used to. Although the existence of robot vacuum cleaners do not seem to pose serious risks, a number of more complex devices, like autonomous cars, may prove dangerous not only for their users, but also for other traffic participants. Moreover, following [15], we claim that the increasing autonomy of devices requires much more than specific limitations of their "freedom" of conduct (like Asimov's famous rules of robotics) and calls for the moral or ethical reasoning that should be a crucial internal element of the entire decision process.

On the other hand, we consider ourselves far from saying that we need anything like "machine ethics" in the sense that machines should work out their own ethical principles ([15] points out the possible risks of such an approach). For the users and witnesses of the work of autonomous devices, it is important that the machine should follow *human* ethical principles rather than "machine ethics" itself (whatever it would mean). Such an approach is coherent with the so-called weak AI assumption, meaning that we are attempting to create a machine which behaves *like* an intelligent agent, not the machine which *is* intelligent. We believe that such a view on the problem of ethical autonomous devices is, from practical point of view, much more appealing than the approach in which the machine creates its own "ethics" (even if it was possible, how could we guarantee that such ethics would be coherent with ours?).

A vast number of philosophical theories refer to human ethics and present various views on the mechanisms of human moral deliberations and decisions; none of them, however, seems complete and exhaustive. Among all those approaches most influential are: deontology, consequentialism, and virtue ethics. The first one assumes that human ethical choices are made with respect to, mainly deontic, rules

[*]Corresponding author — tomasz.zurek@mail.umcs.pl

and principles; the second one focuses on the analysis of the consequences of decisions; and the third one focuses on the concept of virtues and vices, where morally good actions exemplify virtues and morally bad actions exemplify vices.

Considering the above, we are going to present a discussion and a model of a decision making mechanism which implements some assumptions of two of the above mentioned ethical theories. Since the deontological approach has been already extensively discussed in many other papers (an in-depth analysis can be found in [21]), our aim is to examine how consequentialism and virtue ethics can be implemented in the decision making mechanism of an autonomous device.

The two main approaches to AI-based decision making mechanisms are known as knowledge-driven and data-driven. Although data-driven systems are dominating at present, they still have a number of important limitations. Many researchers point out that the way of overcoming the limitations of machine learning-based mechanisms is the development of hybrid systems which would connect the elements of both approaches. Although we agree with this statement, it is not our goal to present here a complete structure of such an approach. We aim at presenting our model of a knowledge-driven part of such a system which allows for implementing the two ethical theories mentioned above. The model will be created on the basis of the formal background of a value-based model of teleological reasoning presented in [25] and [26], further referred to as the GVR model.

2 Contribution

Our contribution can be summarized as follows:

- We present a discussion of the various approaches to ethics and ethical decision making in the light of autonomous devices;
- We distinguish particular theories which can be implemented in autonomous devices;
- We present a novel model of consequentialist and virtue ethics;
- We illustrate the model with an example;
- We present an in-depth discussion of our model in the light of the existing approaches.

3 Ethical theories

It is common to distinguish three major types of ethical theories: consequentialism, deontological ethics, and virtue ethics. We will continue to use this distinction for the purposes of our paper.

3.1 Consequentialism

Basically speaking, the main characteristics of consequentialist ethical theories is that normative properties depend on consequences [19]; therefore the best decision is the one which will entail most favorable consequences. Opinions differ on whether the consequences should be real, predicted, or predictable; whether reasoning based on consequences should apply to particular situations, or rather to constructing general rules governing ethical considerations; whether there should only be one scale with the general value "pleasure" or "happiness" or should more values be used; and so on.

The founding father of the modern utilitarian school of ethical thinking, Jeremy Bentham, believed that the goodness of actions lies in the maximization of happiness and supported the concept of *utility*, understood as the greatest happiness for the greatest number of people [7]. Bentham's follower, J.S. Mill, developed the utilitarian theory to include more elaborate concepts of, for example, what pleasure and pain actually are. Since then, a number of theories drawing from Bentham's and Mill's utilitarianism have been formed, some of which have drifted away quite far, and they have been functioning under the label of consequentialism coined by Anscombe in her famous essay [3].

For the purposes of this paper, we accept a version of consequentialism understood as:

- agent neutral, meaning that evaluation of the consequences does not change regardless of whose perspective is used;
- preferential, meaning that what is good is the fulfilment of the agent's preferences understood not as a sensation, but as a state of affairs;
- pluralistic, meaning that there are irreducible plural values (as opposed to the monist one fundamental value — "pleasure" or "happiness" or "overall good") to be considered;
- act-based, meaning that consequences are evaluated for each situation.

These assumptions seem legit from the point of view of applying consequentialism in autonomous devices. They attempt to ensure predictable and explainable decisions being made by agents; they take into consideration the fact that autonomous devices are usually created for specific tasks, and do not deal with such a variety of decisions as humans; they in a way overcome a serious limitation of consequentialist theories in humans, namely the impossibility to calculate all consequences for all possible scenarios in a given decision situation [19]. We could therefore cautiously assume that applying consequentialist patterns, or elements of such decision-making in autonomous devices, could yield some interesting results.

3.2 Deontology

Deontological ethics, sometimes referred to as the most popular form of non-consequentialism, assumes that some acts must not be performed even if it may possibly lead to bad results [16]. In other words, some choices must be morally forbidden, regardless of what might happen afterwards — "good" or "bad"; it is the conformity with moral norms which makes an action morally right [2]. Some crucial deontological thinkers include philosophers like Kant, Pritchard, and Ross; more recently — Kamm, Nozick, or Korsgaard [16]. Numerous decision making models have been founded on deontological systems so far, but their analysis would be out of scope of this paper (for an overview, see e.g. [21]). Deontological ethical systems have been analyzed in the light of AI (e.g., for an examination of applying Kantian ethics in autonomous devices of the military sphere, see [22]). Since the model we are going to introduce does not really rely on deontological premises, the discussion of deontology will not be further extended.

3.3 Virtue ethics

The theories labeled as virtue ethics rely on the centrality of the concept of virtue. This and the two other concepts fundamental to virtue ethics include [13]:

- arête — that is virtue, understood as aptitude / disposition / an excellent trait of character, which makes a person good;
- phronesis — practical wisdom; the capacity to reason about virtues; a meta-virtue on which virtuous decisions depend [9];
- eudaimonia — the end of human existence; a state of real, true happiness, human flourishing.

As opposed to focusing on the overall good consequences, or trying to determine what is wrong and what is right, virtue ethics is about what kind of people we want to be and how we wish to shape our lives. The fulfilment of the human potential, full development of oneself as a person — eudaimonia — can be seen both as a result of virtuous life, but also as leading a virtuous life as such [16]. Modern virtue ethics include a number of variations inspired by Aristotle, Plato, the Stoics, Aquinas, but also philosophers like Confucius [18] [24], or Hume and Nietzsche [20]. The understanding of virtue ethics for the purpose of implementation in a decision making mechanism in this paper will be rooted in a broadly eudaimonist ethical framework, with the crucial concepts seen as follows:

- virtues are the foundation for decision making;
- phronesis consists in the reasoning mechanism;
- eudaimonia is achievable.

Given that we perform our research in the spirit of weak AI, applying *human* ethics to artificial agents, it follows that we expect them to *behave* in a certain sort of manner, so that the results would be advantageous to humans. Virtue ethics clearly focuses on the agent as a person; the objective of such research would not be then to create a virtuous machine imitation of a human in pursuit of its happiness, but a device that is able to make a decision which would be considered virtuous by a human. We hardly entertain the idea of eudaimonia as a *machine's* happiness or flourishing; it can obviously only be applied to humans, and thus in this case would involve ensuring happiness/flourishing for humans *through* the decisions of an autonomous device.

4 GVR model

We will begin with a summarized discussion of the model of teleological reasoning from [25], further referred to as the GVR model. Firstly, the naming convention will be presented:

- By upper case letters we denote sets;
- By lower case letters we denote propositions;
- Subscripts denote names of propositions;
- Superscripts denote names of sets;
- Greek letters denote functions;
- Other symbols will be defined later (except trivial logical and set-theory ones).

Definition 1 (State of affairs) *Let $S = \{s_0, s_1, s_2, ...\}$ be a finite, non-empty set of propositions. Each proposition represents one state of affairs. Let γ be a function which returns 1 if a given state of affairs is true and 0 if not. One and only one element from set S can be true: if $(\gamma(s_y) = 1)$, then $\forall_{s_x \in S : s_x \neq s_y}(\gamma(s_x) = 0)$. s_0 is the initial state of affairs.*

We assume that all state of affairs are separate. If we would like to model a case in which more than one state of affairs will be achievable simultaneously, then they should be divided into separate decisions: for example, having two state of affairs s_a, s_b, an agent have such available state of affairs: $S = s_a, s_b, s_{a,b}$. Such an approach may cause a combinatorial explosion, but since set S contains only possible states, then the number of possibility will be significantly lower than 2^n.

Definition 2 (Actions) *As an action we understand an activity which carries a transition from a certain state of affairs to another state of affairs. Actions will be represented by propositions from set $A = \{a_1, a_2, \dots a_k\}$.*

It is worth noticing that a particular action cannot be performed in every state of affairs. The set of all possible actions in all possible states of affairs we denote as AS ($AS \subsetneq A \times S$). Set AS is a set of pairs $AS = \{as_{i,j}, as_{k,w}, ...\}$ in which $as_{i,j} = \langle a_i, s_j \rangle, as_{k,w} = \langle a_k, s_w \rangle$ (the first subscript denotes the name of an action, the second subscript denotes the name of a situation). Each pair represents that a given action (for example, a_i) can be performed in a given state of affairs (for example, s_j). By AS^j (where $AS^j \subset AS$) we denote a set of actions possible to perform in a state of affairs s_j.

Function $\delta : AS \to S$ returns the result of performing an action a_i in a state of affairs s_j. By $\delta(as_{ij}) = s_y$ where $as_{ij} = \langle a_i, s_j \rangle$ we denote that the result of performing an action a_i in a state of affairs s_j is s_y.

The presentation of our model will be illustrated by a simple running example (adopted form [25]:

Definition 3 (Transition process) *Let $\varepsilon : AS \times S \to S$ be a partial function which represents performing an action a in a state of affairs s. If $\delta(as_{i,j}) = s_y$ and $\gamma(s_j) = 1$ (the result of performing an action a_i in a state of affairs s_j is s_y), then performing $\varepsilon(as_{i,j})$ causes changing $\gamma(s_j) = 0$ and $\gamma(s_y) = 1$.*

Definition 4 (Situation) *By a situation x_n we understand a particular state of affairs or a result of a particular action performed in a given state of affairs. A set of situations X is a union of sets of states of affairs and results of actions: $X = \{S \cup AS\}$. By $x_n \in X$ we denote an element from set X. By X^j we denote a set of situations available from a state of affairs s_j: $S^j = \{s_j \cup AS^j\}$ (we introduce the concept of situation because state of affairs and actions can become decision options and it is easier to use symbol X instead of $\{S \cup AS\}$)*

Definition 5 (Values) *We have to separate the two meanings of the word value: a value may be understood as a concept or as a process.*

1. *Value as an abstract concept which allows for the estimation of a particular action or a state of affairs and influences one's behaviour. V is a set of values: $V = \{v_1, v_2, \ldots v_n\}$*
2. *Value as a process of estimation of the level of extent to which a particular situation (state of affairs and / or action) x promotes a value v_i. By $v_i(x)$ we denote the extent to which x promotes a value v_i. By $V(X)$ we denote the set of all valuations of all situations.*

It is important to emphasize that values can be promoted to a certain degree by a particular state of affairs or action: $v(as_{i,j})$ represents the degree to which a value v is promoted by a state of affairs which is the result of performing an action a_i in a state of affairs s_j.

Although, similar to [25], we are not going to impose here any particular way of representing the levels of promotion of values, we are not excluding to represent them as numbers. For example, in [26] they were expressed by numbers from range $< 0; 1)$.

By $V^i(X)$ we denote the set of all possible extents to which a value v_i from set V may be promoted by any possible situation $x \in X$.

A partial order $O_i = (\geq; V^i(X))$ represents the relation between extents to which values are promoted: $v_i(x_n) \geq v_i(x_m)$ means that $x_n \in X$ promotes a value v_i to a no less extent that $x_m \in X$. If $v_i(x_n) \geq v_i(x_m)$ and $v_i(x_m) \geq v_i(x_n)$, then extents to which a situation x_n and x_m promotes a value v_i are equal ($v_i(x_n) = v_i(x_m)$). If $v_i(x_n) \geq v_i(x_m)$ and $v_i(x_n) \neq v_i(x_m)$, then $v_i(x_n) > v_i(x_m)$.

In real-life reasoning people do not rely only on a comparison of the levels of promotion of one value; usually, they compare the levels of promotion of various values. Theoretically speaking, they are incompatible, but practically, people compare not only the levels of promotion of various values, but also the levels of promotion of various sets of values.

Definition 6 (Sets of values) *By $V^Z \subset V$ we denote a subset (named Z) of a set of values V which consists of values: $v_i, v_j, \ldots \in V^Z$.*

By $V^{x_i} \subset V$ we will denote a set of values promoted by a situation X_i.

Definition 7 (The level of promotion of a set of values) *By $V^Z(x_n)$ we denote a set of estimations of the levels of promotion of values constituting set V^Z by a situation $x_n \in X$. If $V^Z = \{v_z, v_t\}$, then $V^Z(x_n) = \{v_z(x_n), v_t(x_n)\}$.*

By $V^{x_i}(x_i)$ (when the upper script contains the name of a situation) we denote a set of estimations of the levels of promotion of all values promoted by a situation $x_i \in X$.

