BIAŁYSTOK UNIVERISTY OF TECHNOLOGY

Advances in Computer Science Research

ISSN 2300-715X

13
2016

13
2016

ISSN 2300-715X

This journal is the continuation of *Zeszyty Naukowe Politechniki Białostockiej. Informatyka*
issued in the years 2002-2012, ISSN 1644-0331

THE ADRESS FOR THE CORRESPONDENCE / ADRES DO KORESPONDENCJI:

„*Advances in Computer Science Research*"
Faculty of Computer Science / Wydział Informatyki
Bialystok University of Technology / Politechnika Białostocka
ul. Wiejska 45a, 15-351 Białystok
tel. 85 746 90 50, fax 85 746 97 22
e-mail: znpb@irys.wi.pb.edu.pl
http://irys.wi.pb.edu.pl/acsr

Printing / Druk: Oficyna Wydawnicza Politechniki Białostockiej
Issue / Nakład: 25 copies / egzemplarzy

CONTENTS

1.	Aleksander **Dańko**, Anna **Sawicka**, Wiktor **Dańko** ON EFFECTIVE ALGORITHMS SOLVING REGULARITY OF MARKOV CHAINS	5
2.	Marcin **Kozniewski**, Mario A. **Cypko**, Marek J. **Druzdzel** HOWRELIABLE IS A MEASURE OF MODEL RELIABILITY? BOOTSTRAP CONFIDENCE INTERVALS OVER VALIDATION RESULTS	27
3.	Dariusz **Małyszko** CONSTRUCTION OF CLASSIFIERS IN GENERALIZED COVERING APPROXIMATION SPACES	43
4.	Marek **Parfieniuk**, Andrzej **Szpakowicz** A SYSTEM FOR EVALUATING PERFORMANCE OF VIDEO CODECS IN IMAGE COMPRESSION	59
5.	Michał **Wołosik**, Marek **Tabędzki** SCREEN KEYBOARD ARRANGEMENT OPTIMIZATION FOR POLISH LANGUAGE	75
	The list of reviewers (2016)	95

ON EFFECTIVE ALGORITHMS SOLVING REGULARITY OF MARKOV CHAINS

Aleksander Dańko[1], Anna Sawicka[2], Wiktor Dańko[3]

[1] alxd@poczta.onet.pl

[2] Polish-Japanese Academy of Information Technology, Warsaw, Poland

[3] Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

Abstract: We propose algorithms deciding whether a Markov chain with an $n \times n$ transition matrix M is regular. The lowest complexity of such an algorithm can be not greater than $O(n^3)$ and we argue that it cannot be essentially diminished.

Keywords: Markov chain, ergodic Markov chain, regular Markov chain

1. Introduction

We recall basic definitions (cf. [10]) concerning Markov chains. Let M be an $n \times n$ transition matrix of a Markov chain. Positions of the matrix M will be denoted by $M[i, j]$, $i, j \in \{1, 2, ..., n\}$. $M[i, j]$ will be called $[i, j]$ position of the matrix M. The chain is called *ergodic* if and only if for all $i, j \in \{1, 2, ..., n\}$ there exists a positive natural number p such that the $[i, j]$ position of p–th power of the matrix M^p is positive. Since the position $m_{ij}^{(p)}$ of the p–th power M^k of M gives the probability that the Markov chain, started in state i, will be in state j after p steps, we can say that a Markov chain is called an ergodic chain if and only if it is possible to go from every state to every other state (i.e., all transitions are ultimately possible). The chain is called *regular* if and only if there exists a positive natural number p, such that all position of the p–th power M^p of the matrix M are positive. Obviously, regular chains are ergodic.

It is well known that the Warshall-Floyd algorithm (cf. [1], [2]) can be used to solve whether a Markov chain is ergodic. Its complexity is $\Theta(n^3)$.

Advances in Computer Science Research, vol. 13, pp. 5-25, 2016.

The question of the existence of a polynomial algorithm for the problem whether an ergodic chain is regular remains open (according to the actual knowledge of the authors).

Moreover, we have not find any informal information about the complexity of a mathematical method (e.g., cf. [7]) related to deciding the regularity of Markov chains.

Some interesting and useful conditions characterizing regularity one can find in [5], [6], [7], [9], [10], [11], [12], [13].

The paper presents a $\Theta(n^3 \cdot \log n)$ algorithm solving the regularity of Markov chains. The data for the algorithm is an $n \times n$ matrix of a Markov chain. The algorithm consists of two modules. The first module realizes the Warshall-Floyd algorithm and decides whether M is a matrix of an ergodic Markov chain. Its (worst-case) complexity is $\Theta(n^3)$.

The second module verifies regularity of the Markov chain. The idea is to calculate powers of a Boolean matrix G – the adjacency matrix of the graph of the Markov chain (for details cf. Section 2). The main problem is to find, for a given n, an upper estimate $U(n)$, such that if there exists m satisfying $G^m = E$, then there exist $k \leq U(n)$ such that $G^k = E$, where E denotes the $n \times n$ Boolean matrix with all positions equal to 1. Another question is to minimize the number of matrix multiplication related to verification the condition $G^m = E$.

In the case, where we shall use an efficient algorithm for matrix multiplication, e.g., four Russian algorithm (cf. [1], [2]) or Strassen algorithm (cf. [1], [2]) we finally obtain the worst-case complexity of the second module at most $\Theta(n^3)$.

The position [10] can be treated as a report on programistic experiments illustrating realizations of the algorithm solving regularity of Markov chains.

2. Markov Chains

In this section we recall basic definition and facts related to Markov chains (cf. [5], [6], [7], [10]). We shall use the term "computation" to describe the process of changing states.

A Markov chain is determined by a finite set of states S. Without loss of generality we shall assume that S is a subset of natural numbers of the form $S = \{1, 2, ..., n\}$. A *computation* of a Markov chain will be understand as a finite or infinite sequence of states $\{s_m\}_{m \in L}$, where L is the set of natural numbers N or an initial subset of natural numbers of the form $N_k = \{0, 1, 2, ..., k\}$. We shall say that computation starts at time 0 and moves successively from one state to another at unit time intervals. s_0 will be called the initial state. For a given state s_i, the next state s_{i+1} is chosen in

a random way. The probabilities of passing from a state i to another next state j of a computation, denoted by m_{ij} or $M[i,j]$, for $i,j \in S$, are fixed and form a square, $n \times n$ transition matrix M of the Markov chain. The matrix M satisfies the following conditions:

$$M[i,j] \geq 0, \text{for all } i,j \in \{1,2,...,n\}, \tag{1}$$

$$\sum_{j=1}^{n} M[i,j] = 1, \text{for all } i \in \{1,2,...,n\}. \tag{2}$$

Each matrix M, with positions being real numbers, satisfying the conditions (1), (2), will be called *probabilistic matrix*.

2.1 Theorem (cf., e.g., [7], [10])

The product of probabilistic matrices is a probabilistic matrix.

□

Let $c =< s_0, s_1, s_2, ..., s_p >$ be a computation of the Markov chain. The number p will be called the number of steps (transitions) of the computation c or the length of the computation c and denoted by $|c|$. By the probability of realization of the computation c we shall mean the number

$$pr(c) = \prod_{i=1}^{p} M[s_{i-1}, s_i] \tag{3}$$

It is easy to note that $pr(c) > 0$ if and only if $M[s_{i-1}, s_i] > 0$, for $i = 1, ..., p$.

Let $C_p(i,j)$ denote the set of all computations of the length p that starts in the state i and ending, after p steps, in the state j. The following theorem gives a method of determining the sum of probabilities of computations of $C_p(i,j)$:

2.2 Theorem (cf., e.g., [10])

Let M be transition matrix of a Markov chain. The position $m_{ij}^{(p)}$ of the p–th power M^p of M gives the probability that the Markov chain, started in state s_i will be in state s_j after a computation containing p steps.

□

We shall now formulate the above theorem in a more technical manner. Let $i,j \in S$ be fixed states.

2.3 Proposition (compare [5], [6], [7], [10])

$$M^p[i,j] = \sum_{c \in C_p(i,j)} pr(c) \tag{4}$$

□

2.4 Corollary (compare [5], [6], [7], [10])

$M^p[i,j] > 0$ if and only if there exists a computation c of the length p that starts in the state i and ending, after p steps, in the state j and such that $pr(c) > 0$.

□

Now, we shall formulate the question of regularity of a Markov chain, represented by a transition $n \times n$ matrix M, by means of a Boolean, we shall formulate the question of regularity of a Markov chain, represented by a transition $n \times n$ matrix G_M, being the adjacency matrix of the transition graph $\mathcal{G}_M$ of the chain.

Let M be transition matrix of a Markov chain. We shall define the transition graph $\mathcal{G}_M =< V, E >$ of the Markov chain. The set V_M of vertices of $\mathcal{G}_M$ is the set S of states, $V = S$, and the set $E_M \in S \times S$ of edges is defined in the following way:

$$< i, j > \in E_M \text{ if and only if } M[i,j] > 0,$$

for $i, j \in \{1, 2, ..., n\}$.

We recall that the adjacency matrix of the graph $\mathcal{G}_M$ is the Boolean matrix G_M or defined as follows:

$$G_M[i,j] = \begin{cases} 1 & \text{if } < i, j > \in E_M, \\ 0 & \text{if } < i, j > \notin E_M. \end{cases}$$

The Boolean matrix G_M will be denoted by G, if it does not lead to any misunderstanding.

In the sequel we shall consider paths of graphs. We recall that by a path of a graph we shall understand a finite sequence of vertices $d =< s_0, s_1, s_2, ..., s_p >$, such that $< s_{i-1}, s_i > \in E_M$, for $i = 1, 2, ..., p$. s_0 and s_p are called, respectively, the initial and the final vertex of the path d. The number p will be called the length of the path d and denoted by $|d|$.

2.5 Proposition (compare [[5], [6], [7], [10])

Let M be transition matrix of a Markov chain and let $\mathcal{G}_M$ be the transition graph of the chain. Let $c =< s_0, s_1, s_2, s_3, ..., s_{k-1}, s_k >$ be a sequence of states of the chain. The following conditions are equivalent:

- *the sequence c is a computation of the chain such that $pr(c)$ is positive,*
- *the sequence c is a path of the graph $\mathcal{G}_M$.*

□

In the sequel the symbol $\otimes$ will be used to denote logical multiplication of Boolean matrices; if A is a $p \times q$ Boolean matrix, B is a $q \times r$ Boolean matrix then $C = A \otimes B$ is a $p \times r$ Boolean matrix satisfying

$$C[i,j] = (A[i,1] \wedge B[1,j]) \vee (A[i,2] \wedge B[2,j]) \vee ... \vee (A[i,q] \wedge B[q,j]),$$

for $i = 1,2,...,p,\ j = 1,2,...,r$, where $\wedge$, $\vee$ denote conjunction and disjunction operations. The matrix $\underbrace{G \otimes ... \otimes G}_{p\ times}$ will be called the p-th power of the matrix G and denoted by G^p.

Similarly, for A and B being $p \times q$ Boolean matrices, we define $A \otimes B$ as a $p \times q$ Boolean matrix C satisfying $C[i,j] = (A[i,j] \vee B[i,j])$, for $i = 1,2,...,p,\ j = 1,2,...,q$.

The following fact is analogous to (4) and belongs to the folklore of the graph theory:

2.6 Proposition (cf. [1], [2])

$G_M)^p[i,j] = 1$ if and only if there exists a path in the graph $\mathcal{G}_M$ of the length p, with the initial vertex i and the final vertex j.

□

Propositions 2.4, 2.6 can be written together in a more readable form and can be treated as an equivalent form of 2.5.

2.7 Theorem (compare [5], [6], [1], [2])

Let M be the transition $n \times n$ matrix of a Markov chain and let G_M be the adjacency matrix of the graph $\mathcal{G}_M$. Then for all $i, j = 1,2,...,n$, the following equivalence holds:

$$M^p[i,j] > 0 \ \ if\ and\ only\ if\ \ (G_M)^p[i,j] = 1.$$

□

Matrices with all positions being positive will be called a positive matrices.

2.8 Corollary

Let M be the transition $n \times n$ matrix of a Markov chain and let G_M be the adjacency matrix of the graph $\mathcal{G}_M$. Then for all $i, j = 1,2,...,n$, the following equivalence holds:

$$M^p \text{ is a positive matrix } if \text{ and only } if \ (G_M)^p = E.$$

□

This fact indicates that the problem, whether a Markov chain is ergodic or regular, that concern the powers of the transition matrix M, can be transformed into questions concerning the powers of the Boolean matrix $G = G_M$:

(e) a Markov chain is ergodic if and only if the following condition is satisfied:
$(\forall i \in S)(\forall j \in S)(\exists p \in N)((G_m)^p[i,j] = 1)$
i.e., for all $i, j \in \{1,2,...,n\}$ there exists a positive natural number k such that the $[i,j]$ position of the matrix $(G_M)^p$ is equal to 1.

(r) a Markov chain is regular if and only if the following condition is satisfied:
$(\exists p \in N)(\forall i \in S)(\forall j \in S)((G_m)^p[i,j] = 1)$
i.e., for all $i, j \in \{1,2,...,n\}$ there exists a positive natural number k such that all the position of the matrix $(G_M)^p$ are equal to 1.

Therefore, the question of regularity of a Markov chain, represented by a Boolean $n \times n$ matrix G_M can be formulated as a question, whether the sequence

$$G_M, (G_M)^2, (G_M)^3, ..., (G_M)^i, (G_M)^{i+1}, ...$$

contains the matrix with all positions equal to 1. Let us note that the number of different $n \times n$ Boolean matrices is finite; there is $2^{(n \cdot n)}$ different Boolean $n \times n$ matrices. This means that the above sequence is periodic (cyclic), and enables us to formulate a "brute force" algorithm solving the regularity of Markov chains:

2.9 Algorithm 1

a) the data:
$G = G_M$, a Boolean $n \times n$ matrix G_M representing transition graph of a Markov chain,
b) the result:
the answer, whether it is a matrix of a regular Markov chain,
c) the idea of the algorithm:

```
begin {Algorithm 1}
{first module - begin}
{this is a modification of  Warshall Floyd algorithm}
{--------------------------------------}
  for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
      tr_cl_G[i][j] := G[i][j];

  for (k=1;k<=n;k++)
    for (i=1;i<=n;i++)
      for (j=1;j<=n;j++)
        if ((tr_cl_G[i][k] = 1)&(tr_cl_G[k][j] = 1))
        then tr_cl_G[i][j] := 1;

  test_erg = 1;
  for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
       if (tr_cl_G[i][j] = 0) test_erg := 0;
{--------------------------------------}
{first module - end}

{--------------------------------------}
{second module - begin}
  if (test_erg = 0) then test_reg = 0
  else
    begin
      H := G;
      p := 1;
      u:= exp(2,n*n);
      if (H = E) then test_reg := 1
      else test_reg := 0;
        while ((test_reg = 0) & (p<=u)) do
          begin
            H := H ⊗ G;
            if (H = E) then test_reg := 1;
            p = p + 1;
          end;
    end;
```

```
{--------------------------------------}
{second module - end}
end.
```

d) the worst–case complexity: $O(2^{(n\cdot n)})$ matrix multiplications,
e) the upper estimate for analyzed powers of the matrix G:
$U(n) = 2^{(n\cdot n)}$,
f) the idea of the correctness of the algorithm:
the correctness of the algorithm is a simple corollary related to remarks made after points (e), (r) above.