Definition 8 (Value-extent preference) *A partial order $OR = (\triangleright; 2^{V(X)})$ represents a preference relation between various values and various sets of situations: $V^Z(x_n) \triangleright V^Y(x_m)$ means that the extent to which values from set V^Z are promoted by a situation x_n is preferred to the extent to which values from set V^Y are promoted by a situation x_m.*

Properties of the OR relation:

- Relation OR is a strict partial order, hence it is irreflexive, asymmetric, and transitive.
- If V^Z is a set of values promoted by a situation x_1 ($V^Z \subseteq V^{x_1}$) and $V^X \subseteq V^Z$, then:
$$V^X(x_1) \triangleright V^Y(x_2) \Rightarrow V^Z(x_1) \triangleright V^Y(x_2).$$

How to determine whether the extents to which all values promoted by one situation are preferred to the extents to which all values are promoted by another situation? This is not a trivial issue, because we have to balance between different levels of promotion of different values. This problem has been extensively discussed in [25] and [26], where two different approaches to this problem were presented. Although we believe that this topic still requires further development, it is outside the scope of our paper.

On the basis of order OR a new relation may be constructed:

Definition 9 (Preference) *A partial order $OP = (\gg; X)$ represents a preference relation between various situations: $x_n \gg x_m$ means that a situation x_n is preferred to a situation x_m.*

Preference between various situations can be derived on the basis of preferences between the extents to which various values are promoted by the analyzed situations: If the extents to which values are promoted by a situation x_1 are preferred to the extents to which values are promoted by a situation x_2, then a situation x_1 is preferred to a situation x_2:

$$V^{x_1}(x_1) \triangleright V^{x_2}(x_2) \Rightarrow x_1 \gg x_2 \quad (1)$$

4.1 Goals

As it has already been noticed, we need to draw a distinction between a few different kinds of goals ([25]):

Definition 10 (Abstract goals) *Abstract goals are goals represented by the minimal extents to which a particular situation promotes a given set of values (in our model the valuation of any target state of affairs can be considered only with the valuation of the action which leads to it, hence by a situation we will understand a particular action performed in a particular state of affairs):*

- $GA = \{ga_1, ga_2, ...\}$ — a set of abstract goals.
- By $v_n min(ga)$ we denote the minimal extent to which the promotion of a value v_n satisfies a goal ga.
- By $v_n(x_1) \geq v_n min(ga)$ we denote that a goal ga is satisfied by a situation x_1 with respect to a value v_n.
- By $v_n \in ga$ we denote that the minimal extent of a given value v_n is declared in a goal ga: $v_n \in ga \leftrightarrow \exists v_n min(ga)$.

Definition 11 (Abstract unreachable goals) *Abstract unreachable goals are abstract goals where the agent desires one value to be promoted as much as possible, while other values to be promoted no less than to a certain extent:*

- $GUA = \{gua_1, gua_2, ...\}$ — a set of abstract unreachable goals.
- By $v_n min(gua)$ we denote the minimal extent to which the promotion of a value v_n satisfies a goal gua.
- Function $\omega : GUA \rightarrow V$ returns the value which should be promoted to the maximal possible extent. A value $v_m : \omega(gua) = v_m$ will be called the key value of an abstract unreachable goal gua. Note that value v_m is also the ordinary goal of a goal gua ($v_m \in gua$) and it has its own minimal extent to which it should be promoted.

Definition 12 (Material goals) *Material goals are particular situations which satisfy given abstract goals.*

- $GM = \{gm_1, gm_2, ...\}$ — a set of material goals.
- A material goal is a particular action performed in a given state of affairs where: $gm_k = (as_{t,m})$, while $as_{t,m} \in AS$.
- By $sat(gm_k, ga_l)$ we denote that a material goal gm_k satisfies an abstract goal ga_l and is possible to achieve:

$$sat(gm_k, ga_l) \leftrightarrow$$

$$(\forall_{v_i \in ga_l}(v_i(as_{t,m}) \geq v_i min(ga_l)) \wedge as_{t,m}) \wedge$$

$$(gm_k = (as_{t,m}) \in AS \wedge \gamma(s_m) = 1).$$

Definition 13 (Practical goal) *A practical goal is a material goal which is achievable, which satisfies the agent's abstract goal, and which the agent is going to reach. A practical goal will be denoted as gp. The agent can only have one practical goal at a time.*

4.2 Inference rules

In our model, inference is based on the defeasible inference rules (by defeasibility we understand the possibility of defeating the conclusion obtained with the use of such rule by other, stronger, rule. For more detailed discussion of defeasibility of inference rules see [17]). In [25], a number of argumentation schemes are proposed which we will use as inference rules in our model. Argumentation schemes are forms of argument which represent stereotypical patterns of human reasoning [8]. Argumentation schemes in computational models are usually interpreted as defeasible inference rules and they constitute a part of a whole argumentation framework providing a basis on which the process of reasoning is conducted. Such a framework for GVR model was introduced in [26] along with new, fully-formalized versions of the schemes.

Inference rules used in our model have an antecedent part and a consequence part, separated by a double bar which stands for the sign of defeasible inference. Below is presented a selected list of inference rules[1]:

AS2 Generalized practical reasoning[2]: If in circumstances s_m performing an action a_t is preferred to remaining in s_m, $as_{t,m}$ is also preferred to any situation available from states_m, and $as_{t,m} \in AS$, then an action a_t should be performed:

$$\frac{\begin{array}{c}\exists_{s_m \in S} \exists_{as_{t,m} \in AS} \forall_{as_{k,m}} : \\ VO^{as_{t,m}}(as_{t,m}) \rhd VO^{s_m}(s_m) \wedge \\ VO^{as_{t,m}}(as_{t,m}) \rhd VO^{as_{k,m}}(as_{k,m})\end{array}}{\varepsilon(as_{t,m})}$$

AS3 Reasoning with abstract goals: If in the current circumstances s_m achieving an abstract goal ga_k is possible by an action a_t performed in s_m, then an action $as_{t,m}$ becomes the practical goal gp:

$$\frac{\begin{array}{c}\exists_{ga_k \in GA} \exists_{s_m \in S} \exists_{as_{t,m} \in AS} : \\ \gamma(s_m) = 1 \wedge \\ sat(as_{t,m}, ga_k)\end{array}}{gp = as_{t,m}}$$

AS5 Goal-driven practical reasoning In the current circumstances s_m, in order to achieve the practical goal gp, an action a_t should be performed:

$$\frac{\begin{array}{c}\exists_{s_m \in S} \exists_{as_{t,m} \in AS} : \\ (\gamma(s_m) = 1) \wedge \\ (gp = as_{t,m})\end{array}}{\varepsilon(as_{t,m})}$$

[1]To preserve the cohesion of designations, the names of argumentation schemes will be the same as used in [25].

[2]This is a modified version of the AS2 from [26].

4.3 Inference mechanism

[25] discusses a number of inference rules as well as a simple argumentation framework which allows for reasoning about goals and making decisions concerning the fulfilment of these goals. An exhaustive discussion (including the problems of conflict resolving, etc.) of the inference rules' structure and properties is included in [25] and [26].

5 GVR and various ethical theories

In this section we demonstrate how various ethical theories can be represented with the use of the GVR model.

We claim that the GVR model allows for modeling of the process of moral decision making with the use of various ethical theories. In our opinion, the utilization of different inference rules allows for representing the reasoning processes guided by ethical theories.

The model will be illustrated by a simple running example:

Example 1 (Running example) *Suppose an autonomous car in right-hand traffic. On both sides of the lane are straight lines (it is not allowed to cross those lines). On the right side of the lane is a wall. Suppose that suddenly a child appears in front of the car. The child is too close for the car to stop. The car has three options:*

1. *turn left and cross the left line, which results in breaking the traffic law;*
2. *turn right and hit the wall, which results in destroying the car and causing injuries to passengers;*
3. *go straight, which results in hitting and possibly killing the child.*

5.1 GVR and consequentialism

Consequentialism assumes that the decision options are evaluated in the light of their expected consequences. The agent chooses the option in which the overall evaluation of consequences is better than in other options. In our model, following the pluralistic approach, we introduce a set of values instead the general notion of the concept of happiness. Moreover, we believe that values do not have binary character only, but they can be promoted to various extents. On the basis of the above, we assume that the levels of promotion of values can be a basis of the comparative analysis of the decision options' consequences. Therefore, the agent analyses the consequences of the considered options represented by the levels of promotion of values. S/he chooses the option which promotes values to the higher level than other ones.

Such a understanding of consequentialism can be represented with the use of the GVR model. In the model, the decision options are represented by situations (see definition 4). Set X represents all available states of affairs with actions allowing for achieving them. V is the set of values; by $v_i(x_j)$ we denote the level of promotion of value $v_i \in V$ by situation $x_j \in X$ (the consequence of choosing decision option x_j interpreted in the light of value v_i). Every decision option promotes every value from set V to a particular level. The agent's decision is made by the comparison of consequences (the levels of promotion of values) of all available decision options (situations from set X).

Intuitively, the simplest case is when there is one situation ($x_k \in X$) which promotes every value from set V to a higher extent (or at least no lower) than other situations from set X:

$$\forall_{v_l \in V} \forall_{x_j \in X} \exists_{x_k \in X} (v_l(x_k) \geq v_l(x_j))$$

Since real life situations are usually not so simple (the same situation strongly promotes only some values, while other ones are promoted to much lower extents), the system needs a mechanism of determining the overall preference between the extents of values' promotion. The GVR model introduces order OR which represents such a preference, while [26] introduces the mechanism of calculating the preference[3].

The basic assumption of consequentialism is that the agent chooses the option in which the overall evaluation of consequences is preferred to other options. In order to represent such a pattern, we have to introduce a new inference rule:

AS6 Consequentialist reasoning: If in circumstances s_m performing an action a_t is preferred to remaining in s_m and consequences of $as_{t,m}$ are preferred to consequences of any other situation (action available from state s_m), then action a_t should be performed:

$$\begin{array}{c} \exists_{s_m \in S} \exists_{as_{t,m} \in AS} \forall_{x \in X} : \\ x \neq as_{t,m} \\ as_{t,m} \gg x \\ \hline\hline \varepsilon(as_{t,m}) \end{array}$$

The above inference rule can be illustrated by the example:

Example 2 (Running example cont.) *Assume the above example with the autonomous vehicle. Let s_{car} be a state of affairs in which the car is now (described in the above example). We have 3 decision options (set X contains):*

1. *to turn right: $as_{right,car}$*
2. *to turn left: $as_{left,car}$*
3. *to continue going straight: $as_{straight,car}$*

Let's assume three values:

1. *observing the law: v_{law}*
2. *life of pedestrian (child): v_{child}*

[3]In this paper we are not going to discuss how to obtain OR and, for the sake of this study, we assume that we obtained it with the use of the mechanism presented in [26]. For a more profound analysis of this issue, see [25] which presents a discussion of the relations between orders from set O and OR, and [26] which introduces the mechanism of obtaining OR on the basis of the relative levels of promotion of values (represented by numbers) with the use of function Θ.

3. life of passenger of the car: $v_{passenger}$

Every decision option (situation) and actual state of affairs promotes every value to a particular extent: $v_{law}(s_{car})$, $v_{law}(as_{right,car})$, $v_{law}(as_{left,car})$, $v_{law}(as_{straight,car})$, $v_{child}(s_{car})$, $v_{child}(as_{right,car})$, $v_{child}(as_{left,car})$, $v_{child}(as_{straight,car})$, $v_{passenger}(s_{car})$, $v_{passenger}(as_{right,car})$, $v_{passenger}(as_{left,car})$, $v_{passenger}(as_{straight,car})$.

Let us assume the following orderings in O*:*

- $O_{law} = \{v_{law}(as_{straight,car}) > v_{law}(as_{right,car}),$ $v_{law}(as_{straight,car}) > v_{law}(as_{left,car})\}$
  From the point of view of observing the traffic rules, going straight (the car does not cross the straight lines) is preferred to both turning right or turning left.
- $O_{child} = \{v_{child}(as_{left,car}) > v_{child}(as_{straight,car}),$ $v_{child}(as_{right,car}) > v_{child}(as_{straight,car})\}$
  From the point of view of the life of a pedestrian (child), both turning left and turning right is preferred to going straight, which can result in hitting the child.
- $O_{passenger} = \{v_{passenger}(as_{left,car}) > v_{passenger}(as_{right,car}),$ $v_{passenger}(as_{straight,car}) > v_{child}(as_{right,car}),\}$
  From the point of view of the life of the vehicle passenger, both turning left and going straight is preferred to turning right (there is wall on the right side, hitting the wall can cause danger to the passenger's life).

In order to model consequentialist reasoning, we need orders OP *and* OR.

Order OR *contains:* $\{VO^{as_{left,car}}(as_{left,car}) \rhd VO^{as_{straight,car}}(as_{straight,car}),$ $VO^{as_{left,car}}(as_{left,car}) \rhd VO^{as_{right,car}}(as_{raight,car})\}$

The above, on the basis of def. 9, can be expressed as: $as_{left,car} \gg as_{straight,car}$ *and* $as_{left,car} \gg as_{right,car}$,

Which means that turning left is preferred to both turning right and going straight. In other words, the consequences of breaking the law (crossing the line) are preferred to injuring the pedestrian or car passenger.

On the basis of the above and inference rule AS6, one can conclude that since turning left is preferred to any other action from set X*, then turning left should be performed.*

5.2 GVR and virtue ethics

Eudaimonist virtue ethics is rooted in ancient Greek philosophy, especially in the works of Aristotle. Although it is probably the oldest of the major ethical paradigms, it is the least implemented one in the light of possible application to autonomous devices. The human-centered character of this theory and the ambiguous character of the definition of virtue are the most important reasons for the lack of successful models of virtue-based ethical decision making.

In order to create a model of virtue-based ethical decision making system, it is necessary to introduce a clear definition of what we understand as virtue and what does it mean that a particular decision or act exemplifies a virtue.

We realize that we attempt to formalize a very abstract and obscure concept, but we are convinced that the formalization of the concept of virtue is necessary for the sake of our goal. The main assumptions on the basis of which we can introduce and formalize the concept of virtue are:

- a virtue represents a human's desired moral attitude;
- virtue should be related to values or, in other words, values represents virtues;
- a given decision option can satisfy or not a given virtue;
- a decision is ethical if it satisfies a virtue or virtues.

One can find the above assumptions as constraining the initial meaning of this term, but we believe that such simplifications are inevitable in successful modeling of complex real life concepts.