Because of its complexity, Algorithm 1 is of any use in practice. As we have mentioned above, the main problem for speed up this algorithm, is to find, for a given n, an upper estimate $U(n)$, such that if there exists q satisfying $(G_M)^q = E$, then there exist $p \leq U(n)$ such that $(G_M)^p = E$. However, another question related to quality of the algorithm, is to minimize the number of matrix multiplication related to verification whether $(G_M)^p = E$.

To construct effective algorithms solving regularity of Markov chains we need some facts related to Boolean matrix multiplication (Section 3) and the length of cycles, containing all vertices, for transition graphs of ergodic Markov chains (Section 4).

3. Remarks on Boolean matrix multiplication

We shall start with several simple facts, concerning Boolean matrix multiplication. In the authors opinion, these simple facts are rather well known and are elements of the practice of matrix multiplication.

A notation will be used: assume natural ordering between logical values: $0 < 1$. We define an ordering in sets of Boolean matrices of the same dimension: for two Boolean $p \times q$ matrices A, B we shall write

$$A \leq B \text{ if and only if } (A[i,j] < B[i,j] or A[i,j] = B[i,j], for i = 1,2,...,p, j = 1,2,...,q).$$

By I_n we shall denote the $n \times n$ Boolean matrix defined as follows:

$$I_n[i,j] = \begin{cases} 1 & \text{if } i \neq 1 \\ 0 & \text{if } i = 1 \end{cases}, \text{ for } i,j = 1,2,...,n.$$

By E_n we shall denote the $n \times n$ Boolean matrix defined as follows:

$$E_n[i,j] = 1, \text{ for } i,j = 1,2,...,n.$$

We shall often write I and E instead of I_n and E_n if it does not lead to any misunderstanding.

We shall start with a simple remark:

3.1 Remark

Let H, $K \geq I$ be $n \times n$ Boolean matrices. Then $H \otimes K \geq H$.

□

3.2 Corollary

Let $K \geq I$ be an $n \times n$ Boolean matrix.
Then

$$K \leq K^2 \leq K^3 \leq K^4 \leq ...$$

is a non-decreasing sequence of matrices.

□

We end this section with the following two (equivalent) lemmas:

3.3 Lemma

Let M be transition matrix of a Markov chain with n states and let $G = G_M$ be the $n \times n$ Boolean matrix of the transition graph of the chain. If for some natural m, $G^m = E$, then

$$E = G^m = G^{m+1} = G^{m+2} = G^{m+3} = ...$$

□

3.4 Lemma

Let M be transition matrix of a Markov chain with n states. If for some natural m, M^m is a positive matrix, then

$$M^{m+1}, M^{m+2}, M^{m+3}, ...,$$

are also positive matrices.

□

4. Cycles in graphs of ergodic chains

In this section we shall consider transition graphs of ergodic Markov chains. It is easy to argue that in the case of graphs of ergodic chains there exists cycles containing all vertices. Our aim is to estimate the length of such cycles.

The main fact concerning this problem is the following

4.1 Lemma

Let G_M be the transition graph of an ergodic Markov chain with n states. Then there exists a cycle containing all vertices with the length at most $(n^2-n) = O(n^2)$.

The idea of the proof.

We start with a simple remark: if G_M is the graph of an ergodic chain G_M then there exists a cycle in G_M containing all vertices.

Consider such a cycle $d =< s_0, s_1, s_2, s_3, ..., s_{k-1}, s_k >$, where $s_i \in S = \{1, 2, ..., n\}$, for $i = 1, 2, ..., k$, containing all vertices. Denote by m the number of occurrences of the vertex s_0 in the cycle d. Without loss of generality we can assume, for readability, that $s_0 = 1$. Therefore d can be presented as

$$< 1, s_1^1, s_2^1, ..., s_{r_1}^1, 1, s_1^2, s_2^2, ..., s_{r_2}^2, ..., 1, s_1^m, s_2^m, ..., s_{r_m}^m >$$

where $r_1 + r_2 + ... + r_m + m = k$. Let us assign to each $s \in S\ \{1\} = \{2, 3, ..., n\}$, the number $i(s) \in \{1, 2, ..., m\}$, such that s occurs among $\{s_1^{i(s)}, s_2^{i(s)}, ..., s_{r_{i(s)}}^{i(s)}\}$.

Suppose $m > n$˘1. This means that if we remove from the sequence d all subsequences of the form $< 1, s_1^j, s_2^j, ..., s_{r_j}^j >$, where $j \in \{1, 2, ..., m\}\ \{i(2), i(3), ..., i(n)\}$ then the sequence obtained in this way also forms a cycle of G_M and contains all elements of the set S. Moreover, the number of occurrences of the element $s_0 = 1$ in the obtained sequence does not exceed $n-1$.

In an analogous manner we can repeat this removing for all remaining elements $s = 2, 3, ..., n$ and argue that each element of S does not occur in the result sequence more than $n-1$ times.

□

We now formulate a simple remark:

4.2 Remark

If p is the length of a cycle of G containing all vertices of the Markov chain, then $G^p \geq I$.

□

From Lemma 4.1 we obtain:

4.3 Corollary

Let $G = G_M$ be the $n \times n$ Boolean matrix of the transition graph of an ergodic chain. Then, for some $p \leq n^2 - n$, $G^p \geq I$.

□

The following example shows that there exist transition graphs of Markov chains with n states, such that minimal length of a cycle containing all states, is $\Theta(n^2)$.

4.4 Example

Let

$$V = S = \{1, 2, 3, ..., 3k-1, 3k, 3k+1\},\ n = 3k+1,$$

and

$$\begin{aligned} E \subset V \times V = \{&< 1,3 >, < 1,6 >, < 1,9 >, ..., < 1,3(k-1) >, < 1,3k >\} \cup \\ &\{< 3,6 >, < 3,9 >, ..., < 3,3(k-1) >, < 1,3k >\} \cup \\ &\{< 6,9 >, ..., < 3,3(k-1) >, < 1,3k >\} \cup \\ &... \\ &\{< 3(k-2), 3(k-1) >, < 3(k-2), 3k >\} \cup \\ &\{< 3(k-1), 3k >\} \cup \\ \{&< 3,2 >, < 3,4 >, < 6,5 >, < 6,7 >, ..., < 3k, 3k-1 >, < 3k, 3k+1 >\} \cup \\ &\{< 2,1 >, < 5,1 >, ..., < 3(k-1)-1, 1 >, < 3k-1, 1 >\} \cup \\ &\{< 4,1 >, < 7,1 >, ..., < 3(k-1)+1, 1 >, < 3k+1, 1 >\}. \end{aligned}$$

It is easy to note that the sequence below is an example of a cycle of the graph $< V, E >$, containing all vertices, of minimal length equal to $k \cdot (k+5) = (1/9) \cdot (n-1) \cdot (n+5) = \Theta(n^2)$:

$$\begin{aligned} &1,3,2,\ 1,3,4,\ 1,3,6,5,\ 1,3,6,7,\ 1,3,6,9,8,\ 1,3,6,9,10,..., \\ &1,3,6,9,...,3k,3k-1,\ 1,3,6,9,...,3k,3k+1. \end{aligned}$$

□

5. Polynomial algorithms solving regularity of Markov chains

As we have mentioned above, the initial data for such an algorithm is an $n \times n$ Boolean matrix G_M representing transition graph of a Markov chain. The considerations of

Section 2 enables us to formulate the "brute force" algorithm consisting in analyzing powers of the matrix G_M:

$$G_M, (G_M)^2, (G_M)^3, ..., (G_M)^i, (G_M)^{i+1}, ..., (G_M)^{n \cdot n}.$$

Since the sequence of powers of the matrix G_M is periodic (the number of different $n \times n$ Boolean matrices is $2^{(n \cdot n)}$, we have determined an upper estimate $u(n) = 2^{(n \cdot n)}$, such that if there exists q satisfying $(G_M)^q = E$, then there exist $p \leq u(n)$ such that $(G_M)^p = E$.

We shall now use the fact (Corollary 3.3):

$$\text{if } G^m = E, \text{ then } E = G^{m+1} = G^{m+2} = G^{m+3} = ...$$

to minimize the number of matrix multiplication related to verification whether $(G_M)^p = E$, for some $p \leq 2^{(n \cdot n)}$.

5.1 Algorithm 2

a) the data:
$G = G_M$, a Boolean $n \times n$ matrix G_M representing transition graph of a Markov chain,
b) the result:
the answer, whether it is a matrix of a regular Markov chain,
c) the idea of the algorithm:

```
begin {Algorithm 2}
{first module - begin}
{--------------------------------------}
      . . .
{--------------------------------------}
{first module - end}

{second module - begin}
{--------------------------------------}
  if (test_erg = 0) then test_reg = 0
  else
    begin
      H := G;
      p := 1;
```

```
        u := n*n;
        if (H = E) then test_reg := 1
        else
          begin
            test_reg := 0;
            while ((test_reg = 0) & (p < u)) do
              begin
                p = 2*p;
                H := H ⊗ H;
                if (H = E) then test_reg := 1;
              end;
          end;
      end;
  {--------------------------------------}
  {second module - end}
  end.
```

d) the worst–case complexity: $O(n^2)$ matrix multiplications,

e) the upper estimate for analyzed powers of the matrix G:
$U(n) = 2^{(n \cdot n)}$,

f) the idea of the correctness of the algorithm:
the correctness of the algorithm immediately follows from Lemma 3.3. and the remark formulated below:

□

5.2 Remark (refers to d))

To decide regularity of an ergodic Markov chain suffices n^2 matrix multiplications.

The idea of the proof.

To argue the fact, that for deciding regularity of an ergodic Markov chain suffices $O(n^2)$ matrix multiplications, let us note that the sequence of values of the matrix variable H, produced by the algorithm, is the following sequence of powers of the matrix G:

$$G^1, G^2, G^4, G^8, G^{16}, G^{32}, ...$$

This sequence can be also presented as

$$G^{2^0}, G^{2^1}, G^{2^2}, G^{2^3}, G^{2^4}, G^{2^5}, ..., G^{2^{(n \cdot n)}}, ..., G^{2^m}.$$

and the values of exponents are subsequent values of the variable p. Moreover, the whole number m of repetitions of the loop "while" of the second module of Algorithm 2 is the least number satisfying $2^m \geq 2^{(n \cdot n)}$. According to Lemma 3.3 if the sequence

$$G^1, G^2, G^3, G^4, G^5, ..., G^{2^{(n \cdot n)}},$$

contains the matrix E, $G^k = E$, then $E = G^{k+1} = G^{k+2} = G^{k+3} = ... = G^{2^{(n \cdot n)}}$, and therefore, the sequence

$$G^{2^0}, G^{2^1}, G^{2^2}, G^{2^3}, G^{2^4}, G^{2^5}, ..., G^{2^m}$$

where q is the least number satisfying $2^m \geq 2^{(n \cdot n)}$, also contains the matrix E. Thus $m = O(n^2)$.

□

5.3 Corollary

If we use the ordinary procedure of Boolean matrix multiplication then the worst–case complexity of Algorithm 2 is $\Theta(n^5)$.

□

5.4 Corollary (cf. also Corollary 5.12)

If we use the method of four Russian or the Strassen method for matrix multiplication then the worst–case complexity of Algorithm 2 is of the order less than $O(n^5)$.

□

We shall now use the facts presented in Section 3 (Boolean matrix multiplication) and Section 4 (cycles in transition graphs of ergodic chains) to improve Algorithm 1 in another manner.

5.5 Algorithm 3

a) the data:
   $G = G_M$, a Boolean $n \times n$ matrix G_M representing transition graph of a Markov chain,
b) the result:
   the answer, whether it is a matrix of a regular Markov chain,
c) the idea of the algorithm:

```
begin {Algorithm 3}
{first module - begin}
{--------------------------------------}
      . . .
{--------------------------------------}
{first module - end}

{second module - begin}
{--------------------------------------}
  if (test_erg = 0) then test_reg = 0
  else
    begin
      H := G;
      p := 1;
      u := n*(n-1);
      if (H >= I) then test_I := 1
      else test_I := 0;
      {first loop - begin}
      while (test_I = 0) do
        begin
          H := H ⊗ G;
          if (H >= I) then test_I := 1;
          p = p + 1;
        end;
      {first loop - end}
      {first loop is repeated at most (n^2 - n) times,
       because the chain is ergodic}
      {this loop ends with test_I = 1 and H >= I}
      K := H;
      p := 1;
      u := n*n - n;
      if (H = E) then test_reg := 1
      else test_reg := 0;
      {second loop - begin}
      while ((test_reg = 0) & (p < u))
        begin
          H := H ⊗ K;
          if (H = E) then test_reg := 1;
```

```
            p = p + 1;
          end;
        {second loop - end}
        {second loop is repeated
         ??????? NO MORE ??????? than (n^2 - n) times}
        {if the chain is regular then this loop
         ends with test_reg = 1 and H = E}
      end;
  {--------------------------------------}
  {second module - end}
  end.
```

d) the worst–case complexity: $O((n^2-n)+(n^2-n)) = O(n^2)$ matrix multiplications,

e) the upper estimate for analyzed powers of the matrix G:
$U(n) = (n^2-n)\cdot(n^2-n) = n^2\cdot(n-1)^2$,

f) the idea of the correctness of the algorithm:
the correctness of the algorithm immediately follows from Lemma 4.1 and Corollary 4.3.

5.6 Remark (refers to f) and e))

To decide regularity of an ergodic Markov chain suffices to consider the powers G^q of the matrix G, representing transition graph of the chain, for $q \leq n^2\cdot(n-1)^2$.

The idea of the proof.