On the basis of the above assumptions, we can assume that a virtue can be represented by a set of minimal extents to which the decision should promote a particular set of values.

Definition 14 (Virtue) *Let virtue will be represented by the minimal extents to which a particular situation should promote a given set of values:*

- $VIRTUES = \{vrt_1, vrt_2, ...\}$ — a set of virtues
- By $v_n min(vrt)$ we denote the minimal extent to which the promotion of a value v_n satisfies a virtue vrt.
- By $v_n(x_1) \geq v_n min(vrt)$ we denote that a virtue vrt is satisfied by a situation x_1 with respect to a value v_n.
- By $v_n \in vrt$ we denote that the minimal extent of a given value v_n is declared in a virtue vrt: $v_n \in vrt \leftrightarrow \exists v_n min(vrt)$.

In other words, virtue can be seen as a kind of abstract goal, but the goal which does not represent the agent's personal needs or desires, but the ideal moral attitude.

Having a model of virtue, it is possible to define when a particular decision option (situation) will satisfy a given virtue:

Definition 15 (Satisfaction of a virtue) *Assuming a situation $x_i \in X$ and virtue $vrt_j \in VIRTUES$, situation x_i will satisfy virtue vrt_j: $Vsat(x_i, vrt_j)$ iff* $\forall_{v_n \in vrt} : v_n(x_i) \geq v_n min(vrt_j)$

The definition of satisfaction of a virtue relates to only one virtue. Let $VSAT(x_i)$ denote that a particular decision option satisfies all virtues:

Definition 16 (Satisfaction of all virtues) *We say that a particular situation $x_i \in X$ satisfies all virtues iff:*
$VSAT(x_i)$ *iff* $\forall_{vrt_i \in VRT} Vsat(x_i, vrt_i)$

The above definitions allow us to evaluate a particular decision in the light of virtues, hence we can say that a particular decision exemplifies a given virtue if this virtue is satisfied by the situation. Additionally, we can say that a particular situation is moral if it satisfies all virtues.
The possibility of evaluation of a particular situation in the light of virtues allows us to update the existing inference rules in order to model moral reasoning:

AS2' Generalized moral practical reasoning: If in circumstances s_m performing an action a_t is preferred to remaining in s_m, a_t is preferred to other actions available from S_M, $as_{t,m} \in AS$, and $as_{t,m}$ satisfies all virtues, then action a_t should be performed:

$$\exists_{s_m \in S} \exists_{as_{t,m} \in AS} \forall_{as_{k,m} s.t. VSAT(as_{k,m})} : \frac{\begin{array}{c} VO^{as_{t,m}}(as_{t,m}) \rhd VO^{s_m}(s_m) \\ VO^{as_{t,m}}(as_{t,m}) \rhd VO^{as_{k,m}}(as_{k,m}) \wedge \\ VSAT(as_{t,m}) \end{array}}{\varepsilon(as_{t,m})}$$

AS3' Reasoning with abstract goals: If in the current circumstances s_m achieving an abstract goal ga_k is possible by an action a_t performed in s_m and $as_{t,m}$ satisfies all virtues, then action $as_{t,m}$ becomes a practical goal gp:

$$\exists_{ga_k \in GA} \exists_{s_m \in S} \exists_{as_{t,m} \in AS} : \frac{\begin{array}{c} \gamma(s_m) = 1 \wedge \\ sat(as_{t,m}, ga_k) \wedge \\ VSAT(as_{t,m}) \end{array}}{gp = as_{t,m}}$$

The above inference rules are extended versions of the AS2 and AS3 inference rules to which we add an extra condition which ensures that a chosen decision is moral.
The model can be illustrated by the example:

Example 3 (Running example, cont...) *Suppose the above example with the autonomous car. We would like to make a decision on the basis of a generalized moral practical reasoning inference rule (AS2').*
In order to do that, it is necessary to declare the virtue:
Let us assume one virtue with minimal extents of three values mentioned earlier: $vrt_{car} = \{v_{law} min(vrt_{car}), v_{child} min(vrt_{car}), v_{passenger} min(vrt_{car})\}$.
In order to compare the extents to which particular decisions promote different values with the virtue, we have to add the following orderings to subsets of O:

- *Let* O_{law} *additionally contain:* $\{v_{law}(as_{straight,car}) > v_{law} min(vrt_{car}$
  $v_{law}(as_{left,car}) > v_{law} min(vrt_{car}$
  $v_{law}(as_{right,car}) > v_{law} min(vrt_{car}\}$
  From the point of view of observing the traffic rules, every decision is above the threshold (we assumed that crossing the line is not a serious violation of the rules).

- $O_{child} = \{v_{child}(as_{left,car}) > v_{child}min(vrt_{car}),$
  $v_{child}(as_{right,car}) > v_{child}min(vrt_{car}),$
  $v_{child}min(vrt_{car}) > v_{child}(as_{straight,car})\}$
  From the point of view of the life of a pedestrian (child), both turning left and turning right promote the value to the higher extent than the threshold (the life of a pedestrian is preserved); going straight promotes the life of a pedestrian below the threshold (this decision can result in hitting the child).

- $O_{passenger} = \{v_{passenger}(as_{left,car}) > v_{passenger}min(vrt_{car}),$
  $v_{passenger}min(vrt_{car}) > v_{passenger}(as_{right,car}),$
  $v_{passenger}(as_{straight,car}) > v_{passenger}min(vrt_{car})\}$
  From the point of view of the life of the vehicle passenger, both turning left and going straight is above the threshold, while turning right is below (hitting the wall can endanger the passenger's life).

In order to make a decision, the autonomous car will use the generalized moral practical reasoning inference rule (AS2'): The car is in the state of affairs $s_{car} \in S$ and it has three possible decisions to make (X contains):

1. *turn right: $as_{right,car} \in X$*

2. *turn left: $as_{left,car} \in X$*

3. *continue going straight: $as_{straight,car} \in X$*

Now we examine when the conditional part of AS2' will be satisfied. It will be satisfied if there will be available an option which is preferred to other options (1st line) and this option will satisfy the virtue (2nd line).

Since order OR contains:
$\{VO^{as_{left,car}}(as_{left,car}) \rhd VO^{as_{straight,car}}(as_{straight,car}),$
$VO^{as_{left,car}}(as_{left,car}) \rhd VO^{as_{right,car}}(as_{raight,car}),\}$
then we can say that the action $as_{left,car}$ is preferred to other ones (1st line of the conditional part of AS2' is satisfied).
Now the 2nd line of the conditional part is examined:
Because:

- $v_{law}(as_{left,car}) > v_{law}min(vrt_{car})$*: the level of promotion of value V_{law} is above the threshold of virtue vrt_{car},*

- $v_{child}(as_{left,car}) > v_{child}min(vrt_{car})$*: the level of promotion of value V_{child} is above the threshold of virtue vrt_{car},*

- $v_{passenger}(as_{left,car}) > v_{passenger}min(vrt_{car})$*: the level of promotion of value $V_{passenger}$ is above the threshold of virtue vrt_{car},*

then we can say that action $v_{law}(as_{left,car})$ satisfies the virtue and it can be performed: $\varepsilon(as_{left,car})$.

6 Discussion

In this section we present a discussion of our approach in the light of moral philosophy and the existing implementations of consequentialist and virtue ethics in autonomous devices.

6.1 Moral philosophy and the model

In this section we will discuss how our model addresses the assumptions of both analysed theories.

6.1.1 Consequentialist ethics

The key element of the consequentialist approach is that the basis of evaluation of decisions are their consequences. Since the decisions in our model are made on the basis of the levels to which values are promoted by decisions *and* their consequences (state of affairs to which a particular action leads), then we can say that our model fulfills the key assumption of consequentialist ethics. We have also initially assumed that our model would be founded on a particular version of consequentialism, that is:

- agent neutral – the implementation of our mechanism with the same knowledge in different agents will produce the same results (evaluation of the consequences does not change regardless of whose perspective is used);
- preferential – the basis for the decision are levels of promotion of values by the action and state of affairs to which the action leads;
- pluralistic – we introduce a set of values instead of one overarching value ("pleasure" or "happiness"). However, the levels of promotion of values are the basis on which the overall evaluation of the decision is made (the decisions are compared with the use of order OR), which is coherent with the crucial assumption of consequentialism;
- act-based – all available decision options in a particluar situation are taken into consideration.

6.1.2 Virtue ethics

We assumed that our model is created within the eudaimonist ethical framework. In such an approach the fulfilment of the human potential, full development of oneself as a person — eudaimonia — can be seen as a result of virtuous life. The key concept is of such an approach is the concept of virtue. In our model, by a virtue we understand the minimal acceptable levels of promotion of a set of values. Such an understanding of a virtue is, obviously, a simplification of this concept, but it allows for representing the key elements of virtue ethics in autonomous devices.

In section 3.3. we assumed that virtues are the basis for decisions and euadaimonia is achievable. Such a view on virtues and eudaimonia implies the binary character of this concepts: virtue can be fulfilled (or not) and eudaimonia can be achieved (or not). Since virtuous life should lead to eudaimonia, then we can assume that if a device makes decisions which fulfill the virtues, then it achieves eudaimonia (note that by eudaimonia we do not understand "eudaimonia" of a machine, but eudaimonia of humans). In other words, if a device makes a decision which promotes values to the levels above the thresholds established by virtue, then it is moral, i.e., brigs about eudaimonia.

Another key concept of virtue ethics is phronesis, understood as practical wisdom or – more suitably in this case – the capacity to reason about virtues. In our model this concept is represented by the whole reasoning mechanism containing sets S, AS, X, $V(X)$, orders O, OR, OS, inference rules, and inference mechanism. On the basis of this mechanism a device can infer which decision leads to eudaimonia, i.e., which decision satisfies all virtues.

6.2 Existing approaches to modeling ethical decision making

Firstly, we have to explain why we chose to use a knowledge-based mechanism instead of the most popular machine learning-based ones. The main reason for our choice is that ML-based systems function in the so-called black-box style, and they do not allow for explaining their knowledge and decisions. It is worth noticing that the lack of possibility of explaining decisions and the uncertainty of reasons for the decisions made by the ML-based devices is one of the key disadvantages of the ML-based autonomous devices, especially those which can be dangerous to humans (the device should follow clear and understandable instructions and have predictable behavior). On the basis of the above, we assumed that the part of the system which is responsible for ethical decision making should be constructed on the basis of the knowledge driven paradigm rather than the data-driven one. This does not exclude the possibility of using ML-based mechanisms in other elements of autonomous devices, like object detection, decision option extraction, evaluation of decision options, etc. However, this is beyond the scope of this paper.

Although the ethics of autonomous agents is the object of debate in a number of papers, there is a very limited number of approaches implementing aspects of consequentialist ethics and, even fewer addressing virtue ethics. The authors of [21] introduce a comprehensive survey of various models of ethical reasoning. They noticed that most of the models are constructed on the basis of deontological ethics. The authors recognize 9 papers which model censequentalist ethics and a few more implementing hybrid approaches (connecting deontological and consequentialist ethics). Most of the models of consequentialist ethics are based on the utility theory, without distinguishing other values which constitute the general utility. In comparison, in our work, following the pluralistic theory, we have assumed that consequences should be modeled in the light of a set of values whose overall evaluation serves as the criterion in the agent's choice. Moreover, most of the models presented in [21] are proposed for implementation in specific devices designed for particular tasks (e.g., household robots or the continuous prisoner's dilemma), without consideration of a more general perspective on the functioning of such a device. In our approach we provide a model which is more comprehensive and avoid narrowing it to a particular usage.

The authors of [21] claim not to recognize any "pure" implementation of virtue ethics, but they do notice that some of them use elements of virtue ethics. However, since [21] do not introduce a discussion of the concept of virtue, then it is not quite clear how they actually recognize these virtue ethics elements.

Most of the formally-oriented papers discussed in the [21] do not present an in-depth analysis of the ethical theories, focusing on the formal and implementational aspects only (they usually merely introduce a basic definition of the ethical theory). In order to overcome this limitation, in our paper we have discussed our approach in the light of different variants of ethical theories (see section 2).

Below we present several interesting models of consequentialism and virtue ethics (some of which were discussed in [21]) with some comments referring to our model.

The authors of [12] present the analysis of the behaviour of agents equipped with elements of utilitarian and virtue ethics in Continuous Prisoner's Dilemma. Every agent in the experiment has two parameters: resource and reputation. In order to represent utilitarian ethics, the authors compare the total sum of resources of all agents: the higher the sum, the more ethical behaviour. In order to model virtue ethics, the authors evaluate the cooperation level and compare it to the threshold. The overall evaluation by the agent is made on the basis of both utilitarian and virtue ethics premises. [12] has some elements similar to our model (the utilisation of threshold in the representation of the virtue ethics), but the general approach is much more simplistic: the authors of [12] use only one value in the utilitarian and virtue based perspective, there is no reasoning mechanism; the authors do not introduce the mechanism which may allow for making decisions, but they try to model the behaviour of agents involved in a specific kind of the Prisoner's Dilemma. Moreover, it is not quite clear how they understand a virtue. If the virtue is connected with the action (as claimed by the authors), the paper could arguably exemplify a deontological approach.

The authors of [23] present a very simplistic approach to model utilitarian ethics. Using SOAR architecture (State, Operator, and Result) and 3 rules of robotics (Asimov), they create a robot supporting the preparation of meals for a family. The robot evaluates an order made by a family member in the light of health issues and has to choose between one of the three options (obey, disobey, or partially obey the order). Each option is evaluated by a utility score, representing how much a given option influences health (a positive number represents that the order is good for health, a negative means that it is not). The total sum of influence is the basis for decision. The authors of [23] focus on the construction of the robot and fail to introduce any broad approach to modeling a utilitarian perspective on ethical behavior in robots. The paper lacks a discussion of the source of utility scores, influence of other values, etc.