Let us note that the second module is realized, provided that the chain is ergodic. In this case (cf. Corollary 4.3) the first loop ends its computation with the value of variable `test_I` being 1 and the value of the variable `H` being a matrix K, $K \geq I$. The second loop starts with the initial value of the variable `K` being the value of the variable `H` at the end of first loop. Then the value of the variable `K` (at the start of the second loop) is the power G^p of the matrix G, where $p \leq n\cdot(n-1)$ (cf. Corollar 4.3) and $G^p > I$. This means that the number of positions of this matrix that are equal to 1 is at least n. Denote by $up(L)$, for an $n\times n$ Boolean matrix L, the number of positions equal to 1. Therefore, according to Corollary 4.3, for the subsequent values of the variable `H`,

$$K, K^2, K^3, K^4, ...$$

during the realization of the second loop, we have

$$up(K) \leq up(K^2) \leq up(K^3) \leq up(K^4) \leq ...$$

Since, for each position (K^m) of this sequence of matrices, $up(K^m) \leq n^2$, this sequence is periodic (cyclic) and the length of the period is less than (n^2-n).
This means that to decide, whether this sequence contains the matrix E, it is sufficient to analyze only (n^2-n-1) first positions of the sequence.
Thus, the final value of the variable `H` at the end of second loop is the power G^q of the matrix G, where $q \leq (n^2-n)\cdot(n^2-n)$.
Recapitulation: Algorithm 3 ends his computation with the value of the variable `H` being E and the value of the variable `test_reg` being 1, if and only if, $G^m = E$ for some m, such that $m \leq (n^2-n)\cdot(n^2-n) = n^2\cdot(n-1)^2 = n^4 - 2\cdot n^3 + n^2$.

□

From Lemma 3.3 immediately follows

5.7 Corollary

A Markov chain with the matrix G is regular if and only if the matrix $G^{n^2\cdot(n-1)^2}$ is equal to E.

□

Now, we shall improve Algorithm 3 using the same method that we have used in passing from Algorithm 1 to Algorithm 2:

5.8 Algorithm 4

a) the data:
$G = G_M$, a Boolean $n \times n$ matrix G_M representing transition graph of a Markov chain,
b) the result:
the answer, whether it is a matrix of a regular Markov chain,
c) the idea of the algorithm:

```
begin {Algorithm 4}
{first module - begin}
{--------------------------------------}
      . . .
{--------------------------------------}
{first module - end}
{second module - begin}
{--------------------------------------}
if (test_erg = 0) then test_reg = 0
else
```

```
  begin
    H := G;
    p := 1;
    u := n*n*(n-1)*(n-1);
    if (H = E) then test_reg := 1
    else test_reg := 0;
    while ((test_reg = 0) & (p < u))
      begin
        p = 2*p;
        H := H ⊗ H;
        if (H = E) then test_reg := 1
        else test_reg := 0;
      end;
  end;
{second module - end}
{---------------------------------------}
end.
```

d) the worst–case complexity: $O(\log_2((n^2-n)\cdot(n^2-n))) = O(\log_2 n)$ matrix multiplications,

e) the upper estimate for analyzed powers of the matrix G:
$U(n) = (n^2-n)\cdot(n^2-n) = n^2\cdot(n-1)^2$,

f) the idea of the correctness of the algorithm:
the correctness of Algorithm 4 is an immediate consequence of Remark 5.9 formulated below:

5.9 Remark (refers to f))

To decide regularity of an ergodic Markov chain suffices to consider the powers of the matrix G representing transition graph of the chain:

$$G^{2^0}, G^{2^1}, G^{2^2}, G^{2^3}, G^{2^4}, G^{2^5}, ..., G^{2^q}, ...$$

for least q satisfying $2^q \geq n^2\cdot(n-1)^2$, *i.e.,* $q \geq \log_2(n^2\cdot(n-1)^2) = O(\log_2 n)$.
The idea of the proof.

From Remark 5.6 it follows that to solve regularity of a Markov chain it is sufficient to analyze the sequence of powers of the matrix G representing transition graph of the chain:

$$G^1, G^2, G^4, G^8, G^{16}, G^{32}, ..., G^m, ..., G^{n^2\cdot(n-1)^2},$$

for $m \leq n^2 \cdot (n-1)^2$. From Lemma 3.3 it follows, that if this sequence contains the matrix E, $G^m = E$, then $E = G^{m+1} = G^{m+2} = G^{m+3} = ... = G^{n^2 \cdot (n-1)^2}$. Therefore, the sequence

$$G^{2^0}, G^{2^1}, G^{2^2}, G^{2^3}, G^{2^4}, G^{2^5}, ..., G^{2^q},$$

where q is the least number satisfying $2^q \geq U(n) = n^2 \cdot (n-1)^2$, also contains the matrix E. It is easy to note that $q = O(\log_2 n)$.

□

The theorem below summarizes all the observations:

5.10 Theorem

M is the matrix of a regular Markov chain if and only if all elements of the sequence of powers of the matrix G,

$$G^{U(n)}, G^{U(n)+1}, G^{U(n)+2}, ...$$

are equal to the matrix E.

□

In the case, where we shall use efficient algorithms for matrix multiplication, e.g., four Russian algorithm or Strassen algorithm we finally obtain the worst-case complexity of the second module at most $\Theta(n^3)$. The worst–case complexity of four Russian algorithm (cf. [1], [2]) is $O(n^3/\log_2 n)$. The worst–case complexity of Strassen algorithm (cf. [1], [2]) is $O(n^{\log_2 7})$.

5.11 Corollary

Consider the case, where the Boolean matrix multiplication $\otimes$ is realized in Algorithm 4 by means of four Russian algorithm (cf. [1], [2]). Then the worst–case complexity of this version of Algorithm 4 is $O(n^3)$.

□

5.12 Corollary

Consider the case, where the Boolean matrix multiplication $\otimes$ is realized in Algorithm 4 by means of Strassen algorithm (cf. [1], [2]). Then the worst–case complexity of this version of Algorithm 4 is less than $O(n^3)$.

□

6. Final remarks

It was a little surprising for the authors that the modules corresponding respectively to investigations of ergodity and regularity of Markov chains are of comparable complexity.

We would like to stress that the proof of the correctness of the algorithms proposed in the paper do not make any use of facts characterizing Markov chains in terms of eigenvalues of matrices.

6.1 Remark

Algorithms presented in the paper were implemented and tested by Krystian Moraś in [10].

The implemented (in C++) version of Algorithm 4 one cane find at: p.wi.pb.edu.pl/wiktor–danko .

References

[1] Aho A.V., Hopcroft J.E., Ullman J.D.: Design and Analysis of Computer Algorithms, Addison-Vesley, 1981.

[2] Cormen T., Leiserson C., Riverst R., Stein C.: Introduction to Algorithms, Massachusetts Institute of Technology, 2001.

[3] Dańko A., Dańko W.: Improving Pseudo-Random Generators, International Conference on Biometrics and Kansei Engineering 24-28 June, Cieszyn, Poland, pp. 163-166, 2009.

[4] Dańko W.: The Set of Probabilistic Algorithmic Formulas Valid in a Finite Structure is Decidable with Respect to Its Diagram, Fundamenta Informaticae, vol. 19 (3-4), pp. 417-431, 1993.

[5] Feller W.: An Introduction to Probability Theory and Its Applications, John Wiley and Sons, Inc., New York, London, 1961.

[6] Josifescu M.: Finite Markov Processes and Their Applications, John Wiley and Sons, New York, London 1988.

[7] Kubik L., Krupowicz A.: Introduction to Probability Theory and Its Applications, PWN, Warszawa, 1982, (in Polish).

[8] Mirkowska G., Salwicki A.: Algorithmic Logic, D.Reidel Publ. Co. & PWN, Warsaw, 1987.

[9] Modica G., Poggiolini L.: A First Course in Probability and Markov Chains, John Wiley and Sons, 2013.

[10] Moraś K.: Markov Chains. Analysis of Regular Chains, bachelor thesis, Bialystok University of Technology, 2015, (in Polish).
[11] Ross S. M.: Introduction to Probability Models Academic Press, 2003.
[12] Rubino G., Sericola B.: Markov Chains and Dependability Theory, Cambridge University Press, 2014.
[13] Snell Laurie J.: Introduction to Probability, Random House Inc., New York, 1988.

EFEKTYWNE ALGORYTMY ROZSTRZYGANIA REGULARNOŚCI ŁAŃCUCHÓW MARKOWA

Streszczenie W pracy proponujemy algorytmy rozstrzygające regularność łańcuchów Markowa o macierzy przejść rozmiaru $n \times n$. Najniższa złożoność takiego algorytmu może być nie większa niż $O(n^3)$ i podana jest argumentacja, że nie można jej istotnie obniżyć.

Słowa kluczowe: łańcuch Markowa, ergodyczny łańcuch Markowa, regularny łańcuch Markowa

HOW RELIABLE IS A MEASURE OF MODEL RELIABILITY? BOOTSTRAP CONFIDENCE INTERVALS OVER VALIDATION RESULTS

Marcin Kozniewski[1,3], Mario A. Cypko[2], Marek J. Druzdzel[1,3]

[1] School of Information Sciences, University of Pittsburgh, Pittsburgh, Pennsylvania, USA

[2] The Innovation Center for Computer Assisted Surgery, University of Leipzig, Leipzig, Germany

[3] Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

Abstract: A researcher testing a model will frequently question the reliability of the test results, understanding well the intuition that verification performed on a handful of cases is less reliable than verification based on very large numbers of cases. Because a limited number of verification cases happens pretty often in very specific domains, a question of practical importance is, thus, how reliable is a reported reliability measure.
We propose a methodology based on deriving confidence intervals over various measures of accuracy of Bayesian network models by means of bootstrap confidence intervals. We evaluate our approach on ROC and calibration curves derived for a model derived from an UC Irvine Machine Learning Repository data set and a sizeable (over 300 variables) practical model constructed using expert knowledge and evaluated on merely 66 accumulated real patient cases. We show how increasing the number of test cases impacts the width of confidence intervals and how this can aid in estimating a reasonable number of verification cases that will increase the confidence in model reliability.

Keywords: Bayesian networks, bootstrap confidence intervals, validation

1. Introduction

Bayesian networks (BNs) [13] allow for intuitive, flexible, yet theoretically sound, modeling of uncertain domains. They are acyclic directed graphs, in which nodes represent random variables and edges represent direct dependencies between pairs of variables. These dependencies are expressed numerically by means of conditional

Advances in Computer Science Research, vol. 13, pp. 27-41, 2016.

probability distributions of every node conditional on its direct predecessors (parents) in the graph.

BN models readily combine a variety of available information sources, such as data and expert opinion. A variety of methods for building Bayesian network models have been devised. When data are readily available, networks can be learned automatically [14,4,16]. When no data are available, networks can be entirely elicited from experts (e.g., [5], described later in this paper). Combinations of the two approaches can be used in all cases when some data are available.

BN models are compact representations of the joint probability distribution (JPD) over the variables that they represent. Given an observation of a subset of BN's variables (evidence), it is possible to calculate the conditional posterior probability distribution over the remaining variables. This probability can be used for risk assessment, diagnosis, prognosis, and other tasks. Before a Bayesian network model can be embedded into a decision support system, one would like to know its reliability in terms of the accuracy of its results. The problem is of particular importance in all applications where a potential system error may cause serious practical implications, such as in most medical applications. Arguably the strongest, objective way of assessing the accuracy of a system is to test the system on a set of cases that it has never seen. Several statistical methods are used for assessing different aspects of model quality, such as accuracy, sensitivity and specificity, receiver operating characteristic (ROC) curves, area under the ROC curve (AUC), calibration curve, etc. [8,9,12,6,7].

When the network has been learned from data, one can set aside a subset of data and use that subset for the purpose of verification after learning the network from the remaining (training) records. When a system has been constructed from expert opinion, one can collect independently real cases and use these for verification. In both cases, the number of verification cases is usually limited, either because of lack of data (for example, one might hope that there are not too many data records for serious airplane malfunctions; a limited number of patients is seen with some rare disorders) or because setting aside data records for verification purposes takes them away from the learning set and reduces the quality of the model at the outset. A researcher testing a model on a limited number of cases will frequently question the reliability of the verification result, understanding well the intuition that tests performed on a handful of cases are less reliable than those based on very large numbers of cases. After all, a limited number of cases will rarely cover all important elements of the model. Practical questions of vital importance are, thus, how reliable is a reported reliability measure and is there a need for more cases for reliable verification.

In this paper, we address these questions by deriving confidence intervals over various measures of accuracy of Bayesian network models. Our approach is based on

producing bootstrap confidence intervals from the available verification data. These intervals are going to be broad when the number of verification cases is small and narrow when their number is large. Too broad confidence intervals will indicate the need for more verification cases and will help in evaluating the usefulness of a new decision aid. Such approach is well established for ROC curves [15]. Furthermore, we propose to analyze the size of confidence intervals for a subset of the testing set. If the size of a confidence interval does not change too much for a smaller subset of test cases, it means that bringing more data will not increase significantly the reliability of the evaluation of the model.

We evaluate our approach on ROC and calibration curves derived for three Bayesian network models: (1) two models learned automatically from the *Cover Type* data set [1] available from the Irvine Machine Learning Repository, and (2) a sizeable (over 300 variables) practical model for cancer therapy planning, constructed using experts' knowledge for representing the tumor-, lymph nodes- and metastasis staging (TNM staging for laryngeal cancer)[17]. In (2), the number of patient records available for validation is small (only 66 prior patient cases), so the question of model reliability is of vital practical interest. In all cases, we show how increasing the number of test cases impacts the width of confidence intervals.

The remainder of this paper is structured as follows. Section 2. explains two widely used methods for the assessment of model accuracy: ROC curves and calibration curves. Section 3. describes our approach to deriving confidence intervals over model validation measures based on bootstrap confidence intervals. Section 4. describes out experiments testing our method in practice.

2. Model Quality Validation Methods

In this section, we review two important measures of accuracy of probabilistic systems: (1) ROC curves, and (2) calibration curves.

2.1 Receiver Operating Characteristic (ROC) Curves

Several measures of accuracy have been proposed for systems performing tasks such as classification, prediction, or diagnosis. The most straightforward method is accuracy, which, unfortunately does not give sufficient insight into system's performance. When the asymmetry among classes is very large, a system betting always on the prevalent class will perform well, while it may be of practical importance to identify correctly unlikely classes (e.g., rare but serious disorders in the domain of medicine) at the expense of overall accuracy. Much better measure of accuracy are per class

parameters known as sensitivity and specificity, which are expressing the system's ability to identify the class correctly when present and when absent, respectively. Most researchers report evaluation results in a *confusion matrix*, which is a table listing the total number of correctly and incorrectly identified instances for each of the classes.