One of the most important models of consequentialist and virtue ethics is discussed in [6]. The model is constructed on the formal basis of the AATS+V model (presented in [10], [4], [5], and other). The authors use the Action-Based Alternating Transition System to model practical reasoning ([10], [4]). The key point of the model is the valuation function δ which assigns the status of value (promoted (+), demoted (-), neutral (=)) to a transition between states. Hence we may say that the valuation function describes a change of valuation, but not valuation in general, which in real life may be assigned both to a particular state of affairs or to a transition between states. The authors of the above paper return to discuss this issue in [5], where they introduce the logic program Δ which represents the influence of transition between states on the promotion or demotion of values from a value set.

The model of consequentialism presented in [6] replaces values with needs, based on the Maslov's hierarchy. The hierarchy of needs is the basis for the overall evaluation of the preference between consequences. Although there are some similarities between our model and AATS+V (see [25] for details), our approach to consequentialism is different. The main difference results from a different manner of evaluating states: in [6] (and other papers devoted to AATS+V) values can be promoted or demoted by a transition between states (the promotion or demotion of a value is relative w.r.t. a previous state), while in our model values are promoted by a state of affairs (or action), thanks to which we evaluate the absolute, not the relative level of promotion. On the basis of this, in our model, we can evaluate and compare (with the use of order OR) the consequences of decisions, without introducing additional, complex mechanisms (like the hierarchy of needs).

The work [6] also introduces a model of virtue ethics. The key point of this approach lies in the assumption that virtue is an ordering between values. The agent compares decision options in the light of ordered values: if two decisions promote different values, then the agent chooses the one which promotes values which s/he prefers. The main difference between our approach and the Bench-Capon's one consists in the different understanding of the concept of virtue: while in our model virtue is a set of thresholds of the levels of values' promotion, in the Bench-Capon's approach virtue is a hierarchy between values. In our opinion, representation of virtue as a set of thresholds is much more useful, because it prevents the autonomous device from decisions in which the strong promotion of one value, too strongly decreases the level of promotion of the other value.

Although the model of AATS+V allows for representing the concept of threshold ([5] distinguishes two specific kinds of human attitudes: 'maximizer' and 'satisfier,' which are represented by specific rules in Δ) and provides a possibility to point out that it is not necessary to promote some values above the particular level, the model does not protect the system from decreasing one value too strongly.

We focus on the knowledge-based approaches to making ethical decisions, but it is worth mentioning that there also exist some approaches to modeling consequentialist ethics with the use of machine learning mechanisms: in [1] the authors introduce the mechanism in which the system is learning the ethical decision making. The mechanism is constructed with the use of partially observable Markov decision processes and a reinforced learning mechanism. Since the training of the mechanism is based on the expected results of the decisions, it can be seen as the implementation of a kind of consequentialist ethics. A similar approach was also presented in [14], where the authors evaluate decisions in two dimensions: normative (punishment) and evaluative (reward). Since both of the above approaches are machine learning-based mechanisms, they work in a black-box style and they inherit the abovementioned disadvantages of ML-based systems.

6.3 The problem of uncertainty

The model presented in our paper has some limitations which can pose an interesting challenge for future research. One of the most important limitations is the assumption of the certainty of the action results. In real life, the results of a decision are, almost always, uncertain, and hence the levels of promotion of values by a given action also have to be uncertain.
How to deal with such a problem? In the classical decision theory, where the main aim of the agent is to maximize the so-called utility, the concept of Principle of Maximum Expected Utility (PMEU) was introduced [11]. In this approach, the levels of utility of the predicted results of every decision option are multiplied by the probabilities of their results.
Although our model does not feature the utility function but a number of levels of promotion of values, we can use a similar approach to model the uncertainty. We can calculate the expected level of promotion of values by multiplying the levels to which the potential results of an action promote values by their probabilities. However, this topic requires an in-depth analysis concerning, among others, the subjectivity of certainty evaluation. We plan to discuss this issue in future work.

7 Conclusions

The enormous development of autonomous devices which we have been witnessing for some time has triggered an extremely important discussion of the problem of ethical issues of the decisions made by such devices. This debate becomes even more important if we consider devices whose decisions can cause serious danger, like autonomous cars or military autonomous devices. As we have pointed out in the introduction, we are not interested in the development of *machine ethics*, but in the implementation of human ethics in machines. On the basis of such an assumption, we claim that a morally behaving autonomous device should be equipped with a mechanism which allows for making moral decisions. The creation of a formal model of such a mechanism constitutes the main aim of our work. The issues we have considered as crucial are: what we understand as moral behavior and how moral reasoning can be represented in a machine. We claim that an attempt to answer these questions requires an in-depth analysis of the moral philosophy theories, selection and specification of the theories which can be represented in an autonomous device, and introduction of the formal models of the chosen theories.

The main aim of our paper has been to introduce a formal and computational model of an ethical decision making mechanism. In order to fulfill our goal, we have presented the discussion of the main ethical theories and distinguished their specific variants which can be implemented in autonomous devices. We have presented the formal and computational model of these theories (with the formal background of the GVR model [25]) with examples illustrating the decision making mechanisms. We have also presented an in-depth discussion of our model in the light of relevant ethical theories and some existing models of ethical reasoning.

In future work we are going to focus on three research directions. Firstly, we intend to introduce a deeper discussion of the ethical aspects of our approach, in particular elaborating the concepts of value, virtue, and eudaimonia, as well as relations between them. The second research direction will concern the discussion of the problem of uncertainty of the results of an action. Another aspect of the research will include the development of a mechanism of determining the levels to which particular decision options promote particular values. This could possibly lead us to examine the possibility of developing a hybrid system including machine learning-based and knowledge-based aspects.

References

[1] David Abel, James MacGlashan, and Michael L. Littman. Reinforcement learning as a framework for ethical decision making. In Blai Bonet, Sven Koenig, Benjamin Kuipers, Illah R. Nourbakhsh, Stuart J. Russell, Moshe Y. Vardi, and Toby Walsh, editors, *AAAI Workshop: AI, Ethics, and Society*, volume WS-16-02 of *AAAI Workshops*. AAAI Press, 2016. 978-1-57735-759-9.

[2] Larry Alexander and Michael Moore. Deontological Ethics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2020 edition, 2020.

[3] G. E. M. Anscombe. Modern moral philosophy. *Philosophy*, 33(124):1–19, 1958.

[4] Katie Atkinson and Trevor Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171(10-15):855 – 874, 2007.

[5] Katie Atkinson and Trevor Bench-Capon. States, goals and values: Revisiting practical reasoning. In *Proceedings of 11th Intl. Workshop on Argumentation in Multi-Agent Systems*, 2014.

[6] T.J.M. Bench-Capon. Ethical approaches and autonomous systems. *Artificial Intelligence*, 281:103239, 2020.

[7] Jeremy Bentham. *An Introduction to the Principles of Morals and Legislation.* Dover Publications, 1780.

[8] Floris Bex, Henry Prakken, Chris Reed, and Douglas Walton. Towards a formal account of reasoning about evidence: Argumentation schemes and generalisations. *Artificial Intelligence and Law*, (11):125 – 165, 2004.

[9] Nicholas C. Burbules. Thoughts on phronesis. *Ethics and Education*, 14(2):126–137, 2019.

[10] Alison Chorley and Trevor Bench-Capon. An empirical investigation of reasoning with legal cases through theory construction and application. *Artificial Intelligence and Law*, 13(3-4):323–371, 2005.

[11] P. Fishburn. *Utility theory for decision making.* Wiley, 1970.

[12] Aditya Hegde, Vibhav Agarwal, and Shrisha Rao. Ethics, prosperity, and society: Moral evaluation using virtue ethics and utilitarianism. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 167–174. International Joint Conferences on Artificial Intelligence Organization, 7 2020.

[13] Rosalind Hursthouse and Glen Pettigrove. Virtue Ethics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2018 edition, 2018.

[14] Manel Rodriguez-Soto, Juan Antonio Rodriguez Aguilar, and Maite Lopez-Sanchez. Guaranteeing the learning of ethical behaviour through multi-objective reinforcement learning. https://ala2021.vub.ac.be/papers/ALA2021, 2021. ALA2021.

[15] S. Russell. *Human Compatible: Artificial Intelligence and the Problem of Control*. Penguin Publishing Group, 2019.

[16] Krzysztof Saja. *Etyka normatywna*. Universitas, 2015.

[17] Giovanni Sartor. Normative conflicts in legal reasoning. *Artif. Intell. Law*, 1(2–3):209–235, 1992.

[18] May Sim. *Remastering Morals with Aristotle and Confucius*. Cambridge University Press, 2007.

[19] Walter Sinnott-Armstrong. Consequentialism. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition, 2019.

[20] Christine Swanton. *The Virtue Ethics of Hume and Nietzsche*. Wiley-Blackwell, 2015.

[21] Suzanne Tolmeijer, Markus Kneer, Cristina Sarasua, Markus Christen, and Abraham Bernstein. Implementations in machine ethics: A survey. *ACM Comput. Surv.*, 53(6), 2021.

[22] Ozlem Ulgen. Kantian ethics in the age of artificial intelligence and robotics. *Questions of International Law*, 43:59–83, 2017.

[23] Chien Van Dang, Tin Trung Tran, Ki-Jong Gil, Yong-Bin Shin, Jae-Won Choi, Geon-Soo Park, and Jong-Wook Kim. Application of soar cognitive agent based on utilitarian ethics theory for home service robots. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 155–158, 2017.

[24] Jiyuan Yu. *The Ethics of Confucius and Aristotle: Mirrors of Virtue*. Routledge, 2007.

[25] Tomasz Zurek. Goals, values, and reasoning. *Expert Systems with Applications*, 71:442 – 456, 2017.

[26] Tomasz Zurek and Michail Mokkas. Value-based reasoning in autonomous agents. *International Journal of Computational Intelligence Systems*, 14:896–921, 2021.

Assessment of Attractiveness and Trust in Relation to Personality Traits — Literature Review and Research Proposal

Bernadetta Bartosik*
Grzegorz Marcin Wójcik

1 Introduction

Faces are an inseparable element of every person's everyday life. As we walk down the street, we pass many people we may or may not know. Welcome each other with a nod of the head or a smile. Young people are taking selfies more and more often and sharing them on social media. Facial recognition is part of everyday life and is developed from the first moments of life [4], [23], [28]. In order to obtain multidimensional information about a person, even a brief look at the face is enough. The success of social interactions largely depends on how we perceive the person. One can read a lot of information from the face, ranging from age, gender, emotions, health condition to the assessment of attractiveness, trust, dominance [42], [25], [30] ,[24]. On their basis, people often make decisions or trust [8], [37] .

In general, trust can be described as a voluntary action, which in its consequences may be associated with a favorable or unfavorable result of the conduct of others [41]. Information read from the face is involved in the confidence analysis process. Attractiveness, as one of many arguments, has a great influence on the decision. One may suspect that the assessment of attractiveness depends on general, more external signs, while credibility is multifaceted and also applies to internal personality traits. It is therefore highly likely that in the event of insufficient information, credibility will be inferred from the attractiveness of a face, which we are always able to determine. Research shows that attractive people will be more often assessed as trustworthy [33], have more positive personality traits [13], [16], are better treated by others, have higher salaries [26]. Female faces are more attractive than male faces [32]. It has been proven that the emotional expression of the face also has a very significant influence on the assessment of trust. People who smile and display positive emotions will be considered more credible [10], [35]. Trust in

*Corresponding author — bernadetta.bartosik@mail.umcs.pl others has a great impact on the functioning of various types of Internet activity. Sellers with a credible face have a greater number of orders / reservations [17]. This topic is very well presented in research based on the "trust game", where the participant can invest money at a profit or loss. It was noticed that a large number of participants spend more on offers / partners whose face is trustworthy [12], [38]. Moreover, the assessment of trust has a particular impact on starting cooperation with someone [42]. It has been proven that criminals whose face looks credible are more likely to receive lower penalties [2], [40].

Facial recognition and perception are the result of interactions between many neural processes responsible for facial perception [9], [20]. Looking at the face of another person, we can immediately deduce the basic features and make the right decisions. This is especially important in life-threatening situations. The characteristics related to gender and age are recognized the earliest, while the brain then processes identity [14], [15]. The earliest ERP element that appears after face exposure is the N170 potential [7], [21] (Jeffreys, 1996). It is related to the structural encoding of the face in the temporo-occipital regions. The increase in the amplitude of this potential was recorded for inverted faces [22]. It can also be registered by processing social information relating to social categorization and racial prejudice [3], [29]. As already mentioned, identity is processed by the brain later and is recorded within the limits of the N250 potential [39]. In various ways, about 300–600 ms after the stimulus presentation, there are signals related to cognitive processing [11].

In assessing the credibility of other people, great importance is attached to the amygdala, which is associated with the processing of lower-level emotions, especially negative ones that may interact with other structures in order to quickly analyze threats [31], [36]. The amygdala may be activated less in the case of credible-looking faces, or more in the opposite case [19]. Damage to the amygdala causes disturbances in the correct assessment of trust, which means that people with this dysfunction are more likely to positively assess people who are untrustworthy [1].

The face is most often judged by people and it is on its basis that the first judgments can be made. Therefore, a future study is presented below, the aim of which will be to find the relationship between the assessment of attractiveness and the assessment of trust of the real faces shown in the photos and the personality trait of the person making the assessment of these features. No two people are alike in the world. Despite the external similarity, people differ in their behavior, way of thinking or the way of receiving and experiencing emotions. Personality traits are relatively constant properties that can be measured. In the planned study, these features will be determined by psychological tests for each subject and compared with the ratings issued for attractiveness and trust in the presented photos of the face. It is presumed that people who have similar personality traits will give similar assessments, e.g. neurotic people will be less trusting.

2 Tools

The experiment will be developed on the basis of photos of female and male faces. Due to the fact that artificial faces have an impact on the absolute levels of perceived credibility, it was decided to use databases containing photos of real people [5]. The photos were downloaded from online photo databases that make their resources available for research purposes. The following criteria were used when selecting the bases: it should contain photos of men's and women's faces, people in the photos should be of different ages and races, faces should be presented from the front, faces should not show emotions. Develpoment Emotional Faces Stimulus Set (DEFSS) and Mulit-Racial Mega-Resolution (MR2) were selected from among many databases. These are databases characterized by great care of the photos taken and good resolution. DEFSS is a collection of 404 photos of people between the ages of 8 and 30. It is a base showing faces expressing emotions, including happiness, sadness, anger, and lack of emotions. The photos are verified by the respondents in terms of the emotions presented [27]. MR2 is a collection that includes photos of 74 people of different races between 18 and 25 years of age. The faces of the models do not convey any emotions. The collection was verified by the respondents assessing, among others, age, sex, race, attractiveness, and trust [34].

From the above-mentioned databases, 100 photos were selected, 50 of which show female faces and 50 are male faces. All photos show faces with neutral expressions of different origins (49 — European, 31 — African, 20 — East Asian) shown from the front. Research focuses on assessing attractiveness and trust. Given that not all photos have verified these characteristics, a survey was conducted in which 85 UMCS computer science and cognitive science students assessed it. For each photo, three questions were displayed: "How attractive is the person in the photo?", "How can you trust the person in the photo?", "What is the gender of the person in the photo?" The first two questions were answered using a five-point Likert scale, where 1 means not at all and 5 means very much. The responses were statistically analyzed and four groups of photos were distinguished: attractive and trustworthy, unattractive and untrustworthy, attractive and untrustworthy, unattractive and trustworthy. Some of these groups were more numerous than others. For this reason, 6 photos with the highest scores by students were selected from each group (Figure 1). The resulting set will be used to design the EEG experiment.

Personality traits will be determined through psychological tests. Two tests will be used, which are the NEO PI-R Personality Inventory and the IVE Impulsivity Questionnaire. The first one is one of the most frequently used for research. It was created by McCrae and Costa in the nineties, and translated into Polish by Siuta. This test is based on the five-factor model of personality (Table 1).

Initially, it consisted of three factors and each of them had its own subscale. Currently, the personality inventory consists of five factors with their own subscales (Table 2), which constitute the most general dimensions for determining personality characteristics. The test that will be used contains 240 questions that must be answered on a five-point scale from 0 to 4 depending on how true the question is in relation to the participant's feelings.

The second test is the Impulsiveness Questionnaire created by Hans J. Eysenck and Sybil B. G. Eysenck. It consists of 54 questions to which the respondent may answer "yes" if he agrees with the statement or "no" if he does not agree with it. Thanks to IVE, dimensions such as impulsivity (determining the pathological aspect of risky behaviors), empathy (characteristic of people who are not indifferent to the emotions of others), a tendency to take risks.

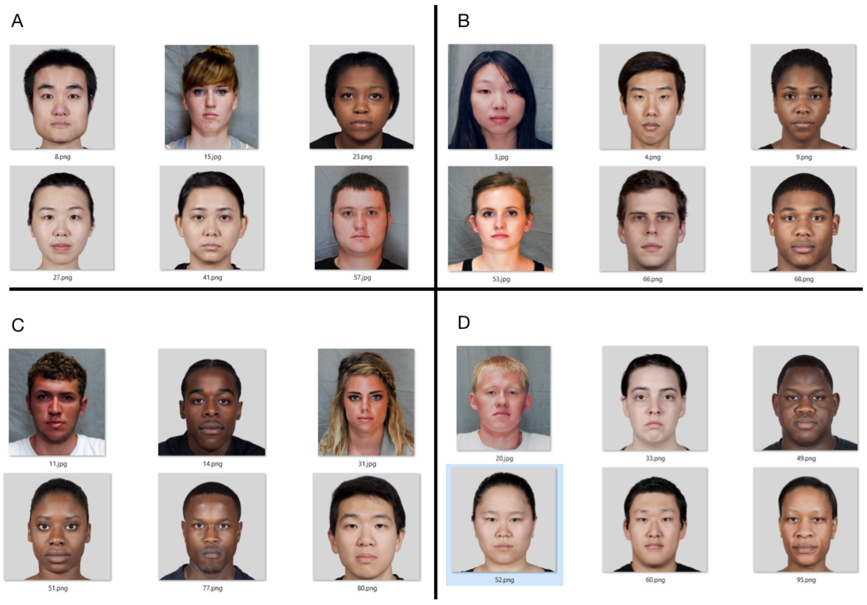


Figure 1: Collection of 24 photos divided into groups: A — unattractive and trustworthy, B — attractive and trustworthy, C — attractive and untrustworthy, D — unattractive and untrustworthy. Photos taken from the following databases: DEFSS [27] and MR2 [34]

3 Procedure

The study will be carried out in the laboratory of the Department of Neuroinformatics and Biomedical Engineering at the Maria Curie-Skłodowska University using an amplifier recording cortical activity through 256 channels (HydroCel GSN 130, EGI) with a frequency of up to 500 Hz. The study will be designed to obtain psychological data from participants and the EEG signal. For this purpose, the whole thing has been divided into two stages. In the first, participants will be asked to remotely complete two psychological questionnaires. The NEO-Pi-R test contains 240 questions that should be answered on a scale of 0 to 4, where 0 means that you do not agree with the statement, and 4 means that you agree with it. In the IVE test, the participant will answer 54 questions, in which he will choose one of the two possible answers "yes" or "no". The second stage will be held stationary. Participants will take part in an experiment in which they will assess attractiveness and confidence, and make a gender selection. On the screen where the participants will be sitting, there will be photos with faces of men and women of different ages and races. These photos were selected from a larger group of photos. Each respondent will assess whether the person in the photo is:

- attractive/unattractive by answering the question "How attractive is the person in the photo?". The answer to this question will be on a scale from 1 to 5, where 1 means not at all and 5 very much,

- trustworthy/not trustworthy by answering the question “To what extent are you able to trust the person in the photo?” In this case, the participant will make a choice using the same scale as for the attractiveness assessment,
- a woman/man answering the question “What gender is the person in the photo?”.

Each time the attempt will start with a black screen displayed for 300 ms, followed by a fixation point with a variable display time, randomly selected in the 100-1200 ms range. After the fixation point disappears, the subject will see a screen with an appropriate question, and after 1200 ms a photo of the face will be displayed, which will disappear only after answering. The maximum time for displaying the photo screen is 20000 ms (Figure 2). Photo stimuli will appear in random order.

After the end of the experiment, each participant will be photographed using the Geodesic Photogeammetry System (GPS). The station has 11 cameras in each corner of the station and each photo shows a different fragment of the cap. Thanks to this, it is known where the individual electrodes are, and using the appropriate software, you can create a model of the brain.

Table 1: Description of the dimensions of the big five

Personality factors	Description
Neuroticism	Described by traits such as anger, fear, guilt. It is the reverse of emotional stability. Neurotic people often deal with stress, they control their behavior less often, they do not feel well in company.
Extroversion	Extroverted people are open, cordial, friendly, energetic, willing to make new friends and look for new experiences. The opposite is introversion.
Openness	It characterizes people looking for new life experiences, tolerant and curious about the world, new tasks are not a problem for them and they are willing to undertake them. People with low openness to experiences are characterized by, among other things, conservativeness and conventionality.
Agreeableness	It is presented as a positive attitude towards people. Agreeable people are characterized by honesty, trust, and a willingness to bring disinterested help. The opposite is self-centeredness.
Conscientiousness	It is the attitude towards goal-oriented action. Conscientious people are usually persistent in pursuing their goals, meticulous, reliable, and tend to be perfectionist and workaholic.

4 Summary

The above work presents the literature in which it is noticed that judgments of the degree of trust in the face are strongly correlated with attractiveness. It is not without reason that the saying “When they see you, they write you like that”. From the face you can read a lot of important information about gender, age and intentions. Some of this information is processed very quickly, thanks to which the person is able to quickly make the right decisions. According to research, the first signals may appear even around 33 ms after the stimulus has occurred [6] [18]. The amygdala plays a big part in the credibility assessment process, as its damage is associated with an incorrect assessment of the credibility of the face. In the proposed study, the faces presented to participants will be subject to an assessment of attractiveness and degree of trust. Apart from the facial assessment, the participants will

Table 2: NEO PI-R Personality Inventory. Division into subscales

Type of test	Personality factors	Component factors
Neo-Pi-R	Neuroticism	Anxiety
		Angry hostility
		Depression
		Self-consciousness
		Impulsiveness
		Vulnerability
	Extroversion	Warmth
		Gregariousness
		Assertiveness
		Activity
		Excitement seeking
		Positive emotions
	Openness	Fantasy
		Aesthetics
		Feelings
		Actions
		Ideas
		Values
	Agreeableness	Trust
		Straightforwardness
		Altruism
		Compliance
		Modesty
		Tendermindedness
	Conscientiousness	Competence
		Order
		Dutifulness
		Achievement striving
		Self-discipline
		Deliberation

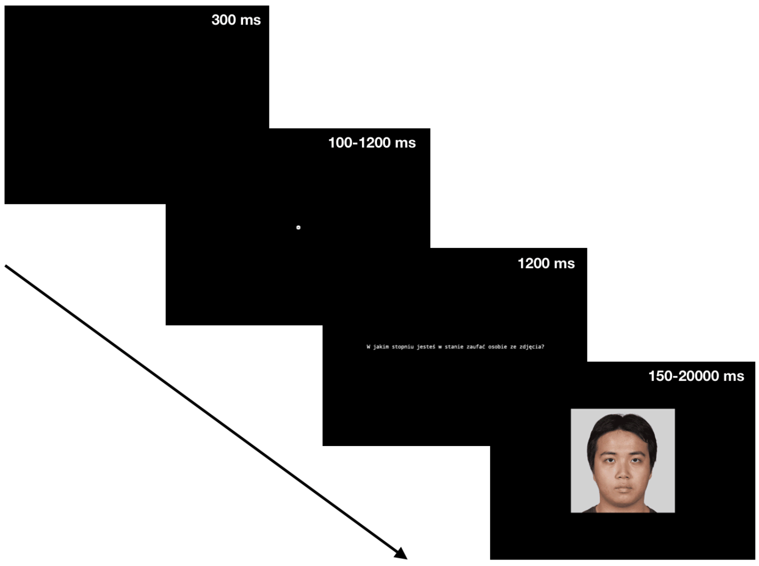


Figure 2: Diagram of a single experiment sample

complete psychological tests that will allow to determine their personality traits. It is planned to evaluate the course of ERP during the assessments and to identify the most active brain areas. Based on the analyzed signal and the individual differences of the respondents, an attempt will be made to determine the relationship between them.

References

[1] Adolphs, R., Tranel, D., & Damasio, A. R. (1998). The human amygdala in social judgment. Nature, 393(6684), 470–474.

[2] Ancāns, K., & Austers, I. (2018). The Influence of Face Trustworthiness on Judgments in Forensic Context. Baltic Journal of Psychology, 19.

[3] Amodio, D. M., Bartholow, B. D., & Ito, T. A. (2014). Tracking the dynamics of the social brain: ERP approaches for social cognitive and affective neuroscience. Social Cognitive and Affective Neuroscience, 9(3), 385-393.

[4] Arcaro, M. J., Schade, P. F., Vincent, J. L., Ponce, C. R., & Livingstone, M. S. (2017). Seeing faces is necessary for face-domain formation. Nature neuroscience, 20(10), 1404.

[5] Balas, B., & Pacella, J. (2017). Trustworthiness perception is disrupted in artificial faces. Computers in Human Behavior, 77, 240–248.

[6] Bar, M., Neta, M., & Linz, H. (2006). Very first impressions. Emotion, 6(2), 269.

[7] Bentin, S., Allison, T., Puce, A., Perez, E., & McCarthy, G. (1996). Electrophysiological studies of face perception in humans. Journal of cognitive neuroscience, 8(6), 551–565.

[8] Bonnefon, J. F., Hopfensitz, A., & De Neys, W. (2017). Can we detect cooperators by looking at their face?. Current Directions in Psychological Science, 26(3), 276–281

[9] Bruce, V., & Young, A. (1986). Understanding face recognition. British journal of psychology, 77(3), 305–327

[10] Calvo, M. G., Álvarez-Plaza, P., & Fernández-Martín, A. (2017). The contribution of facial regions to judgements of happiness and trustworthiness from dynamic expressions. Journal of Cognitive Psychology, 29(5), 618–625.

[11] Calvo, M. G., Gutiérrez-García, A., & Beltrán, D. (2018). Neural time course and brain sources of facial attractiveness vs. trustworthiness judgment. Cognitive, Affective, & Behavioral Neuroscience, 18(6), 1233–1247.

[12] Chang, L. J., Doll, B. B., van't Wout, M., Frank, M. J., & Sanfey, A. G. (2010). Seeing is believing: Trustworthiness as a dynamic belief. Cognitive psychology, 61(2), 87–105.

[13] Dion, K., Berscheid, E., & Walster, E. (1972). What is beautiful is good. Journal of personality and social psychology, 24(3), 285.

[14] di Oleggio Castello, M. V., & Gobbini, M. I. (2015). Familiar face detection in 180ms. PLoS One, 10(8).

[15] Dobs, K., Isik, L., Pantazis, D., & Kanwisher, N. (2019). How face perception unfolds over time. Nature communications, 10(1), 1–10.

[16] Eagly, A. H., Ashmore, R. D., Makhijani, M. G., & Longo, L. C. (1991). What is beautiful is good, but…: A meta-analytic review of research on the physical attractiveness stereotype. Psychological bulletin, 110(1), 109.

[17] Ert, E., Fleischer, A., & Magen, N. (2016). Trust and reputation in the sharing economy: The role of personal photos in Airbnb. Tourism Management, 55, 62–73.

[18] Freeman, J. B., Stolier, R. M., Ingbretsen, Z. A., & Hehman, E. A. (2014). Amygdala responsivity to high-level social information from unseen faces. Journal of Neuroscience, 34(32), 10573–10581.

[19] Haas, B. W., Ishak, A., Anderson, I. W., & Filkowski, M. M. (2015). The tendency to trust is reflected in human brain structure. Neuroimage, 107, 175–181.