Better yet in characterizing the quality of a system is a plot known as *receiver operating characteristic* (ROC) curve [8,9], which shows the tested system's ability to identify a class. The ROC curve plots the true positive rate (i.e., sensitivity) as a function of the false positive rate ($1-$specificity) and shows the ability of a model to distinguish a class for a continuum of decision criteria. The closer an ROC curve is to the upper left corner (0,1) (i.e., to perfect values of sensitivity=1.0 and specificity=1.0), the better the model. When the decision criterion (e.g., a probability threshold) for choosing a class is changed, the sensitivity and specificity for this class also change, within the constraints shown by the ROC curve. The ROC curve can be seen as a exhaustive collection of confusion matrices, as each point on the ROC curve corresponds to one possible matrix, resulting from the modeler's decision when to identify a class and what sensitivity to choose. The ROC curve shows clearly the compromise that the designer of a decision support system has to make by choosing a threshold (and fixing the combination of sensitivity and specificity values), that optimizes the utility of a decision. Figure 1 shows ROC curves for a collection of models predicting the day of female ovulation in the context of a model for fertility awareness [11]. The curves illustrate the idea that the exact values of sensitivity and specificity are the result of a designer's choice. When the model is used by a couple seeking pregnancy, the optimal point and the optimal model are different than when it is used by couples who want to avoid pregnancy. In the latter case, one would want to choose a model with near perfect sensitivity, i.e., with almost-zero false negatives.

Similarly to sensitivity and specificity, the ROC curve is meaningful only in expressing the system's ability to detect a single class. Whenever a system focuses on detecting multiple classes, or multiple grades of a single class, the ROC curve is plotted for a single class and a single value with all remaining classes or values lumped together as a complement of the class in focus.

2.2 Calibration curve

A less popular but not less important measure of quality of probabilistic systems, such as those based on Bayesian networks, is their accuracy in probability estimates. When a system derives the probability of cancer to be 0.07, for example, one would

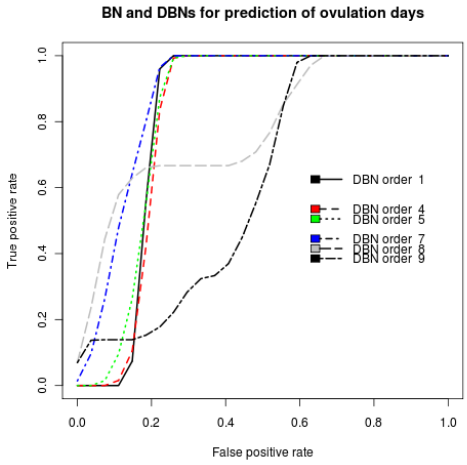


Fig. 1. A collection of ROC curves for a collection of models predicting the day of ovulation [11].

like the actual prevalence of cancer among patients with similar characteristics to be close to 7%. The main reason for importance of precision in probability estimates is that these are fundamental in decision making. No prediction is certain and in order to make a rational decision, a decision maker needs to know the probability distributions over the possible states of the world. Decision theory prescribes an optimal decision to be one that maximizes the expected utility [2]. Utility is a measure of desirability of outcomes that can be combined in the same way as probabilistic expectation. The more precise the probability estimates, the better quality of the resulting decisions.

Accuracy of probability estimates is expressed by *calibration curves*, which are also known in the area of weather forecasting as *reliability diagrams* [12,6]. While the term *reliability diagram* testifies to the importance of accuracy in probability estimates in practical systems, we find it somewhat too general and prefer the term *calibration curve* instead.

The calibration curves express the relationship between the estimated probabilities (horizontal axis) and the observed frequencies (vertical axis) in the data set. In practice, calibration curves are constructed by dividing records with similar probability estimates $\Pr(E)$ produced by the system into bins. For each bin, the frequency $f(E)$ of the event E in the data (e.g., the prevalence of the class in question) is calculated. Calibration curve is a plot of $f(E)$ as a function of $\Pr(E)$. A model that is perfectly calibrated will have a calibration curve that is a diagonal line from the point

(0,0) to point (1,1). Every probability estimate of the model corresponds to identical frequency in the data.

Figure 2 shows an example calibration curve for a system's estimate of the tumor state *t4a* from a TNM staging model. The curve departs from the diagonal line and is also rugged, which is caused by the small size of the test data set (only 66 patient records).

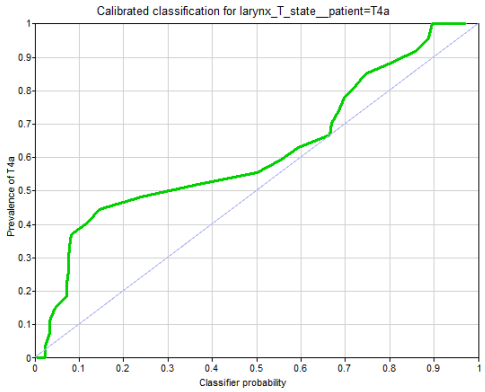


Fig. 2. The calibration curve for the probability of $T4a$ state produced by a TNM staging model.

The calibration curve is capable of expressing overconfidence (or underconfidence) of a system, which provides important information for model builders.

3. Bootstrap Confidence Intervals for Evaluation Statistics

A simple statistical technique for deriving confidence intervals over an estimate can be based on bootstrap resampling [7,3]. Given a representative sample, bootstrap resampling allows to create a large collection of samples with similar statistical properties. Bootstrap resampling has been shown to provide very good results when the original sample is representative for the population. If all records in the sample are indeed independent and identically distributed and originate from the same joint probability distribution, the sample has a very high chance of being representative, i.e., reflect the statistical properties of the distribution. In our work, we start with the sample used for model validation. We assume that it is representative and subsequently use bootstrap resampling to derive the confidence intervals over any statistic based on the original sample. While bootstrap resampling can be used to derive any statistic,

we demonstrate the power of this technique on two statistics of interest in the context of model validation: (1) confidence intervals over the ROC curves, and (2) confidence intervals over the calibration curves.

In the remainder of this section, we describe how confidence intervals and confidence areas for validation curves are obtained. Then, we sketch a technique for capturing the change in the confidence intervals as a function of the size of the test data set.

3.1 Bootstrap confidence intervals

Bootstrap confidence intervals are constructed from statistics calculated for artificial samples drawn from the original sample with replacement. The general procedure is as follows.

i. Having a data set D with n records, create m bootstrap samples by drawing n elements with replacement from D.
ii. For each bootstrap sample D_j, $j = 1, \ldots, m$, calculate the desired statistic for each sample.
iii. Sort the calculated statistics to get $a^{(1)} \leq a^{(2)} \leq \cdots \leq a^{(m)}$.
iv. Take the $(m(\alpha/2))$th element $(a^{(m(\alpha/2))})$ as the lower bound and the $(m(1-\alpha/2))$th element $(a^{(m(1-\alpha/2))})$ as the upper bound of the $100(1-\alpha)\%$ confidence interval.

This method extends readily to confidence intervals over a curve by calculating the confidence intervals over every point on the curve. For each x value we obtain m y-values, which creates a sample to produce a confidence interval of y for each value of x. The extended method can be described schematically as follows.

i. Create m bootstrap samples by drawing n elements with replacement from D.
ii. For each bootstrap sample, create a curve c_j (i.e., ROC or calibration curve).
iii. Iterate through the representative set of values of $x^* \in [0,1]$ (e.g., 100 values from range $[0,1]$ with step of 0.01):
   (a) Generate the set of y values that reflects all the points from generated curves with $x = x^*$,
   (b) Sort the y values
   (c) Create a confidence interval $[y_L, y_U]$ over the y values by taking the $(m(\alpha/2))$th and the $(m(1-\alpha/2))$th elements.
iv. This procedure results in construction of two curves: (1) one curve representing the lower bound of the confidence interval, constructed by points of the y_L value for a given x, and (2) second curve as the upper bound is constructed by the y_U values.

By connecting all lower bounds and all upper bounds of these intervals we obtain the lower and upper bound of the confidence region. Figure 3 shows an example of the confidence region over a ROC curve. In iii we use an iteration step of 0.01.

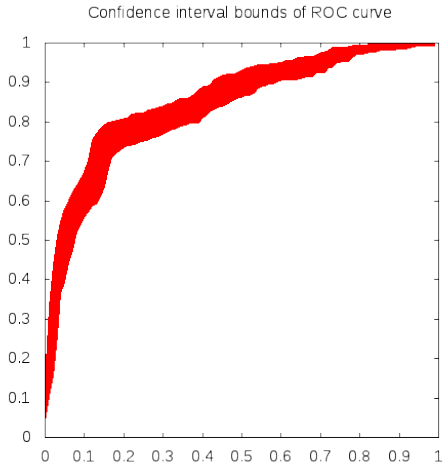


Fig. 3. Sample confidence region over a ROC curve

Confidence regions express the uncertainty about the statistic derived for the model, conditional on the test set used. A plot of the confidence region is useful in determining whether the given data set is sufficient for model assessment with a given validation method.

3.2 Dynamics of the Area of a Confidence Region

A plot of the confidence region suggests a summary statistic, which is the total area of the confidence region (ACR). For both ROC and calibration curves, ACR ranges between 0 and 1. When no validation records are available, there is maximum uncertainty about the ROC and calibration curves and ACR takes the value of 1. We expect, that as the size of the validation set approaches infinity, the area of the confidence region will approach a constant value.

A useful extension of ACR, showing the dynamics of the confidence regions, is a plot of the ACR as a function of the size of the validation set. The larger the validation data set, the smaller the area of the confidence region. We can simulate it by creating subsets of the original dataset. Plotting the area of the confidence region

as a function of the validation set size may be helpful in finding the optimal test set size. We will show the ACR plots in our experiments in Section 4..

4. Evaluation

We evaluate our approach on ROC and calibration curves derived for three Bayesian network models: (1) two models learned automatically from the *Cover Type* data set [1] available from the Irvine Machine Learning Repository, and (2) a sizeable (over 300 variables) practical model for cancer therapy planning, constructed using experts' knowledge for representing the tumor-, lymph nodes- and metastasis staging (TNM staging for laryngeal cancer). The number of validation records (real patient cases) available for the cancer therapy planning model is very small (only 66 prior patient cases). We treat the analysis of this model as an inspiration for a practical application of our work.

4.1 Cover Type Data Set and Models

Our first two models are derived by means of a Bayesian network learning algorithm from the Cover Type data set [1], available from the UC Irvine Machine Learning Repository[4]. The data set consists of 581,012 records over 55 variables. We discretized the continuous variables using uniform interval width discretization method with three intervals. We extracted two disjoint data sets from the original data set for the purpose of our analysis: (1) training data set, one consisting of 81,012 records, and (2) validation data set of 15,000 records for our tests. For the purpose of our experiments, we created a family of validation data sets from (2) by taking subsets of the first 100 records (D_1), the first 200 records (D_2), etc., in such a way that $D_1 \subset D_2 \subset \ldots \subset D_k \subset D$. These subsets simulate the natural process of harvesting more data records for the validation data set.

We created two models from the Cover Type data set (we will refer to them as A and B). For the first model (A), we used all variables and the Bayesian search algorithm [4] implementation in *GeNIe* software[5]. The model consisted of 55 variables connected by a total of 175 arcs. We obtained the second model (B) by mutilating model A. Our mutilation amounted to removing five strongest arcs in model A. Arc strength is a standard function of GeNIe and its theoretical background is described in [10].

[4] `https://archive.ics.uci.edu/ml/datasets/Covertype`

[5] Available free of charge for academic research and teaching use at `http://www.bayesfusion.com/`

We also created a modification of the validation set that contained 50% missing values (these were selected randomly, using uniform distribution).

We ran three experiments for the models A and B:

- model A with original validation data set,
- model B with original validation data set, and
- model A with validation data set with missing values.

In each experiment, we generated regions of confidence as described in Section 3.2 with the size of the validation data set ranging from 100 to 15,000. For each confidence region, we reported the area of the region and the longest confidence interval among x values.

4.2 The TNM-Staging Model

The third model used in our experiments is a detailed representation of the NM staging of laryngeal cancer, created manually by clinical experts from the Leipzig University Hospital (Universitätsklinikum Leipzig, UKL) [17]. The model consists of 303 vertices connected by 334 arcs. Model variables have between 2 and 27 states with average of 4 states. The model is specified by a total of 78,606 parameters obtained from experts. Even though the UKL hospital is highly specialized in head and neck cancer, because challenging head and neck cancers are not common, it receives only around 80 patients per year. The medical records covering more than ten years worth of treatment at UKL are stored, although they are incomplete, unstructured, and recorded mainly in free text format. So far, only 66 patient records have been coded and are suitable for use by the system developers. Many values in these 66 records are missing. The number of observations (values of evidence variables) in these cases vary between 35 and 157 (with the average of 78) per patient record. This is an ongoing project and the data from previous years is successively being added to the validation data set. New patient cases are encoded directly in the destination format.

In our experiment, we derived the confidence regions of the ROC curve and the calibration curve for one of the key decision variables (patient larynx T-state) in the TNM staging model for laryngeal cancer.

4.3 Results

Figure 4 shows the areas of confidence region (ACR) as a function of the validation data set size for each of the states of the *Cover Type* variable. To build this plot, we

created a confidence interval plot for every subset D_i of the validation data set (see Section 4.1) and calculated the area of this plot. Analyzing the plots subjectively, we can see that ACR decreases as the number of validation records becomes larger, reaching a plateau for roughly 2,000 records. The plots show the dynamics of this process and suggests that roughly 2,000 validation records is a reasonable number to get an idea of the accuracy of the model's reliability.

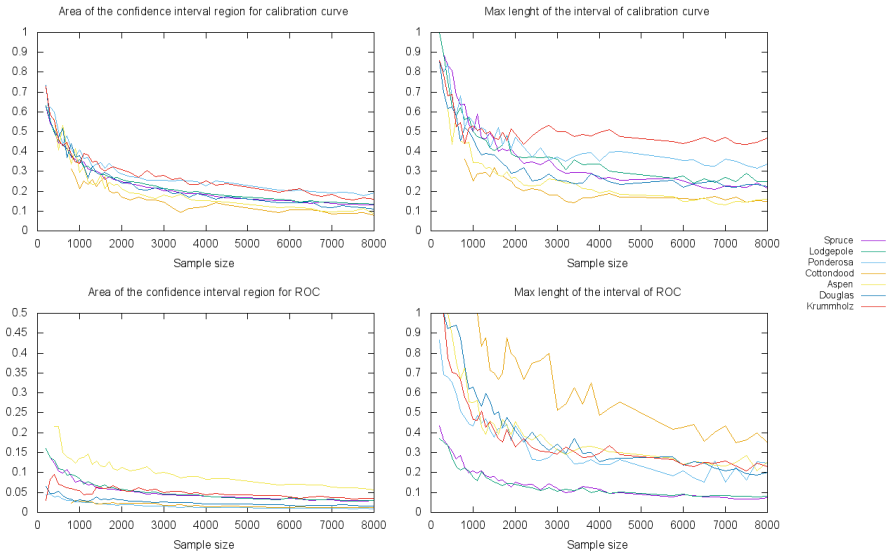


Fig. 4. Changes of areas of confidence regions (ACR) and maximum confidence interval of the ROC and the calibration curve as a function of the number of data records (model A and the original data set).

Figure 5 shows the results of testing the modified model (top) and testing with validation records with missing values (bottom). The results are qualitatively similar to those for model A and the original validation set. Here also 2,000 records seem sufficient for the ACR to reach a plateau beyond which improvement is rather small.