[20] Haxby, J. V., Hoffman, E. A., & Gobbini, M. I. (2000). The distributed human neural system for face perception. Trends in cognitive sciences, 4(6), 223–233.

[21] Itier, R. J., & Taylor, M. J. (2004). N170 or N1? Spatiotemporal differences between object and face processing using ERPs. Cerebral cortex, 14(2), 132–142

[22] Itier, R. J., & Taylor, M. J. (2002). Inversion and contrast polarity reversal affect both encoding and recognition processes of unfamiliar faces: a repetition study using ERPs. Neuroimage, 15(2), 353–372.

[23] Jessen, S., & Grossmann, T. (2019). Neural evidence for the subliminal processing of facial trustworthiness in infancy. Neuropsychologia, 126, 46–53.

[24] Jones, B. C., Little, A. C., Penton-Voak, I. S., Tiddeman, B. P., Burt, D. M., & Perrett, D. I. (2001). Facial symmetry and judgements of apparent health: Support for a "good genes" explanation of the attractiveness–symmetry relationship. Evolution and human behavior, 22(6), 417–429.

[25] Kościński, K. (2007). Facial attractiveness: General patterns of facial preferences. Anthropological Review, 70(1), 45–79.

[26] Langlois, J. H., Kalakanis, L., Rubenstein, A. J., Larson, A., Hallam, M., & Smoot, M. (2000). Maxims or myths of beauty? A meta-analytic and theoretical review. Psychological bulletin, 126(3), 390.

[27] Meuwissen, A. S., Anderson, J. E., & Zelazo, P. D. (2017). The creation and validation of the developmental Emotional Faces Stimulus Set. Behavior research methods, 49(3), 960–966

[28] Mondloch, C. J., Gerada, A., Proietti, V., & Nelson, N. L. (2019). The influence of subtle facial expressions on children's first impressions of trustworthiness and dominance is not adult-like. Journal of experimental child psychology, 180, 19–38.

[29] Ofan, R. H., Rubin, N., & Amodio, D. M. (2011). Seeing race: N170 responses to race and their relation to automatic racial attitudes and controlled processing. Journal of Cognitive Neuroscience, 23(10), 3153–3161.

[30] Oosterhof, N. N., & Todorov, A. (2008). The functional basis of face evaluation. Proceedings of the National Academy of Sciences, 105(32), 11087–11092.

[31] Pessoa, L., & Adolphs, R. (2010). Emotion processing and the amygdala: from a'low road'to'many roads' of evaluating biological significance. Nature reviews neuroscience, 11(11), 773–782.

[32] Rhodes, G., Chan, J., Zebrowitz, L. A., & Simmons, L. W. (2003). Does sexual dimorphism in human faces signal health?. Proceedings of the Royal Society of London. Series B: Biological Sciences, 270(suppl_1), S93–S95

[33] Shinners, E. (2009). Effects of the "what is beautiful is good" stereotype on perceived trustworthiness. UW-L Journal of Undergraduate Research, 12, 1–5.

[34] Strohminger, N., Gray, K., Chituc, V., Heffner, J., Schein, C., & Heagins, T. B. (2016). The MR2: A multi-racial, mega-resolution database of facial stimuli. Behavior research methods, 48(3), 1197–1204.

[35] Sutherland, C. A., Young, A. W., & Rhodes, G. (2017). Facial first impressions from another angle: How social judgements are influenced by changeable and invariant facial properties. British Journal of Psychology, 108(2), 397–415.

[36] Tamietto, M., & De Gelder, B. (2010). Neural bases of the non-conscious perception of emotional signals. Nature Reviews Neuroscience, 11(10), 697–709.

[37] Todorov, A., Olivola, C. Y., Dotsch, R., & Mende-Siedlecki, P. (2015). Social attributions from faces: Determinants, consequences, accuracy, and functional significance. Annual review of psychology, 66, 519–545.

[38] Van't Wout, M., & Sanfey, A. G. (2008). Friend or foe: The effect of implicit trustworthiness judgments in social decision-making. Cognition, 108(3), 796–803.

[39] Werheid, K., Schacht, A., & Sommer, W. (2007). Facial attractiveness modulates early and late event-related brain potentials. Biological psychology, 76(1–2), 100–108

[40] Wilson, J. P., & Rule, N. O. (2015). Facial trustworthiness predicts extreme criminal-sentencing outcomes. Psychological science, 26(8), 1325–1331.

[41] Yamagishi, T., Kanazawa, S., Mashima, R., & Terai, S. (2005). Separating trust from cooperation in a dynamic relationship: prisoner's dilemma with variable dependence. Rationality and society, 17(3), 275–308.

[42] Zebrowitz, L. A., & Montepare, J. M. (2008). Social psychological face perception: Why appearance matters. Social and personality psychology compass, 2(3), 1497–1517.

Liquid State Machines for Real-Time Neural Simulations

Karol Chlasta*
Grzegorz Marcin Wójcik

1 Introduction

Our brain consists of approximately a hundred billion neurons. The data provided by different authors have led to a broad range of 75–125 billion neurons in the whole brain [15]. Neurons are organised in microcircuits, also known as neural columns. They differ by purpose in the human brain [8].

Excitable media, as introduced by [11] in the theory of synchronous concurrent algorithms, provide a great framework for computations. A new approach to microcircuit computing was suggested by Maass [16]. In Maass's Liquid State Machine (LSM) theory, the brain, or its fragments, are treated as a liquid. The model provides an alternative to the Turing machine [19]. Moreover, a mathematical analysis shows that there are in principle no computational limitations of liquid state machines [16].

Neural networks are a great tool for modelling different systems, including complex biological or technological systems. Artificial neural networks facilitate biomedical signal processing, biomedical data analysis and interpretation, as well as models for the analysis of system behaviour, including the prognosis of results of selected activities [20].

Several successful applications of the LSM framework have been delivered in the area of artificial neural networks, or to solve engineering tasks such as the design of nonlinear controllers [18]. Moreover, we know that cortical microcircuits seem to be very useful for computing on perturbations [24].

Direct observation of neuronal activity of individual neurons is not possible given the current state of art in human neuroimaging [1]. To achieve brain-scale simulations and to investigate emergent properties of brain circuits we need better building blocks to simulate the whole neural circuits with higher fidelity.

There has been an increased development in the performance and capability of neural simulations in the past years. As a result the simulator and supercomputer technology have now developed to the point that the actual process of setting up and simulating a realistic neural model is now practical [9].

*Corresponding author — karol@chlasta.pl

In spite of that some key challenges remain. One of them is that the number of synaptic connections in the neural models offering higher fidelity have shown to exceed the current memory capacity of hardware that is available to researchers [14]. The other major challenge to the simulation of neuronal networks is handling both the computations and data generated by the large number of synaptic inputs to a single neuron, and scaling these neurons to the larger structures [13].

This paper presents a simulation framework for spiking neural network simulation and provides a simple, extensible retina-LGN-cortex model. We also leverage a novel distributed simulation setup, with in-memory processing of the readout signal. We also extend the prior work of [23], in building a higher fidelity bio-inspired visual system resembling mammalian visual cortex. The new model has a more complex retina, allowing to process input patterns generated by viewing a resolution of a MINST dataset [6] in near real-time. We achieve simulation results from four LSM columns, and calculate the Euclidean distance of states in each of the four columns of the system to illustrate the differences in spiking patterns.

We present a model that helps in understanding the signal processing phenomena within each column, and is extensible for further research on signal processing in multiple hypercolumns of the brain.

1.1 Liquid State Machines

In contrast to common computational models, Maass's Liquid State Machine introduced in 2002 [16] does not require information to be stored in stable states of a computational system. In the same year a Liquid Computer term was coined as a novel strategy for real-time computing on time series [18]. Such a computer consists of the four main components:

- Input $i(\cdot)$, a continuous input stream.
- Liquid column, or a filter L^N that maps the input into function $Y^N(t)$:

$$Y^N(t) = (L^N i)(t). \tag{1}$$

- Memory-less readout map m^N.
- Output $o(t)$:

$$o(t) = m^N(Y^N(t)). \tag{2}$$

In Maass's LSM architecture described in [16] the function of time series $i(\cdot)$ is injected as input into the liquid column L^N, creating at time t the liquid state $Y^N(t)$, which is transformed by a memory-less readout map m^N to generate an output $o(t)$.

Maass introduced two macroscopic properties of LSM, a Separation Property (SP) and an Approximation Property (AP), that can provide improvements with respect to classification applications. SP measures the dispersion between projected liquid states from different classes, whereas the AP indicates the concentration of the liquid states that belong to the same class.

The architecture is under active development. Recently [22] proposed to use a group of locally connected LSM reservoirs to form an ensemble of liquids. This approach could simulate higher levels of connectivity in LSMs that are in close proximity, and lower levels of connectivity for these that are spatially further apart.

1.2 Real-time processing

We propose to combine the LSM architecture with Apache Spark [27]. Apache Spark can be used as a distributed, fault tolerant data processing engine for extracting insights at scale with near-real time speeds [28]. It is an open source computing framework that unifies streaming, batch, and interactive big data workloads. We use it to improve the analysis of outputs generated by neural columns simulated with GENESIS [3] programming framework.

The fundamental building block of Spark architecture is Resilient Distributed Dataset (RDD). RDDs are in-memory objects on which all the operations on Spark platform are performed, and it happens in a distributed way [26].

An RDD is a collection of entities, similar to rows or records. RDD functionality makes the distributed processing possible by allowing to split the data it contains across all the data nodes of a Spark cluster. RDDs are immutable; once created they cannot be edited, updated, or appended to. They are considered resilient because they tolerate node failures within the cluster, and can be reconstructed in case of a node failure [26, 29].

From the programming perspective they are similar to Java collections, but under the Spark layer they are partitioned and distributed across multiple computing nodes. Unlike in the case of Hadoop and HDFS, Spark RDDs represent the data in-memory for each of the machines in the cluster. The distribution of the data across the computational nodes allows Spark to process the data in parallel. Several processes can be run on an individual subset of data by a cluster [29].

RDDs can be mutated (be appended or changed). There are only two operations that are permitted on an RDD: [28]:

- Transformation (resulting in creation of a new RDD, with all the edits that are needed).
- Action (or request for a result, which causes Spark to execute a set of transformations defined for a dataset).

Spark follows lazy evaluation of operations by keeping a record of the series of transformations requested by the user on the dataset. It groups the transformation in an efficient way when an action is requested. This allows an RDD to be reconstructed even if the node it lives on crashes. RDD can be created when a file is read, or a transformation of another RDD is called. Each RDD keeps metadata in which it keeps track of where it came from. This feature is called the RDD lineage, and it allows an RDD to reconstruct itself through re-performing all the recorded transformations [26].

Since its beginnings, the platform offers machine learning libraries. Spark 1.x provides support for ML with *spark.mllib*. Spark 2.x, the current version works with *spark.ml* and it offers an entirely new set of APIs for developers, which can work with data frames, and not directly with Resilient Distributed Datasets. The other advantage of the platform for machine learning is that Spark offers libraries for hyper-parameter tuning, allowing to choose the best model for a given use case [17]. Moreover, the recent developments in the Spark project increased the execution speed on the platform between 10 and 100 times [29].

2 Materials and methods

2.1 Signal processing model

We propose a bio-inspired model of a visual system that consists of two main modules. Our approach is in line with the LSM architecture proposed by Maass [16]. There are two main components of our model:

1. Input (Retina, as presented in Fig. 1).
2. Liquid (Cortex, built of four identical LSM columns, each as presented in Fig. 2).

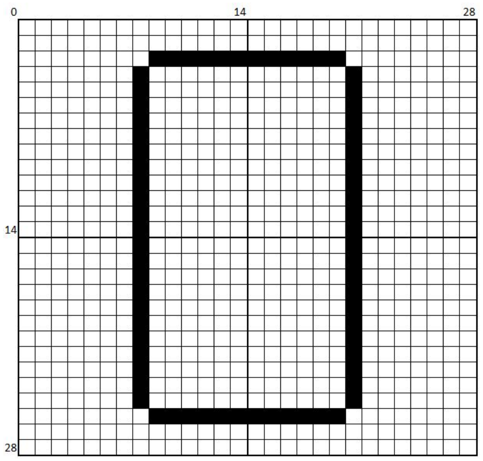


Figure 1: Retina of the proposed visual system with the stimulating patterns (Input)

All the simulations discussed in this paper were programmed using GEneral NEural SImulation System (GENESIS) [4]. All the neurons used in the simulations were built according to the Hodgkin-Huxley model [10].

Similar to the old model presented in [23], our new Hodgkin-Huxley Liquid State Machine (HHLSM) model uses the same high fidelity multi-compartmental neurons. The soma of each neuron uses biologically similar voltage-activated sodium and potassium channels. We build on a well known conductance-based model describing how action potentials in neurons are initiated and propagated in electrical circuit [25]. The GENESIS parameters we used in our simulations can be organised into four groups:

- Main resistances $R_x = 0.3\ \Omega$, $R_n = 0.33\ \Omega$.

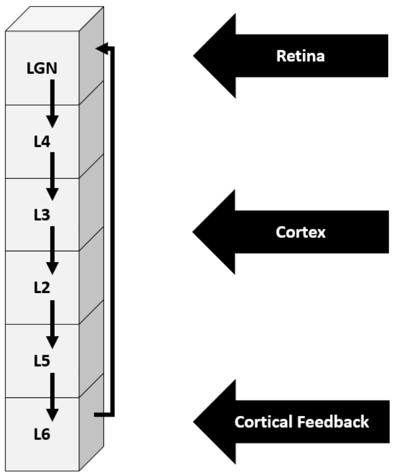


Figure 2: Structure of the LSM column, a fundamental computational microcircuit of our model (Liquid)

- Capacitance $C_n = 0.01$ F and potential incl. $E_n = 0.07$ V, $E_k = 0.0594$ V (soma compartment), $E_k = 0.07$ V (dendrite).
- Conductance $G_K = 360\ \Omega^{-1}$ and $G_{Na} = 1200\ \Omega^{-1}$ (for each of the ionic channels).
- Physical characteristics, a soma with circular shape and diameter of 30 μ, dendrites and axon length of 100 μ.