For the TNM staging model, which is a real clinical model, we had only 66 validation records, so the dynamic of the ACR measure is hard to derive. We report only the calibration curves with their 90% confidence intervals. Figure 6 (left) shows the confidence interval of the calibration curve plot for stage T2 laryngeal cancer. The curve seems to suggest that the model is underconfident – the probabilities of the T2 stage of laryngeal cancer tend to be concentrated around moderate probabilities,

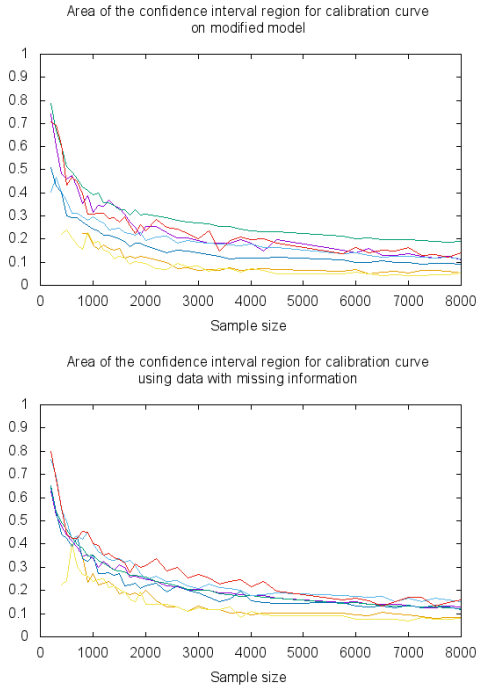


Fig. 5. Changes of areas of confidence region (ACR) as a function of the number of validation data records. Model B tested on the original data set (top). Model A and the data set with missing values (bottom).

while the corresponding frequencies are more extreme. Figure 6 (right) shows an identical plot for the TNM staging model for stage T1a of laryngeal cancer. The model seems to be better calibrated but the confidence intervals are wide and more data are needed to assess the model response more precisely.

5. Conclusion

This paper proposed a methodology for deriving confidence intervals over various measures of accuracy of a Bayesian network model by means of bootstrap resampling. While any validation statistic is amenable for confidence interval analysis, we applied our method to deriving confidence intervals over ROC curves and calibration curves. We have proposed capturing the uncertainty over confidence intervals by means of areas of confidence region (ACR) and have shown how these change as a function of the number of records in the validation data sets. Such plots allow

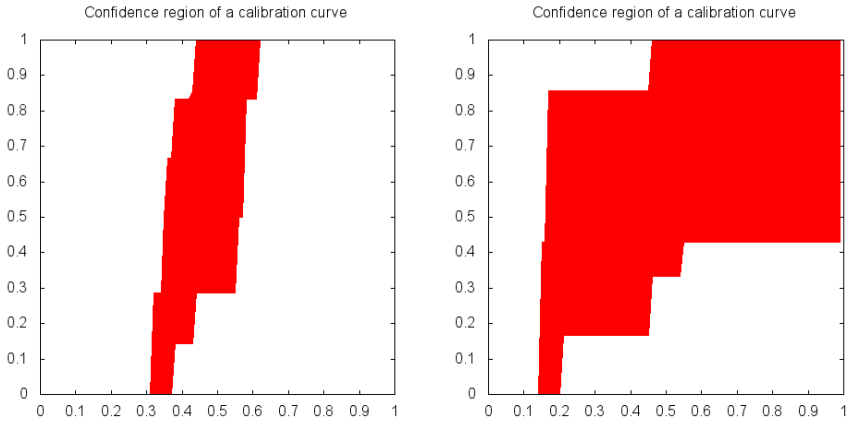


Fig. 6. The 90% confidence region over the calibration curve for the tumor state T2 (left) and state T1a (right) in the TNM-staging model.

for finding a number of records that are necessary to gain reasonable confidence in validity of the model.

The areas of confidence region (ACR) as a function of the number of records curves seems to follow a systematic shape. It may be worth to investigate whether this shape can be characterized theoretically. If so, the number of records needed for a reasonable confidence in evaluation can be predicted a-priori, based on a limited number of validation records.

Finally, our approach gives no insight into the reasons for possibly low confidence in validation, such as flaws in the model structure or its numerical parameters. To gain insight into the possible reasons for low accuracy, more detailed approaches are needed, focusing on the feedback between model inputs and its outputs.

Acknowledgements

We acknowledge the support the National Institute of Health under grant number U01HL101066-01. Implementation of our work is based on GeNIe and SMILE, a Bayesian modeling environment available free of charge for academic research and teaching use at `http://www.bayesfusion.com/` This research has been done in collaboration with the ICCAS research group ”Digital Patient and Process Modeling” funded by the German Ministry of Research and Education.

References

[1] J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, 1999.

[2] R.T. Clemen and T. Reilly. *Making Hard Decisions: Introduction to Decision Analysis*. Duxbury Press, 2005.

[3] P. R. Cohen. *Empirical Methods for Artificial Intelligence*, volume 139. MIT Press Cambridge, 1995.

[4] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[5] M. A. Cypko, D. Hirsch, L. Koch, M. Stoehr, G. Strauss, and Denecke K. Webtool to support medical experts in probabilistic modelling using large bayesian networks with an example of rhinosinusitis. *Studies in Health Technology and Informatics*, 216:259–263, 2014.

[6] M. H. DeGroot and S. E. Fienberg. The comparison and evaluation of forecasters. *The Statistician*, 32:12–22, 1983.

[7] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. CRC Press, 1994.

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

[9] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[10] J. R. Koiter. Visualizing Inference in Bayesian Networks. Master's thesis, Delft University of Technology, June 2006.

[11] A. Łupińska-Dubicka and M. J. Druzdzel. Modeling dynamic processes with memory by higher order temporal models. In A. Hommersom and P.J.F. Lucas, editors, *Foundations of Biomedical Knowledge Representation: Methods and Applications: Lecture Notes in Artificial Intelligence*, volume 9521, pages 219–232. Springer Verlag, 2015.

[12] A.H. Murphy and R.L. Winkler. Reliability of subjective probability forecasts of precipitation and temperature. *Applied Statistics*, pages 41–47, 1977.

[13] J. Pearl. Probabilistic reasoning in intelligent systems, 1998.

[14] J. Pearl and Th. S. Verma. A theory of inferred causation. In J.A. Allen, R. Fikes, and E. Sandewall, editors, *KR–91, Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452, Cambridge, MA, 1991. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[15] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.Ch. Sanchez, and M. Müller. pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, 12(1), 2011.

[16] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer Verlag, New York, 1993.

[17] M. Stoehr, M. Cypko, K. Denecke, H.U. Lemke, and A. Dietz. A model of the decision-making process: therapy of laryngeal cancer. *International Journal of Computer Assisted Radiology and Surgery*, 9(Suppl 1), 2014.

JAK WIARYGODNA JEST MIARA OCENY MODELU? BOOTSTRAPOWE PRZEDZIAŁY UFNOŚCI DLA MIAR DOKŁADNOŚCI MODELU

Streszczenie Przy testowaniu modelu należy zdawć sobie z tego sprawę, że weryfikacja modelu przy pomocy małego zbioru danych jest mniej przekonywująca niż weryfikacja bazująca na dużym zbiorze danych. Często napotyka się sytuację, w której do analizy modelu dysponujemy nieznaczną ilością rekordów. Nasuwa się pytanie o wiarygodność oceny modelu.

Proponujemy w takiej sytuacji przyjrzeć się bootrstrapowym przedziałom ufności różnych miar dokładności modelu. W tej pracy określamy bootstrapowe przedziały ufności dla krzywych ROC i krzywych kalibracji modeli uzyskanych z danych z repozytorium UC Irvine. Czynność powtarzamy dla modelu skonstruwanego na podstawie wiedzy ekspertów (ponad 300 zmiennych) i testowanego na 66 zebranych rekordach pacjentów. Pokazujemy jak wzrost liczby rekordów wpływa na szerokość bootstrapowych przedziałów ufności oraz jak taka analiza może pomóc w określeniu liczby rekordów, która może podwyższyć rzetelność weryfikacji modelu.

Słowa kluczowe: sieci bayesowskie, bootstrapowe przedziały ufności, walidacja

CONSTRUCTION OF CLASSIFIERS IN GENERALIZED COVERING APPROXIMATION SPACES

Dariusz Małyszko

Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

Abstract: Emerging intelligent information systems are pushing existing mathematical foundations into new directions. Generalized covering approximation spaces present abstract data model useful in development of new data analysis methods. The paper introduces construction of rough classifiers in generalized covering approximation spaces. The main idea comes from generation of rough coverings in feature space and calculation of rough covering descriptor. Data are divided into data blocks and each data block statistic and bounding block is calculated . Feature space is divided into feature blocks. For each data bounding block, its inclusion into feature block is calculated and rough covering descriptor is created. Rough covering descriptor is embedded in the generalized covering approximation spaces with standard, fuzzy and probabilistic coverings giving robust theoretical framework in design, implementation and application of classification algorithms.

Keywords: Generalized approximation spaces, covering approximation spaces, rough covering approximation spaces

1. Introduction

Modern computer systems are shifting into transformation to intelligent information systems requiring extension of existing mathematical theories into new directions. Rough set theory presents framework designed for handling imprecise, vague, incomplete, uncertain information. A fundamental methodology of the theory of rough sets has been grounded by means of division of universe object into classes defined by an equivalence relation. In order to extend this point of view, many research has been focused on presenting more generalized data models. Classical rough set theory based equivalence relations have been replaced by more general binary relations, coverings, fuzzy sets, neighborhood systems and general approximation framework.

Recently, the concept of neighborhood has been introduced to define and study different types of neighborhood-based covering rough sets. The covering-based rough

Advances in Computer Science Research, vol. 13, pp. 43-58, 2016.

sets is one of the most important extensions of the classical Pawlak rough sets. In the covering-based rough set theory, various kinds of rough sets were already defined and studied in the literature. The relation among different covering approximation operators has been presented in [1], [2]. In [3] a framework for the study of covering approximation operators by the element based, the granule based and the subsystem based definitions. Approximation operators in covering based rough sets are presented in [4]. Topological properties of covering rough sets are investigated in [5]. Fuzzy coverings rough sets introduced in [6] are linking covering rough set theory and fuzzy rough set theory.

The generalized rough sets in neighborhood system is another important extension of the classical Pawlak rough sets. Rough set extensions have been presented in [7]. Rough feature covering model creates coverings of universe in the generalized approximation spaces. Coverings create neighborhoods of objects, inclusion function is based upon inclusion of coverings.

Rough Extended Framework presented in [8], [9], [10] extensively developed method of data analysis based upon data structure inferred from metric relations in rough, fuzzy and probabilistic approach. The theory of rough sets and fuzzy sets have applied in many image analysis algorithms as described in [11] and can also be combined with various computational intelligence techniques.

Rough covering model embeds into generalized covering approximation spaces with coverings creating neighborhood system, inclusion function describes degree of inclusion for coverings. Established generalized approximation space model gives good foundations for data analysis. Feature coverings are based upon clustering and thresholding of feature space. Rough covering model incorporates standard, fuzzy and probabilistic data model, giving interoperability. The main contribution of the paper is presentation of precise framework for creation rough covering descriptor and algorithm for data classification that is embedded in generalized covering approximation spaces.

This paper has been structured in the following way. In Section 2. the introductory information about rough sets, approximation spaces and generalized covering approximation spaces has been presented. In Section 3. the covering types of generalized covering approximation spaces have been described. Construction of classifiers based upon rough descriptors in generalized approximation spaces is given in Section 4.. Next, concluding remarks and future research are given.

2. Generalized covering approximation spaces

In this Section, definition of generalized deep feature covering approximation space is given with introduction of inclusion function for this space. Next approximation neighborhood systems in these spaces are introduced that form rough approximations of feature coverings.

Definition 1 *Universe consists of objects, further referred to as universe objects. Universe object consists of vector of n-dimensional points by default in R^n space.*

Universe object is the vector of m points, $p = (p_1, \ldots, p_m)$. Each point $p_i = (x_1, \ldots, x_n)$ is described by n attributes or features from attribute set $A = \{a_1, \ldots, a_n\}$. In case of the number of points of the universe object equal 1, the classical definition of the universe is obtained. Each attribute a_i has domain of possible values, forming real interval defined as $da_i = (min_{a_i}, max_{a_i})$. In that way, the concept of attribute space is introduced

Definition 2 *Attribute space or feature space $V = (da_i \times \cdots \times da_n)$ is the space of all possible feature values for each attribute, it means $da_i = (min_{a_i}, max_{a_i})$. Given the universe U, attribute space for this universe will be denoted as V(U).*

Corollary 1. *Both elements of the universe U and attribute space V(U) consist of n-dimensional points so they have the same domain and may be compared.*

Universe objects as the sets of n-dimensional points are divided into data blocks forming data covering. The same applied to attribute space V(U) as the sets of feature vectors divided into feature blocks forming feature coverings.

Definition 3 *A data block $N = \{p_1, \ldots, p_b\}$ is a set of n-dimensional points of universe object. The order of a data block is the number of points of data block.*

Definition 4 *A feature block is a set of n-dimensional points. The order of a feature block is the number of points of feature block including infinite case. Feature blocks of universe objects are denoted as $f=\{p_1, \ldots, p_b\}$ and feature blocks of attribute space are denoted as $F=\{p_1, \ldots, p_b\}$.*

Definition 5 *Selection of subsets of universe creates universe covering and selection of subsets of universe object creates universe object covering.*

Theorem 1. *Each covering (subsets) of universe object of n-dimensional points defines feature blocks f.*

As universe objects consists of n-dimensional points, selecting some number of n-dimensional points fulfills the definition of feature block.

Theorem 2. *Each covering (subsets) of feature space V(U) defines feature blocks F.*

As feature space consists of n-dimensional points, selecting some number of n-dimensional points fulfills the definition of feature block.

Definition 6 *A feature block $F = \{p_1, \ldots, p_m\}$ has min, max and avg values determined as*

$$a(F) = avg(p_1, \ldots, p_m)$$
$$m(F) = min(p_1, \ldots, p_m)$$
$$M(F) = max(p_1, \ldots, p_m)$$

where values avg, min, max are calculated for each attribute over all n-dimensional points forming feature block.

Definition 7 *A hyperbox defines region in n-dimensional space determined by means two n-dimensional points, formally defined as the Cartesian product of intervals In an n-dimensional space of real numbers R^n, let $a, b \in R^n$ be given points such that $a < b$, i.e. $\forall_{1 \leq i \leq n}\ a_i < b_i$. A hyperbox H=(a,b) is the set of points satisfying condition*

$$H \subset R^n, x = \{x_1, \ldots, x_n\}, x \in H \iff \forall_{1 \leq i \leq n} \quad a_i < x_i < b_i$$

where x_j is j-th feature of x.

Theorem 3. *Each feature block F defines H hyperbox in R^n by assuming*

$$H = (m(F), M(F))$$

A direct consequence of theorem 1 is the fact

Corollary 2. *Each data covering defines H hyperboxes in R^n.*

A consequence of theorem 2 is the fact

Corollary 3. *Each feature covering defines H hyperboxes in R^n.*

Corollary 4. *Feature blocks F of attribute space form some kind of universe objects with accordance to definition 1.*

From corollary 4 results that feature blocks F can be compared to universe objects.

Definition 8 *Hyperbox $H = (a,b)$ has min, max and avg values determined as*

$$a(H) = (a+b)/2$$
$$m(H) = a$$
$$M(H) = b$$

In that way, that comparison between data blocks and feature blocks is possible on the base of inclusion of their respective hyperboxes. Selection of similar indiscernible objects creates elementary sets that describe our knowledge about the universe. Elementary sets are called precise sets. All other sets are considered to be rough, imprecise, vague. An information system IS = (U,A) consists of objects described by attributes. Attribute a_i is defined as a function $a_i : U \to V_{a_i}$ where V_{a_i} is a set of attribute values. The same applies to extended definition of universe objects from definition 1. Information system with objects divided into classes by reflexive, symmetric and transitive equivalence R on U is called an approximation space. Lower and upper approximations of any subset X of U are defined as two sets completely contained or partially contained in the equivalence classes. This approach have been generalized by extending equivalence relations to tolerance relations, similarity relations, binary relations leading into formulation of the concept of Generalized approximation spaces.

Definition 9 *A generalized approximation space is a tuple GAS = (U, N, $\mathfrak{v}$) with N is a neighborhood function defined on U with values in the powerset $P(U)$ of U. The overlap function $\mathfrak{v}$ is defined on the Cartesian product $P(U) \times P(U)$ with values in the interval [0, 1] measuring the degree of overlap of sets.*

The lower GAS_* and upper GAS^* approximation operations can be defined in a GAS by

$$GAS_*(X) = \{x \in U : \mathfrak{v}(N(x),X) = 1\},$$
$$GAS^*(X) = \{x \in U : \mathfrak{v}(N(x),X) > 0\}.$$

Generalized approximation spaces present environments with specialized rough set models applied such as similarity based rough set model with reflexive neighborhood function and variable precision model with different thresholds definition in overlap functions.

Definition 10 *Let C be a family of nonempty subsets of U creating a covering of U. The ordered pair CAS = (C, U) is called a covering approximation space.*

Definition 11 *Let every universe object x of U has a family of nonempty subsets of x creating a covering of x. The ordered pair DAS = (D, U) is called a deep covering approximation space.*

Starting from covering approximation spaces and defining more specialized neighborhoods with overlap function we obtain the following definitions.

Neighborhood system N_S with two distinct coverings L, U for each universe object is called approximation neighborhood system.

Definition 12 *Let C be a family of nonempty subsets of attribute feature space V(U) associated with U creating a covering of U. The ordered pair CAS = (C,V(U)) is called a feature covering approximation space.*

Definition 13 *Given covering approximation space CAS, approximation neighborhood system N_S assigns for each object x two coverings L, U containing that object* $N_S = \{L,U \in \mathcal{C} : x \in L,U\}$

Approximation neighborhood system R_S with two distinct coverings L, U for each universe object satisfying condition $L \subset U$ is called rough approximation neighborhood system R_S.

Definition 14 *In covering approximation space CAS, rough approximation neighborhood system* $R_S(x) = \{L,U \in \mathcal{C} : x \in L,U\}$ *defines two coverings L, U containing x, satisfying* $L \subset U$.

Definition 15 ***A generalized covering approximation space*** *is a system GCS = (U, C, R_S, υ) where R_S is a rough approximation neighborhood system defined on U with covering C, and υ is the overlap function measuring inclusion of coverings from C.*

In order to apply coverings to universe objects the following notion of generalized deep covering approximation space has been introduced.

Definition 16 ***A generalized deep covering approximation space*** *is a system GDS = (U, D, R_S, υ) where R_S is a rough approximation neighborhood system defined on universe objects of U with covering D, and υ is the overlap function measuring inclusion of coverings from D.*

In order to embed into covering approximation spaces additional coverings of feature space the notion of generalized feature covering approximation space has been given.

Definition 17 ***A generalized feature covering approximation space*** *is a system GFS = (U, $\mathcal{C}$, $\mathcal{F}$, R_S, υ) where R_S is a rough approximation neighborhood system defined on U with covering $\mathcal{C}$, covering of feature space $\mathcal{F}$ and υ is the overlap function measuring inclusion of coverings from $\mathcal{C}$ and $\mathcal{F}$.*

Generalized feature approximation spaces have coverings of universe and coverings of feature space. When coverings of universe are replaced by coverings of universe objects the following definition of generalized deep feature covering approximation space is obtained.

Definition 18 ***A generalized deep feature covering approximation space*** *is a system GFDS = (U, $\mathcal{D}$, $\mathcal{F}$, R_S, υ) where R_S is a rough approximation neighborhood system defined on universe objects U with coverings $\mathcal{D}$ of universe objects, covering $\mathcal{F}$ of feature space and υ is the overlap function measuring inclusion of two coverings from $\mathcal{D}$ and $\mathcal{F}$.*

Definition 19 ***Inclusion function*** *in generalized covering approximation spaces has been defined in the way presented below*

The degree of feature hyperbox inclusion represented by feature overlap function is defined as

$$v(H_i,H_j) = \begin{cases} 1 & \text{if } a(H_i) \in H_j \\ 0.5 & \text{if } H_i \cap H_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

with $a(H_i)$ representing average value for hyperbox H_i. This inclusion function is of general purpose type and finds similarity for data represented as multidimensional hyperboxes.

The degree of feature blocks inclusion represented by feature overlap function is defined as

$$v(F_i,F_j) = \begin{cases} 1 & \text{if } a(F_i) \in F_j \\ 0.5 & \text{if } F_i \cap F_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

with $a(F_i)$ representing average value for feature block F_i. This type of inclusion function leads to finding feature block similarity. Given feature block F_i the set of feature blocks similar to F_i can be determined.

The degree of data blocks and feature blocks inclusion represented by overlap function is defined as

$$v(N_i,F_j) = v(H_i,F_j) = \begin{cases} 1 & \text{if } a(N_i) \in F_j \\ 0.5 & \text{if } H_i \cap F_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

with $a(N_i)$ representing average value for data block N_i. Feature blocks F_i and data blocks N_i represent some kind of universe objects as given in corollary 4, 1 and may be compared by inclusion function. The inclusion $v(N_i,F_j)$ gives information which feature blocks are similar to given data blocks.

The degree of feature blocks and data blocks inclusion represented by overlap function is defined as

$$v(F_i,N_j) = v(F_i,H_j) = \begin{cases} 1 & \text{if } a(F_i) \in H_j \\ 0.5 & \text{if } H_i \cap N_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

with $a(F_i)$ representing average value for feature block F_i. Feature blocks F_i and data blocks N_i represent some kind of universe objects as given in corollary 4, 1 and may be compared by inclusion function. The inclusion $v(F_i,N_j)$ gives information which data blocks are similar to given feature blocks.

The degree of data blocks inclusion represented by overlap function is defined as

$$v(N_i,N_j) = v(H_i,H_j) = \begin{cases} 1 & \text{if } a(N_i) \in H_j \\ 0.5 & \text{if } H_i \cap H_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

with $a(N_i)$ representing average value for data block N_i. This type of inclusion function leads to finding data block similarity. Given data block N_i the set of data blocks similar to N_i can be determined.

In generalized deep covering approximation spaces, existence of universe objects coverings and feature coverings gives possibility of defining two additional rough approximation neighborhood systems.

Approximation neighborhood system F_S assigns two distinct feature coverings L, U for each universe objects coverings satisfying condition $L \subset U$ is called rough approximation neighborhood system F_S.

Definition 20 *Approximation neighborhood system F_S that assigns two distinct universe objects coverings L, U for each feature coverings is called feature approximation neighborhood system F_S.*

Definition 21 *Approximation neighborhood system F_S that assigns two distinct universe objects coverings L, U for each feature coverings satisfying condition $L \subset U$ is called rough feature approximation neighborhood system F_S.*

The concept of rough feature approximation neighborhood system has been defined and applied into generalized deep covering approximation spaces during construction of rough classifier framework. In the introduced rough extended model the universe with data represented by universe objects, is divided into data blocks N_i being covering of universe objects, features space is divided into feature blocks F_i. By selecting inclusion function of type $v(N_i, F_j)$ similarity of data blocks and feature blocks is determined. For each data block N_i its inclusion $v(N_i, F_j)$ in feature block F_j is calculated. Each data block N_i is described by its hyperbox determined by means of min, max feature block f_i values. Image blocks that have feature mean value in the F_i are assigned to its lower approximations, each feature block that is contained in the (min, max) hyperbox are assigned to its upper approximation. Given selected feature block F_j, lower and upper approximation of the feature block is defined as follows

$$G_*(F_j) = \{N_i \in U : \mathfrak{v}(N_i, F_j) = 1\},$$
$$G^*(F_j) = \{N_i \in U : \mathfrak{v}(N_i, F_j) > 0\}.$$

For the feature covering $F = \{F_1, \ldots, F_s\}$ the lower and upper approximaions are defined as follows

$$G_*(F) = \bigcup_{F_i} \{N_i \in U : \mathfrak{v}(N_i, F_j) = 1\},$$
$$G^*(F) = \bigcup_{F_i} \{N_i \in U : \mathfrak{v}(N_i, F_j) > 0\}.$$

Introduced concept of generalized covering approximation spaces gives theoretical foundations into creation of rough feature covering model described in the next section. Metric spaces define the distance function that makes possible to compare

objects, their similarity, relations, data structure and extract fuzzy and probabilistic properties of the objects of the universe giving at the same time interoperability of different data models.

3. Covering models for generalized covering approximation spaces

Covering models are created by means of data object relation to the selected set of data centers. This reference set of data objects performs as the set of thresholds or the set of cluster centers. In general data object are analyzed by means of their relation to the selected number of cluster centers. Cluster centers are regarded as representatives of the clusters.

Standard rough coverings - In standard setting data similarity is measured by means metric. In rough clustering approaches, data points closest to the given cluster center or sufficiently close relative to the selected threshold type, are assigned to this cluster lower and upper approximations. The upper approximations are calculated in the specific, dependent upon threshold type and measure way presented in the subsequent paragraphs.

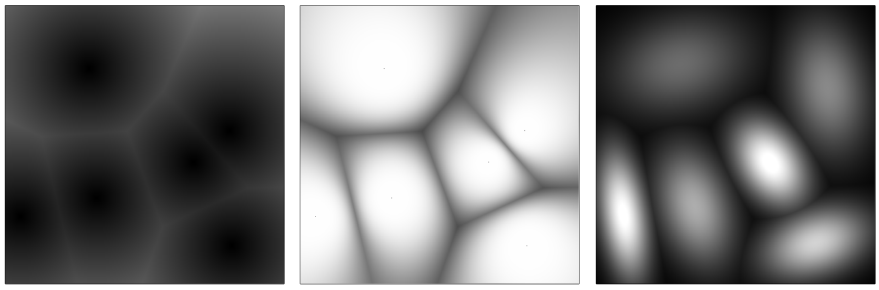
Fuzzy coverings - Fuzzy coverings are created by calculation of fuzzy membership values for all data points of the universe to cluster centers. Assignment to fuzzy coverings is based upon fuzzy membership values satisfying predefined conditions based upon threshold values.

Probabilistic coverings - Probabilistic coverings are calculated by means of probability distribution values within defined limits. In this type of covering probability distributions determines which data belongs to the coverings.

Neighborhood type - Neighborhoods are defined on the base of threshold value for the given type of coverings. Threshold neighborhoods defined data objects that are within given threshold value in the selected covering type, it means standard, fuzzy and probabilistic. Difference neighborhood define similar objects on the base their distance not higher to the nearest neighborhood.

Upper and lower neighborhoods - The upper and lower neighborhood approximations are defined on the base neighborhood type and feature type. In this way, lower approximation and upper approximation are defined independently.

In Fig. 1 three types of coverings have been presented. In Fig. 1 a standard coverings based upon standard metric has been presented. In Fig. 1 b fuzzy coverings based upon fuzzy membership values have been presented. In Fig. 1 c probabilistic coverings are presented.



a b c

Fig. 1. Three types of coverings, (a) standard coverings (b) fuzzy coverings, (c) probabilistic coverings

4. Rough covering classifiers in generalized covering approximation spaces

Construction of rough classifier is presented for image data but the classifier is applicable to any numeric data. The image is denoted as I=$\{p_1,\ldots,p_m\}$ representing image pixels and at the same time forming universe object according to definition 1. Each pixel forms n-dimensional point as values of arbitrary n features denoted as x=$\{x_1,\ldots,x_n\}$. In this way, each image consists of m pixels described by numeric features a. In next step, image blocks are defined as N=$\{N_1,\ldots,N_r\}$ with $K=\{1\ldots,r\}$ denoting the index set for image blocks and $N_i \subset I$ with $N_i=\{p_1,\ldots,p_r\}$. Image blocks are created according to selected block creation strategy. Feature space is divided into feature blocks F=$\{F_1,\ldots,F_s\}$ with $L=\{1\ldots,s\}$ denoting the index set for feature blocks. Further, for each image block $N_i=\{x_1,\ldots,x_r\}$ its min, mean, max values are calculated and hyperbox H_i is determined. Both image blocks and feature blocks may be distinct (separable) or may overlap.