The exact values for these parameters were achieved experimentally by [25] to provide a high biological fidelity of the model. A detailed description of the Hodgkin-Huxley model can be found in [10].

The Retina was built on a 28×28 square-shaped grid and divided into four patches (2×2). Each patch is connected to one of the four HHLSM columns which simulate Lateral Geniculate Nuclei (LGN), and later the ensemble of cortical microcircuits. The retinal cells are only connected to the LSM column through the LGN layer. Each HHLSM consists of 1024 neural cells placed in a cuboid of $8 \times 8 \times 16$. The structure of each column in the models is the same (Fig. 2). The model contains four columns in total that form the Liquid stimulated by Retina.

There are 90% of excitatory connections established among layers and neurons of each layer and 10% of inhibitory connections. Additionally, Layers L6 of LSM columns are connected with LGNs of other HHLSMs in the same way (i.e. with the probability of 10%), simulating the corticothalamic feedback. For simplicity, we did not implement the pathway of the possible inter-column connections. Each connection in the model is characterised with a delay parameter and random weight. We can treat the Liquid as a single hyper-column made of four independent neural columns, a part of periodic structure of the simulated cortex.

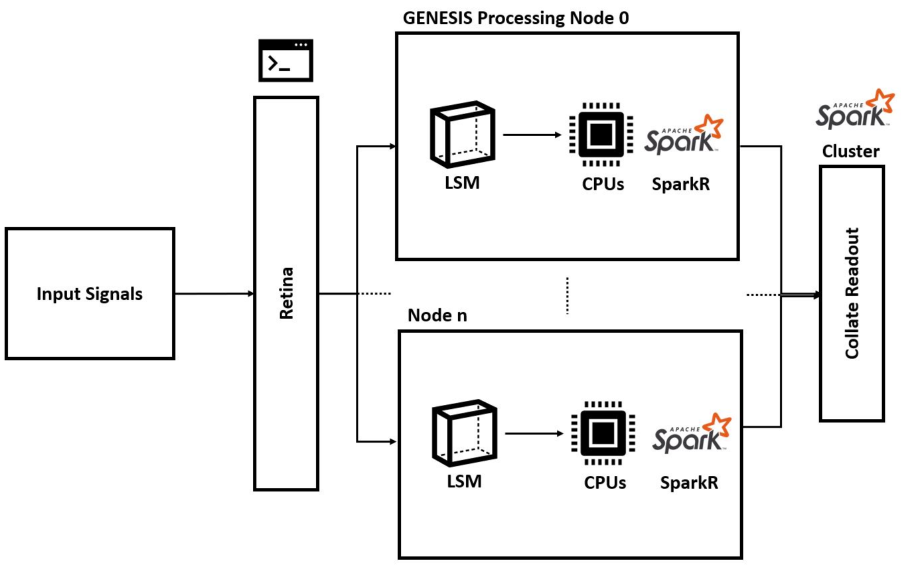


Figure 3: High level simulation setup including GENESIS and Spark components of the framework

2.2 Experimental setting

We performed all the simulations described in this paper using a simple Spark cluster with a single master node, and two worker nodes. The system was built around computing resources available on a free Google Colaboratory Cloud platform [2]. The machine was equipped with the Intel Xeon processing units running at 2.30 GHz, using 256KB L2 Cache in a single computing component, and having 2 cores per chip. The 64-bit system had 12 GB of RAM, and worked under the control of Linux (Ubuntu 18.04.5 LTS) with the kernel version of 4.19.112+. We installed the latest Spark-3.0.2 with hadoop 2.7 using OpenJDK Runtime Environment 11.0.10. The model was implemented in GEneral NEural SImulation System GENESIS v.2.4[1]. Both results and source code for the simulations are available on GitHub repository[2]. Fig. 3 presents a diagram with simulation setup, including GENESIS and Spark components, along-site their relative roles in the overall approach.

3 Results

We created a system consisting of 784 retina cells and a Liquid State Machine organised into 4 computational columns, 1024 cells each. In contrast to the original Maass's LSM using integrate and fire neurons, our architecture is built of a more biologically realistic model of neural cells proposed by Hodgkin-Huxley.

[1]GENESIS v.2.4 https://github.com/dbeeman/genesis-2.4beta-files.
[2]LiquidComputer https://github.com/KarolChlasta/LiquidComputer.

In total our system was built using 4880 artificial neurons. They simulated a simple visual system, processing input signals through a square-shaped grid of 28x28 pixels. We stimulated the retinal cells with a pattern resembling the digit of 0. The input signal (Fig. 1) was encoded in the Liquid state, and a unique pattern of spikes was observed in each of four LSM columns, as visualised in Fig. 6 and 7, and measured in Table 1.

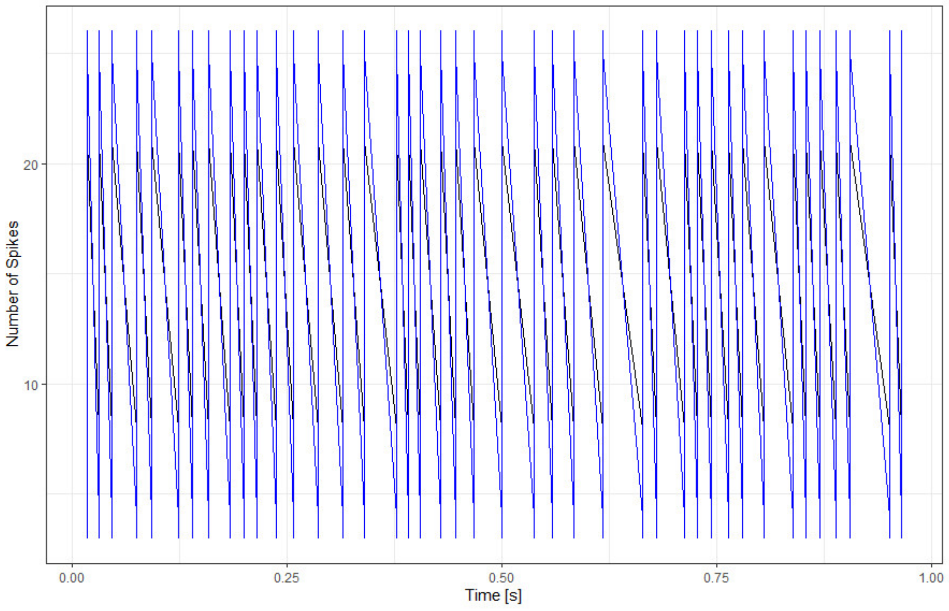


Figure 4: Number of spikes generated by Retina (Input) stimulation for a 2D (28x28) setup within the simulation time of 1 second

Fig. 4 and 5 present two views of the spiking response of our retina grid. Retinal cells chosen for each pattern were stimulated with random spike trains with the average rate of 200 Hz. The spikes marked in black were triggered by input signals simulating the pattern of 'shorter part' of the 0 shape ($2 * 12$ stimulating spike trains), whereas the spikes marked in blue were generated by the 'longer parts' of the 0 shape stimulating our retina model ($2 * 22$ stimulating signals). This is important to observe, because it shows that our retina response generates expected signals when stimulated, and these signals can be passed to the subsequent parts of the system, as presented in Fig. 2.

Fig. 6 and 7 present cumulative spiking activity for each of the LSM columns. We performed a 2D kernel density estimation to visualise the structure of cumulative spikes with contour lines. We observe different dynamics of these spikes generated within each LSM column. Such behaviour is typical for LSMs, and can be measured for each column, using their vectors of states.

Finally, Table 1 presents the matrix of euclidean distances between four vectors of states of each LSM column during the time of experiment. This different pattern is an important feature for the downstream processing of visual information be-

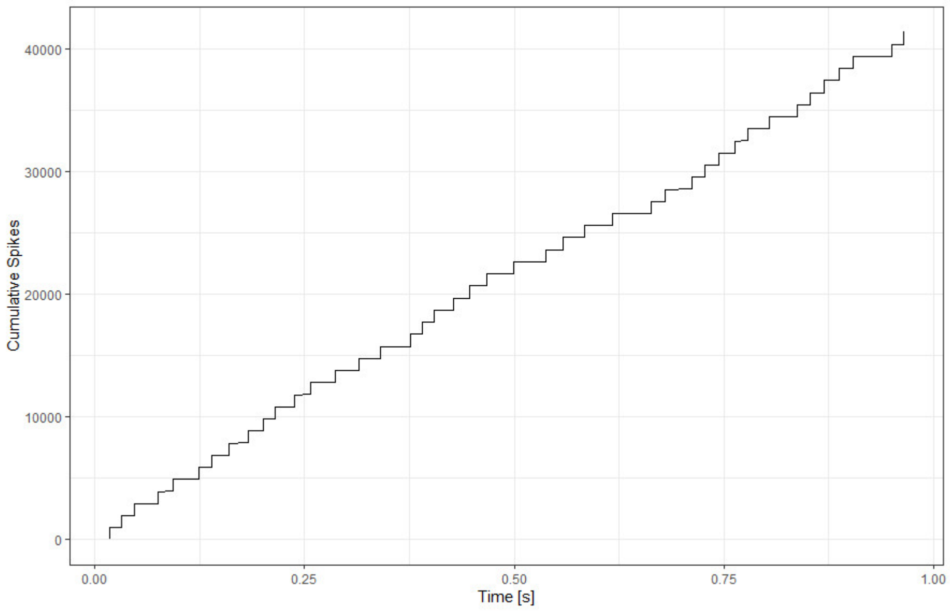


Figure 5: Cumulative number of Retina spikes in each ms of simulation reaching 40000 before the end of simulation time of 1 second

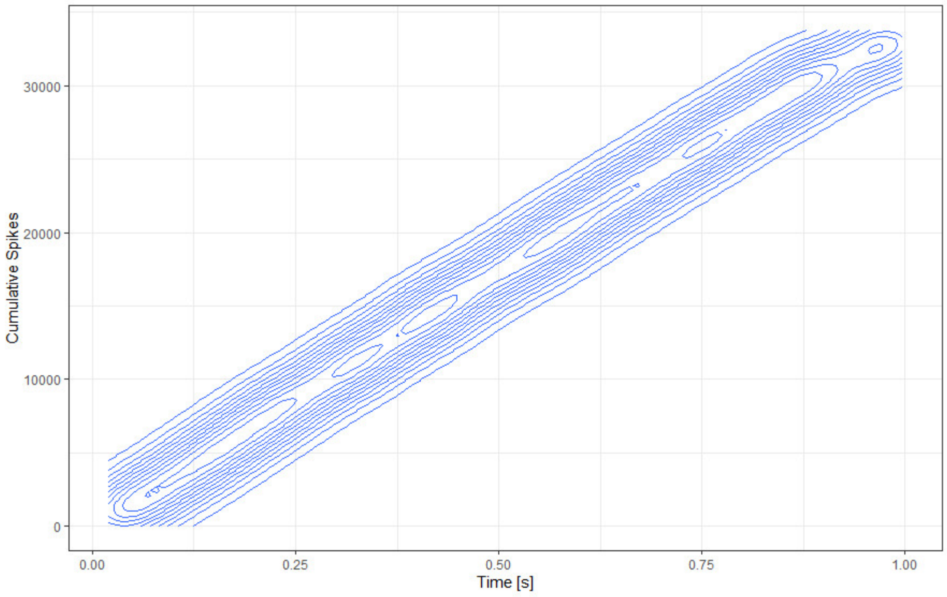
Table 1: Euclidean distance between vectors of states of all four LSM columns

Euclidean Distance	**Column 1**	**Column 2**	**Column 3**	**Column 4**
Column 1	0	3.822376	4.308967	4.375152
Column 2	3.822376	0	1.556321	7.617872
Column 3	4.308967	1.556321	0	8.062106
Column 4	4.375152	7.617872	8.062106	0

cause we confirm the liquid computing abilities of neural microcircuits. We observe that the spiking pattern of each of the four columns is slightly different, what is numerically expressed in the distance values calculated for each pair of columns.

We simulated a single second of this simple visual system in 20000 steps. We measured that in the current setup of the system (the simulation time step of 0.00005 second) that a single second of simulation required 285 CPU seconds (4:45 min). Each LMS column generated on average 1.34 MB of spiking data, with retina adding additional 113 KB.

Apart from gathering the spiking times for each neuron, our system can also measure cell membrane potential for each neural cell in the simulation. In this approach, the amount of data generated increases approximately by 165 times, so the additional stream generated in this way reaches 1 GB per second. We have not noticed any unexpected changes of the data stream over the time of simulations.