For feature blocks F_k, image blocks N_i that intersect the hyperbox H_i are assigned to their upper approximations $F^*(k)$ and their measures $D^*(k)$ are increased by 1.0. The feature block(s) F_k that contain(s) mean value of N_i, is considered to contain entirely this image block so belongs to its lower approximation $F_*(k)$ and its measure $D_*(k)$ is increased by 1.0. The degree of data blocks and feature blocks inclusion represented by overlap function is defined as

$$v(N_i,F_j)=v(H_i,F_j)=\begin{cases}1 & \text{if } a(N_i)\in F_j\\ 0.5 & \text{if } H_i\cap F_j\neq\emptyset\\ 0 & \text{otherwise}\end{cases}$$

with $a(N_i)$ representing mean value for data block N_i. In case of precisely selected sets N_i and F_j only indices are used giving the following formula

$$d(i,j) = \begin{cases} 1 & \text{if } a(N_i) \in F_j \\ 0.5 & \text{if } H_i \cap F_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Lower approximation $F_*(i)$ of feature block F_i consists of all image blocks N_j that inclusion $d(i,j)$ is equal 1. The cardinality of all $d(i,j) = 1$ of feature block F_i is denoted as $D_*(i)$ and represents measure of the lower approximation

$$D_*(i) = card\{k \in K : d(i,k) = 1\}$$

Upper approximation $F^*(i)$ of feature block F_i consists of all image blocks N_j that inclusion $d(i,j)$ is greater than 0. The cardinality of all $d(i,j) > 0$ of feature block F_i is denoted as $D^*(i)$ and represents measure of the upper approximation

$$D^*(i) = card\{k \in K : d(i,k) > 0\}$$

Quality of approximation $QA(i)$ describes quantitatively ratio between lower and upper approximations of the feature block F_i

$$QA(i) = D_*(i)/D^*(i)$$

Roughness $R(i)$ of feature block F_i gives measure of the uncertainty of the feature block

$$R(i) = 1 - QA(i) = 1 - D_*(i)/D^*(i)$$

As a whole, all $R(i)$ roughness values describe rough descriptor of the universe data.

$$R = \sum_{i \in L} R(i) = \sum_{i \in L} (1 - QA(i))$$

Following presented procedure, the rough covering descriptor is calculated. The algorithm for classification based upon introduced descriptor has been presented in Algorithm 1. In the first step, all required data structures are prepared, it means input data are divided into data blocks, feature space is divided into feature blocks. In the second step, feature covering descriptor is calculated by assigning data blocks to feature blocks and to their lower and upper approximations. After feature covering descriptor has been calculated, the selected classifier is trained and tested.

Algorithm 1: Rough covering classifier

Prepare data structures
Calculate feature covering descriptor
Perform learning phase of selected classifier algorithm
Perform classification

Data structure preparation

Data block creation - data objects can be divided in arbitrary way that is called block creation strategy. For image data, data block creation strategy may consists in creation overlapping or non-overlapping image blocks. For feature block creation, the strategy described in previous Section may be applied with selection of cluster centers - cc, neighborhood type - nt and covering type - ct.

Algorithm 2: Preparation of data structures

Input - I - image, S - block creation strategy, T - covering creation strategy
Output - N = $\{N_1, \ldots, N_r\}$, F = $\{F_1, \ldots, F_s\}$
Divide image into image blocks $->$ N
Divide feature space into feature blocks $->$ F

Rough covering descriptor

Feature covering descriptor is created as for each data block all feature blocks that intersect bounding box of F_i and feature block that contains $a(N_i)$ are found and their lower and upper approximations measures are increased. The calculation of the feature covering descriptor has been presented in Algorithm 2.

Perform classification

After rough covering descriptor has been calculated for all data, classification may be performed by learning phase and testing phase. Rough covering descriptor may be used for any classification framework suited to this purpose such as Support Vector Machines, neural networks.

Algorithm 3: Rough covering descriptor

Input - I - image, N $->$ data blocks, F $->$ feature blocks
Output - R - quality of approximations and roughness of feature blocks F
foreach *Image block* N_i *in* N **do**
  Calculate hyperbox H_i and mean value a(N_i)
  foreach *feature block* F_p *in* F *which contains average value* $a(N_i)$ **do**
    Increment $D_*(p)$ of this feature block by 1.0
  foreach *feature block* F_p *in* F *contained in* H_i **do**
    Increment $D^*(p)$ of this feature block by 1.0

Computational feasibility

The most difficult part in calculation of rough covering descriptor is finding of all coverings embedded in hyperbox H. The routines Contains (F_i, N_i), Contains (N_i, F_i), Get(F_i), Get(N_i) may be helpful in accomplishing this task. Get(N_i) returns all feature blocks F_i that are intersected by hyperblock H_i. Get(N_i) works by calling for each feature block F_i function Contains(N_i, F_i) returning information whether bounding block of N_i intersects F_i. Function Get(F_i) returns all data blocks N_i that intersect feature block F_i. In case of the number of feature blocks is not large, all the feature blocks may be checked. When the number of feature blocks is large, for example feature blocks are generated from 1000 cluster centers, the routine Get(F_i) is more appropriate.

Conclusions and Future Research

In the paper, new algorithmic approach to construction of rough classifiers has been presented by introduction of rough covering descriptors. Introduced rough covering descriptor merges two different approaches in rough sets. Universe objects belong to coverings as covering approximation spaces. In generalized covering approximation spaces functionality of inclusion function has been given from generalized approximation spaces. Granularity concept is employed by selection of different types of data blocks and different feature blocks. Data descriptors are embedded in generalized covering approximation space with coverings in feature space and inclusion function that measures similarity of data blocks and feature coverings. Rough covering descriptors are based upon assignment of data blocks to feature blocks acting as data descriptor. In this way created rough covering descriptor can be easily used in classification frameworks such as Support Vector Machines framework, neural networks. Coverings in generalized covering approximation spaces include standard,

fuzzy and probabilistic coverings giving robust theoretical framework in design, implementation and application of classification algorithms.

References

[1] Sai Y. Liu G.L. A comparison of two types of rough sets induced by coverings. *Int. J. Approx. Reason*, 50:521–528, 2009.

[2] Zhu W. Relationship between generalized rough sets based on binary relation and covering. *Information Sciences*, 179:210–225, 2009.

[3] Yao B. Yao Y. Covering based rough set approximations. *Information Sciences*, 200:91–107, 2012.

[4] Gomez J. Restrepo M., Cornelis C. Duality, conjugacy and adjointness of approximation operators in covering-based rough sets. *International Journal of Approximate Reasoning*, 55:469–485, 2014.

[5] Zhao Z. On some types of covering rough sets from topological points of view. *International Journal of Approximate Reasoning*, 68:1–14, 2016.

[6] Liwen Ma. Two fuzzy covering rough set models and their generalizations over fuzzy lattices. *Fuzzy Sets and Systems*, pages –, 2015.

[7] Skowron A. Pawlak Z. Rough sets: some extensions. In *Information Sciences 177*, pages 28–40, 2007.

[8] Stepaniuk J. Malyszko D. Granular multilevel rough entropy thresholding in 2D domain. *16th International Conference Intelligent Information Systems, Zakopane, Poland*, pages 151–160, 2008.

[9] Stepaniuk J. Malyszko D. Adaptive rough entropy clustering algorithms in image segmentation. *Fundamenta Informaticae*, 98(2-3):199–231, 2010.

[10] Stepaniuk J. Malyszko D. Adaptive multilevel rough entropy evolutionary thresholding. *Information Sciences*, 180(7):1138–1158, 2010.

[11] Peters J. Pal S. K. *Rough Fuzzy Image Analysis: Foundations and Methodologies*. CRC Press Inc, 2010.

KLASYFIKATORY W UOGÓLNIONYCH APROKSYMACYJNYCH PRZESTRZENIACH POKRYĆ

Streszczenie W pracy przedstawiono nowy sposób konstrukcji klasyfikatorów w uogólnionych aproksymacyjnych przestrzeniach pokryć, definiowanych jako przestrzenie aproksymacyjne zawierające przestrzeń obiektów, pokrycia w tej przestrzeni, oraz pokrycia w przestrzeni atrybutów wraz z zdefiniowaną funkcją zawierania się zbiorów zastosowaną dla pokryć.

Słowa kluczowe: przestrzenie aproksymacyjne, przestrzenie pokryć, przybliżone przestrzenie aproksymacyjne

A SYSTEM FOR EVALUATING PERFORMANCE OF VIDEO CODECS IN IMAGE COMPRESSION

Marek Parfieniuk, Andrzej Szpakowicz

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: This article presents a system for evaluating how well do video codecs perform in still image compression. As it is time consuming and tedious to directly configure and run the reference H.265 (HEVC) and H.264 (MPEG4 AVC) codecs, our tool makes experimenting easier and more efficient. The system provides a graphical user interface for conveniently and quickly preparing input files, for repeatedly running video codecs, and for analysing output data. In the background, there are algorithms for forming video sequences and configuration files, for batch executing an encoder, and for extracting information from output files. Our research was mainly aimed at speeding-up experiments related to compressing an image represented as a video sequence composed of its polyphase components. The system has allowed us to experimentally verify that, like its predecessor, the H.264, the HEVC standard needs modified entropy codes for effectively processing decimated images.

Keywords: video, image, coding, compression, codec, HEVC, H.264, MPEG4 AVC, polyphase, decomposition

1. Introduction

No so many years ago, it has been verified that video codecs can compress images better than advanced algorithms for coding still pictures, like JPEG2000 [1,2,3,6,7,10,11,19,22]. This has been shown for both the H.264 (MPEG4 AVC: Advanced Video Coding) standard [15], which currently dominates the multimedia market as the basis of the DVB-T terrestrial television and Blu-Ray discs, and the H.265 (HEVC: High Efficiency Video Coding) standard [17,18,21], which should be implemented soon for applications that involve UltraHD video. As a result, the HEVC standard has been extended so that it includes the Still Picture Profile [11], and a special file format called HEIF (High Efficiency Image File) have been standardized [4]. A light-weight variant of the reference HEVC encoder has also been developed [5], which is optimized for compressing a single image.

Advances in Computer Science Research, vol. 13, pp. 59-73, 2016.

In the majority of works, an image that has to be compressed using a video codec is represented as a single-frame video sequence. It is encoded only by means of intra prediction, while motion-compensated prediction (MCP) remains unused, even though it is advanced and effective. This apparently seems appropriate as motion estimation-compensation mechanisms are aimed at removing temporal correlation among frames, which has nothing to a single image. However, it has been shown in [12] that dependencies among pixels of an image can be converted into inter-frame correlations, which can be removed by means of motion estimation-compensation. It is only necessary to represent this image as a video sequence that comprises its polyphase components, which result from pixel decimation.

Our experiments, conducted using the H.264 reference software and presented in [12], have shown that the polyphase approach in general performs slightly worse than the single-frame counterpart, but this is not caused by a conceptual fault. The problem is more because the H.264 standard has not been optimized toward processing polyphase components, which have properties different than those of still images.

With the advent of the HEVC, it seems reasonable to check how well the new standard suits the polyphase approach. As the H.265 is more than the H.264 oriented toward still image compression, one can expect that it should effectively compress video sequences constructed from the results of the polyphase decomposition of a picture.

Appropriate tests can be performed using the reference HEVC codec, but experimenting is tedious and time consuming if not supported with a higher-level software. The reference software is difficult to configure, being controlled by more than a hundredth of parameters. An image must be converted into a video sequence before

Table 1. Analyzers of compressed video bitstreams

Name	Company	Web page
CodecVisa	Codecian	www.codecian.com
Zond 265	Solveigmm	http://www.solveigmm.com/en/products/zond
Elecard HEVC bitstream analyzer	Elecard	www.elecard.com
Gitl HEVC bitstream analyzer	Sun Yat-sen University	www.github.com/lheric/GitlHEVCAnalyzer
HEVC Analyzer for Rapid Prototyping (HARP)	Friedrich-Alexander-University	www.lms.lnt.de/HARP
Parabola Verifier	Parabola Research	http://www.parabolaresearch.com/verifier-hevc-validation.html
Intel Video Pro Analyzer (VPA)	Intel	https://software.intel.com/en-us/intel-video-pro-analyzer

running an encoder, and then it is necessary to restore the decoded picture from video frames, in order to evaluate the rate-quality tradeoff. Finally, an encoder must be run many times, for various images and combinations of parameter values.

Table 1 shows a list of the existing analyzers of video bitstreams. Such analyzers are aimed at helping codec developers, television engineers, and creators of multimedia contents. They are advanced but provide means mainly for analyzing existing bitstreams with respect to standard compliance, bit rate, and quality of decoded video. There is no specific support for processing still images, creating bitstreams, and comparing compression results for different settings of an encoder. So we had a reasonable motivation for developing a system tailored to our research needs.

2. The H.265 and H.264 standards for video coding

Video compression requires transforming picture (frame) sequences in such a way that two kinds of redundancy are removed. The first is temporal redundancy, which is related to correlations among pixels of adjacent frames. The second is spatial redundancy, which is a result of similarity of adjacent pixels in a single frame.

Both H.265 and H.264 standards specify hybrid decoders [14], which use both prediction and transforms to convert highly correlated pixels into uncorrelated values, which can be effectively entropy coded. In particular, temporal redundancy is removed by means of motion estimation/compensation among frames, while the spatial redundancy is removed by predicting pixels of a block to be coded from reconstructed pixels of blocks that can be decoded in advance [10]. In addition, DCT-like and Hadamard transforms are applied to prediction residuals [9].

Both standards only specify a bitstream format and explain how to interpret data so as to successfully decode a video sequence. They provide no ready-to-use recipes for implementing a codec which would be able to produce a format-compliant bitstream from video data and to efficiently reconstruct frames. However, both HEVC and H.264 standards are supported by reference software, which makes it easier to understand a standard as well as to develop and evaluate conformant programs, devices, and bitstreams.

The reference H.265 coded is called the HM (HEVC Test Model), while the reference H.264 software is called the JM (Joint Test Model). Their source codes are publicly available, in the C++ and C languages, respectively. Both projects have been maintained by the Fraunhofer Heinrich Hertz Institute (`www.hhi.fraunhofer.de`), as joint standardization projects of the ITU-T Video Coding Experts Group and ISO/IEC Moving Picture Experts Group.

The reference software are command-line applications. Before encoding a video sequence, about one hundredth of parameters must be set, which determine both effectiveness and speed of compression. This can be done entirely via the command line, but it is also possible to provide only one command-line parameter, with is the path to a configuration file, in which particular settings have been specified.

3. Converting images into video sequence by means of polyphase decomposition

Polyphase components of an image result from decimating it, by taking every second pixel vertically and horizontally, as explained in Fig. 1. As either even or odd pixels can be taken, polyphase decomposition results in 4 subimages. Polyphase decomposition and decimation are operations widely used in the research field of filter banks and transforms [20,8], as they allow for simplifying analysis and design of circuits and for efficiently implementing multirate systems.

Figure 2 shows the polyphase decomposition of a fragment of the well-known "Barbara" image. It is difficult to notice differences among polyphase components. They look like the same image, similarly as video frames. So it is intuitively justified to arrange them into a video sequence and to apply a video encoder to such data.

However, polyphase components have characteristics different from those of video frames. Variations among frames result from movements of the video camera and of objects it sees, while image contents (directions of edges and of intensity changes) determines which polyphase components are most similar in a given area. Moreover, video frames are smoother than polyphase components, which suffer from aliasing distortions, as no anti-aliasing pre-filtering precedes decimation.

Regardless of these issues, inter-prediction should work well on decimated images. For smooth regions of an image, a pixel of a polyphase component is usually the average of pixels of another component [13]. This suggests that bidirectional prediction should work better than unidirectional prediction, or that it is better to compress polyphase components as B-frames than as P-frames.