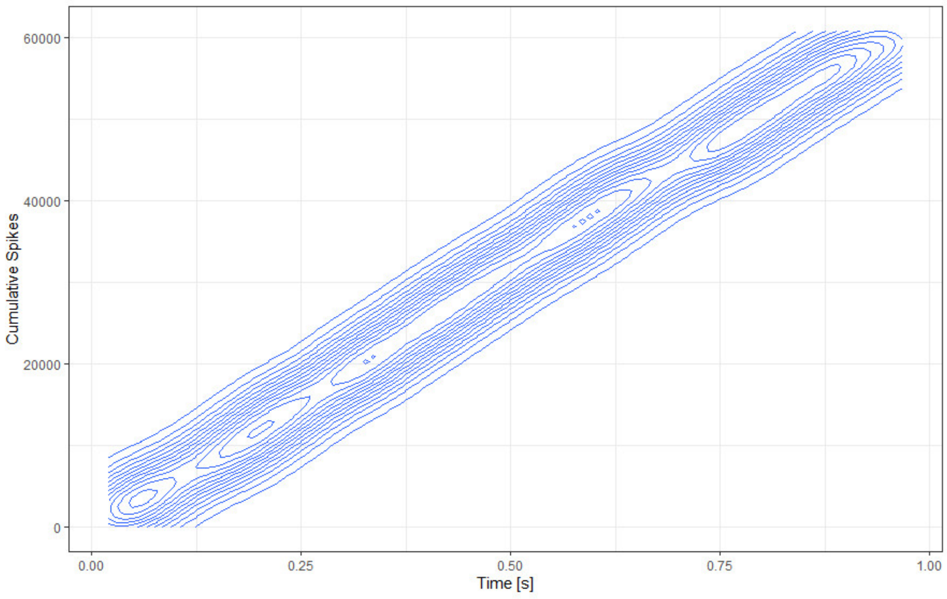
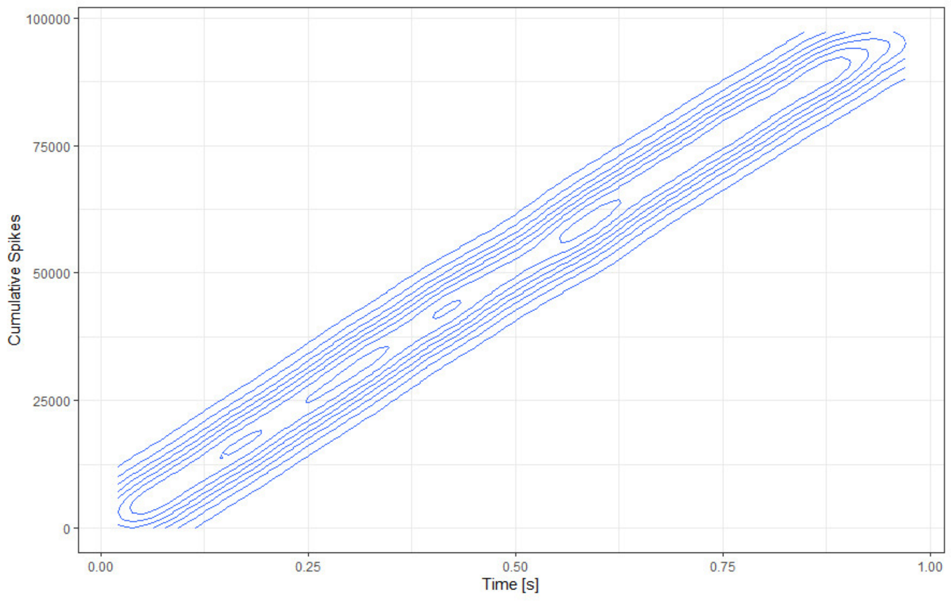


Figure 6: Visual signal processing in the first and second LSM columns of our system



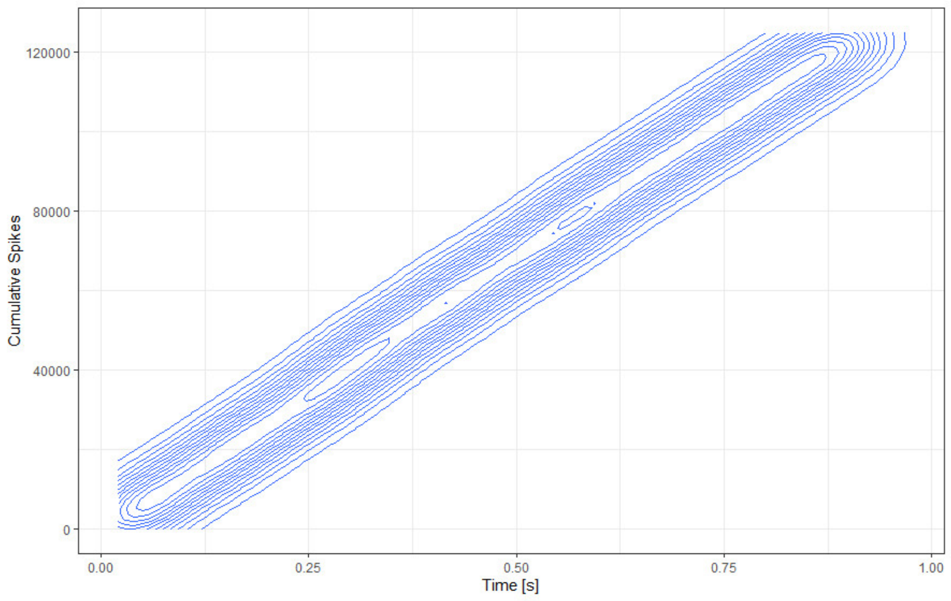


Figure 7: Visual signal processing in the third and fourth LSM columns of our system

4 Discussion

In this paper we demonstrated a proof-of-concept for a liquid state machine, as a bio-inspired computational technique in simulation of a visual system. We report the spike numbers for a complex artificial neural network built of a single 2D retina, and four 3D LSM columns. We simulated 4880 Hodgkin-Huxley neurons organised in layers resembling cortical microcircuits. Application of liquid computing ideas [16] allowed to decrease the number of neurons in the model constructed.

Our results prove that the proposed architecture based on four HHLSMs columns reacts to a visual stimulus in an expected manner, with each of the four LSM columns exhibiting a slightly different spiking pattern. We simulated a single second of the visual system, and that single second required required 285 CPU seconds.

The key limitation of our approach was lack of computational resources to perform longer, and more complex simulation. In future we would also like to use the parallel version of GENESIS to spread the processing across all available CPUs. We believe that the simulation time can be significantly shortened by using a more powerful computational cluster. We expect that our multi-column LSM model can be scaled into a parallel multiprocessor simulation, as the current architecture allows a single column to be simulated on a single processor for optimal performance.

Our model is flexible. The Retina can be formed of other shapes. A single column can be connected to a single retinal cell, or a patch of 4 (2×2), 16 (4×4), 256 (16×16), or more cells, depending on the required stimulus size, and the number of CPUs available. For the proposed retina size (28x28), if each retina cell is connected to a single corresponding HHLSM column, the model can conduct a simulation of about 802 thousand Hodgkin-Huxley neural cells. The current size of retina allows processing of input patterns generated by viewing a MINST [6] dataset, which is a very popular baseline among computer vision researchers.

Apart from neuronal simulations, another possible application of our system could be in automating the operations of the industrial systems, such as the one described in [12]. While evaluating the system, the authors [21] used Electroencephalography (EEG) measurements with EMOTIV EPOC+ 14-Channel Wireless Headset. They reported that the highest increase in mental involvement of the human operator happened during the data supplementation phase, where the value of the investigated mental load reached 90%. Our system could reduce the degree of the operator's mental involvement by providing an intelligent 'robotic eye'. The topology of the container mentioned in [12] allows the system to use our LSM model without any modifications of the retina.

Authors believe that much more realistic models could be developed on the basis of the modelling work presented in this paper. One of the directions of further development of the model proposed could be to include the lateral inhibitory connections between the LSM columns. Inhibitory connections are very common in the neocortex, and lateral inhibition has been shown to play an important role in sharpening the distinctions between similar inputs [5]. Additionally, lateral inhibition plays an important role in population coding, stabilising the mean field potential, and greatly improving the ability of a group of neurons to accurately represent a given stimulus by creating negatively correlated firing patterns [7]. As such, this feature could enhance pattern recognition capabilities of the system, and it is worth including in next version of the model.

These models could be analysed in real time by streaming readout to the Apache Spark cluster, and pushed out to designated databases or live dashboards.

5 Conclusions

To conclude, the proposed LSM architecture coupled with Apache Spark allows for the analysis and visualisation of results for an ever increasing number of simulated neurons, potentially reaching even several millions of neural cells. Adopting the proposed building blocks extends the toolbox of neuroinformatics. It can lead to the creation of even more sophisticated, higher fidelity models and open new possibilities to observe dynamics of such a visual system in real time. Following Maass's [16] ideas and applying a readout for liquid state analysis we can also imagine some new expert-devices able to classify geometrically different and time variable patterns. Therefore, implementing neural models and arranging a proper LSM architecture of the simulated cortex can lead to a better understanding of both machine vision and pattern recognition phenomena taking place in the real brains.

Acknowledgement

We thank Michael Connolly for proofreading.

References

[1] A. P. Alivisatos, M. Chun, G. M. Church, R. J. Greenspan, M. L. Roukes, and R. Yuste. The brain activity map project and the challenge of functional connectomics. *Neuron*, 74(6):970–974, 2012.

[2] E. Bisong. Google colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64. Springer, 2019.

[3] J. M. Bower and D. Beeman. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System.* Springer Science & Business Media, 2012.

[4] J. M. Bower and D. Beeman. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System.* Springer Science & Business Media, 2012.

[5] E. M. Callaway. Local circuits in primary visual cortex of the macaque monkey. *Annual review of neuroscience*, 21(1):47–74, 1998.

[6] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[7] S. Durrant and J. Feng. Negatively correlated firing: the functional meaning of lateral inhibition within cortical columns. *Biological cybernetics*, 95(5):431–453, 2006.

[8] A. Gupta, Y. Wang, and H. Markram. Organizing principles for a diversity of gabaergic interneurons and synapses in the neocortex. *Science*, 287(5451):273–278, 2000.

[9] M. Helias, S. Kunkel, G. Masumoto, J. Igarashi, J. M. Eppler, S. Ishii, T. Fukai, A. Morrison, and M. Diesmann. Supercomputers ready for use as discovery machines for neuroscience. *Frontiers in neuroinformatics*, 6:26, 2012.

[10] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[11] A. Holden, J. Tucker, and B. Thompson. Can excitable media be considered as computational systems? *Physica D: Nonlinear Phenomena*, 49(1-2):240–246, 1991.

[12] M. Jabłoński, R. Tadeusiewicz, A. Piłat, J. Walczyk, P. Tylek, J. Szczepaniak, F. Adamczyk, M. Szaroleta, T. Juliszewski, and P. Kiełbasa. Vision-based assessment of viability of acorns using sections of their cotyledons during automated scarification procedure. *Bio-Algorithms and Med-Systems*, 14(1), 2018.

[13] J. Jordan, T. Ippen, M. Helias, I. Kitayama, M. Sato, J. Igarashi, M. Diesmann, and S. Kunkel. Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. *Frontiers in neuroinformatics*, 12:2, 2018.

[14] S. Kunkel, T. C. Potjans, J. M. Eppler, H. E. E. Plesser, A. Morrison, and M. Diesmann. Meeting the memory challenges of brain-scale network simulation. *Frontiers in neuroinformatics*, 5:35, 2012.

[15] R. Lent, F. A. Azevedo, C. H. Andrade-Moraes, and A. V. Pinto. How many neurons do you have? some dogmas of quantitative neuroscience under revision. *European Journal of Neuroscience*, 35(1):1–9, 2012.

[16] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[17] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

[18] T. Natschläger, W. Maass, and H. Markram. The "liquid computer": A novel strategy for real-time computing on time series. *Special issue on Foundations of Information Processing of TELEMATIK*, 8(ARTICLE):39–43, 2002.

[19] C. E. Shannon. A universal turing machine with two internal states. *Automata studies*, 34:157–165, 1956.

[20] R. Tadeusiewicz. Neural networks as a tool for modeling of biological systems. *Bio-Algorithms and Med-Systems*, 11(3):135–144, 2015.

[21] R. Tadeusiewicz, P. Tylek, F. Adamczyk, P. Kiełbasa, M. Jabłoński, Z. Bubliński, J. Grabska-Chrząstowska, Z. Kaliniewicz, J. Walczyk, J. Szczepaniak, et al. Assessment of selected parameters of the automatic scarification device as an example of a device for sustainable forest management. *Sustainability*, 9(12):2370, 2017.

[22] P. Wijesinghe, G. Srinivasan, P. Panda, and K. Roy. Analysis of liquid ensembles for enhancing the performance and accuracy of liquid state machines. *Frontiers in neuroscience*, 13:504, 2019.

[23] G. M. Wojcik and W. A. Kaminski. Liquid computations and large simulations of the mammalian visual cortex. In *International Conference on Computational Science*, pages 94–101. Springer, 2006.

[24] G. M. Wojcik and W. A. Kaminski. Liquid state machine and its separation ability as function of electrical parameters of cell. *Neurocomputing*, 70(13-15):2593–2597, 2007.

[25] T. Yorozu, M. Hirano, K. Oka, and Y. Tagawa. Electron spectroscopy studies on magneto-optical media and plastic substrate interface. *IEEE translation journal on magnetics in Japan*, 2(8):740–741, 1987.

[26] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, pages 15–28, 2012.

[27] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, et al. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[28] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the twenty-fourth ACM symposium on operating systems principles*, pages 423–438, 2013.

[29] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

Author index

Bartosik, Bernadetta 223
Berezowski, Filip 77
Bielecki, Bartłomiej 141
Bylina, Beata 155, 167
Bylina, Jarosław 77, 167

Chlasta, Karol 233
Chromiak, Michał 59
Codello, Ireneusz 43

Klisowski, Michał 95
Kobus, Adam 43
Krajka, Andrzej 141
Księżopolski, Bogdan 111
Kuniszyk-Jóźkowiak, Wiesława 43
Kuzdraliński, Adam 111

Miśkiewicz, Marek 111

Olszewski, Paweł 129

Panas, Ewelina 187
Piekarz, Monika 9
Potiopa, Joanna 155

Rusinek, Damian 111

Stachura-Żurek, Dorota 199

Wiśniewski, Paweł 155
Wójcik, Grzegorz Marcin 43, 223, 233

Żurek, Tomasz 199

The book presents the latest research conducted at Marie Curie-Skłodowska University in Lublin – or in collaboration with its staff – in the broadly understood field of computer science.

The chapters deal with:

- machine learning and automation of detection in various data sets;
- modern cryptography;
- industrial computer science issues;
- more traditional approaches to data processing – like (geographical) data base systems and parallelization;
- social quations related to information technology;
- ethical/philosophical issues of artificial inteligence;
- some research on the functioning of the human brain and nervous system and its simulation possibility.

The book is addressed to all those who want to become familiar with the areas of research conducted by employees of the Maria Curie-Skłodowska University (in particular, the Institute of Computer Science at the Faculty of Mathematics, Physics and Computer Science). Thanks to concise and clear descriptions of many different issues related to the broadly understood computer science, it will be especially useful for IT/CS students and scientists as well as research staff of enterprises.

wydawnictwo.umcs.eu

UMCS
WYDAWNICTWO

9 788322 795309