However, in order to employ bidirectional prediction, it is insufficient to straightforwardly arrange polyphase components into a 4-frame sequence. This is because the state-of-art video codecs cannot use a single frame as two reference frames for bidirectional prediction of another frame. A workaround is to put a reference frame twice into a video sequence, before and after the latter frame. Additionally, an encoder must be instructed how to handle so specific, redundant video sequences: in which order frames should be processed, and which kind of prediction should be ap-

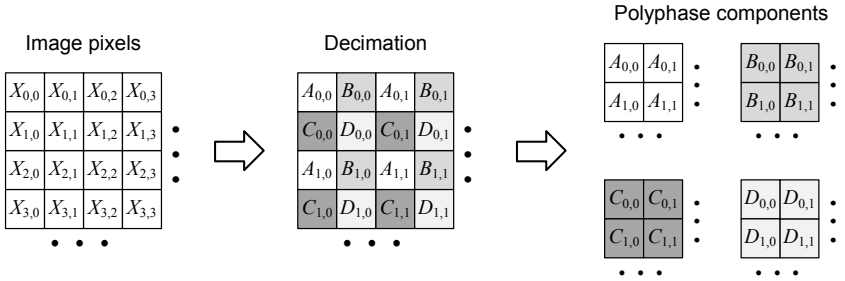


Fig. 1. The principle of polyphase decomposition.

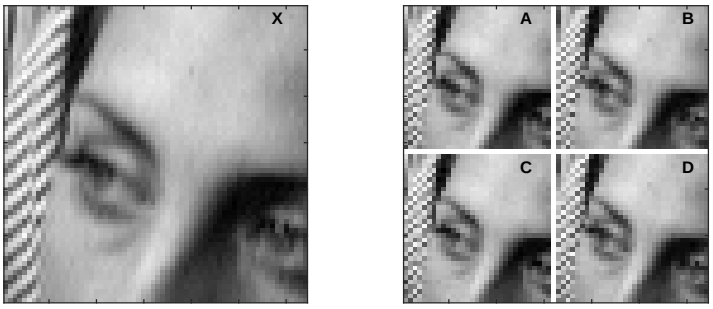


Fig. 2. The polyphase decomposition of a fragment of the "Barbara" image.

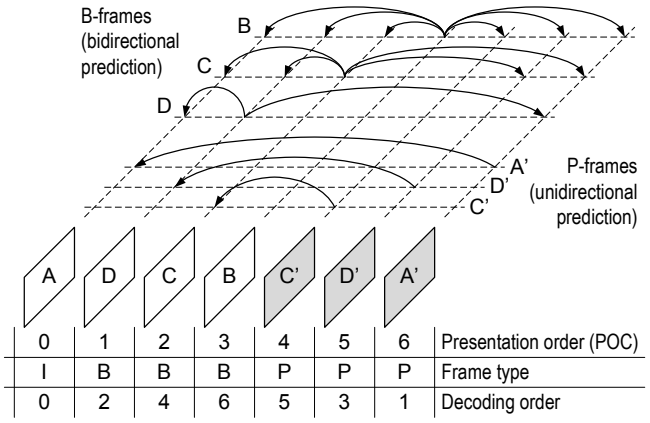


Fig. 3. The GOP structure used to compress polyphase components.

plied to a particular frame. For this purpose, it is necessary to carefully structure the group of pictures (GOP), which is one of the essential settings of a video encoder.

In our experiments, we used the GOP that is shown in Fig. 3, and video sequences in which three polyphase components occurred twice. Only the A component is encoded using intra predictions. The essential copies of the C, B, and D components are encoded using bidirectional prediction, as B-frames, and serve as references for encoding the extra frames as P-frames.

An encoder encodes such video sequences very efficiently, even thought it must process much more data than in the original image. In particular, it represents the extra frames using a negligible number of bits in the output stream, as demonstrated in Table 2.

Table 2. HEVC encoding statistics of video sequences composed of the polyphase components of the "Barbara" and "Finger" images

(a) "Barbara" image, 512×512 pixels

	Quantization parameter									
	QP = 9		QP = 17		QP = 25		QP = 33		QP = 41	
Frame	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**
A	298728	54.09	199416	46.15	111472	38.90	55416	32.62	19056	26.69
A’	104	54.09	96	46.15	96	38.90	88	32.62	88	26.69
D	248520	53.43	150752	45.67	62192	38.19	14208	31.21	1712	25.98
D’	112	53.43	112	45.67	96	38.19	96	31.21	96	25.98
C	197752	53.20	104408	45.65	25616	38.07	1944	31.94	224	26.63
C’	112	53.20	112	45.65	104	38.07	104	31.94	104	26.63
B	196728	53.20	102920	45.61	26736	38.00	1712	31.07	248	26.00

(b) "Finger" image, 256×256 pixels

	Quantization parameter									
	QP = 9		QP = 17		QP = 25		QP = 33		QP = 41	
Frame	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**	**Bit count**	**PSNR [dB]**
A	110376	54.35	84408	46.92	61840	38.97	36440	30.16	11728	22.19
A’	72	54.35	72	46.92	80	38.97	80	30.16	80	22.19
D	104304	53.67	77376	46.21	53424	37.87	27656	28.93	720	20.26
D’	88	53.67	88	46.21	80	37.87	80	28.93	80	20.26
C	100368	53.29	74256	46.26	50488	37.86	23936	28.74	144	20.62
C’	88	53.29	88	46.26	80	37.86	88	28.74	88	20.62
B	100104	53.47	74208	46.20	50384	37.80	24376	28.81	120	20.33

4. Functionalities and architecture of the system

The main functionality of our system is to provide a graphical user interface (GUI) for preparing configuration files for a video encoder, for running a reference software and evaluating its output data, and for managing and analyzing archived results of many experiments. Figure 4 shows three main windows that have been created for these purposes.

The window in Fig. 4a allows for selecting an image to be compressed, for choosing between HEVC and H.264 reference encoders, and for designating a combination of parameter values whose effect on compression results has to be tested. Before running an encoder, a configuration file is created based on a template, whose fragments are substituted in accordance with user's preferences. Also a data file with video frames is created in the planar YUV format [16], which is the only format accepted by the reference software.

The YUV format represents video frames without any compression. Row-by-row and column-by-column, for each pixel, values are listed that describe luminance, which is denoted as Y, as well as two, blue and red, chrominances, denoted by U and V, respectively. There are neither headers nor marks that would determine frame boundaries, and there is no indication of chroma subsampling. A software that has to process a YUV file must be informed by a user about the video resolution and color format of this file.

Both H.265 and H.264 reference encoders produce three files: a report, a bit stream of the encoded image, and the decoded video sequence in the YUV format. Our system converts the sequence back into an image file and archives all files in a newly-created directory for later usage. The window in Fig. 4b allows for investigating the results of a single run of the encoder. The original and decoded images can be compared visually, at the background of the compressed bit rate and objective measures of quality, the PSNR and MSE.

The window in Fig. 4c allows for extracting information from many directories with archived data. Data can be reviewed in a tabular form or plotted if this is preferred by the user. It is also possible to copy data to the system clipboard so as to export them to other applications.

The system has been implemented in Java, using the JDK 8 and Netbeans 8.1 development environments. It uses mainly the standard Java libraries, as the only exception is the "JFreeChart" library for drawing plots. The GUI is based on the Java Swing API. As to the reference software, the HM version 16.2 and the JM version 19.0 were used in our research, compiled for 32-bit Windows operating systems.

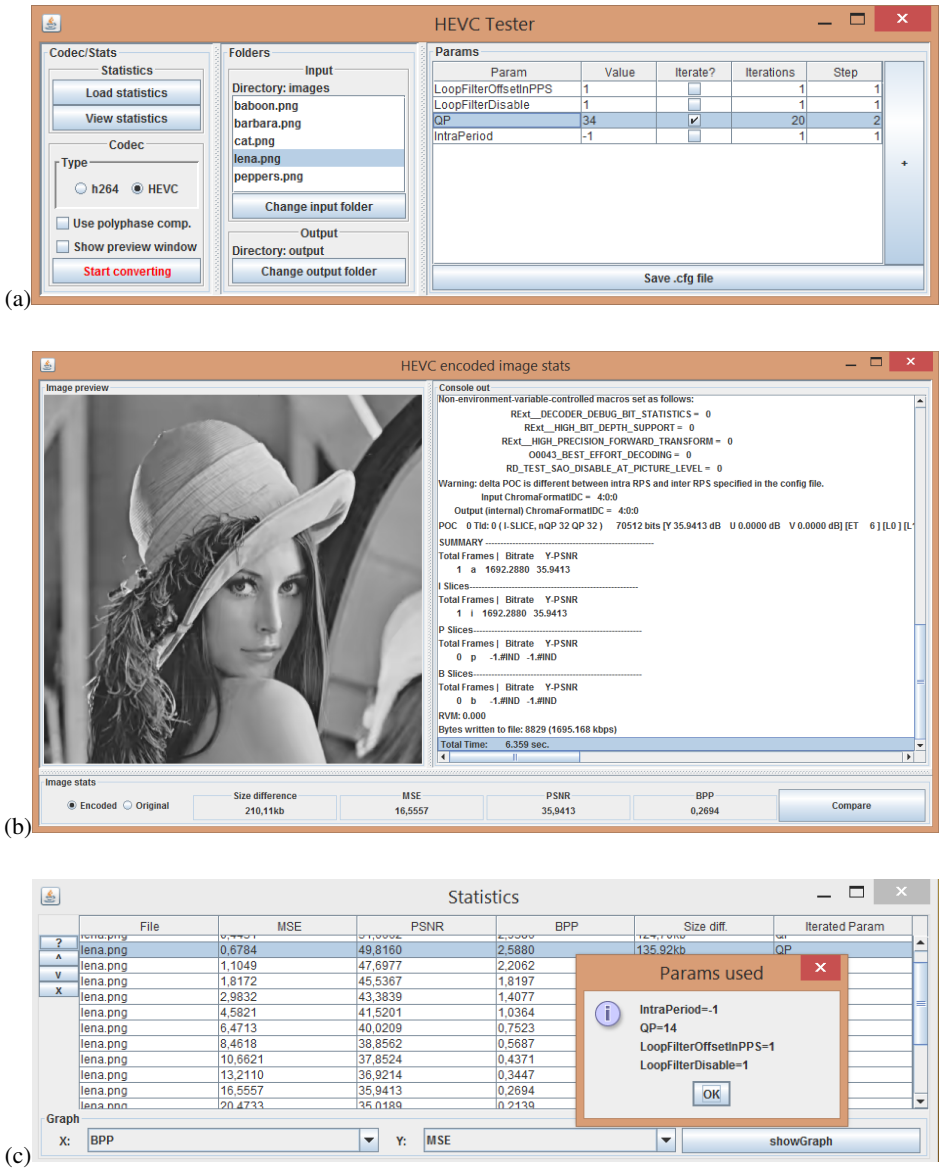


Fig. 4. The graphical user interface for evaluating performance of video codecs: (a) the window for configuring experiments, (b) the window for evaluating results of a single run of an encoder, and (c) the window for reviewing archived results and determining trends in data.

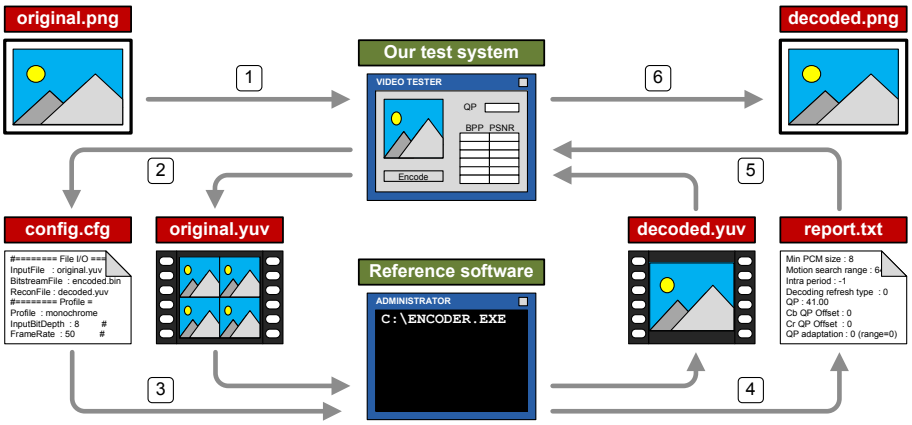


Fig. 5. The dataflow in our system for testing video codecs.

The algorithms and data structures behind the windows are not trivial, even though our application is essentially a wrapper to the reference implementations of the HEVC and H.264 standards. They cooperate in accordance with the dataflow shown in Fig. 5, exchanging data via files.

The source code of our system consists of about 3000 lines and about 20 classes, about half of which are related to video encoding. In particular, the main class, "CodecManager", implements interfaces to the reference encoders and means for managing their input and output files. The main data structures have been encapsulated in the "GrayscaleMedia" class, which is responsible for storing both original image and corresponding video sequence, and allows for converting files into 2D arrays of numbers, which represent image pixels. Another important class is "ImageUtils", which provides functions for converting images between disk files and numerical arrays. In particular, it provides methods for converting a pixel array into its polyphase components.

Video encoding is realized by external executables, which are launched by the system in accordance with user preferences, determined using the GUI. Before launching an encoder chosen by the user, the system performs two tasks. Firstly, it converts a given image into a YUV file, which may be a seven- or single-frame video sequence, depending on whether the polyphase approach has to be tested. The second step is to analyze GUI controls and to produce a configuration file for the encoder.

After launching a reference software, our system monitors the progress in encoding, collects output data, and saves results for later analysis. In case of successful execution, the encoders produce two main files: a bitstream with encoded informa-

tion (a file with the ".hevc" or ".h264" extension) and the decoded video sequence in the YUV format. Our system converts the latter into a PNG file, so as to allow the decoded image to be evaluated visually and compared to the original.

5. Representative experimental results

The system we have developed allows for easily gathering various and thorough experimental results. Firstly, data can be obtained that are necessary to plot rate-quality curves like those in Fig. 6. Secondly, details about how a particular polyphase component has been encoded can be extracted from encoder reports, in order to create tables like Table 2. Finally, original and reconstructed images can be compared visually so as to evaluate subjective quality and compression artifacts, whose examples are shown in Fig. 7.

In general, our results obtained for the HEVC conform with those presented in [12] for the H.264 standard. Coding a single frame is usually more effective than coding a sequence of polyphase components, but the latter method performs only slightly worse and sometimes works equally well. This suggests that the polyphase approach should be successful after adapting the reference software to processing alias-contaminated images, especially by adjusting entropy codewords to a limited set of motion vectors and to a reduced search range for motion estimation.

Figure 7 shows artifacts caused by compressing an image as a single frame and as a sequence of polyphase components. It can be seen that one cannot depend on objective evaluation of decoded images if he would like to compare the approaches. The difference between images seems considerable if measured in terms of the PSNR, but they are more or less similar visually. For compressing polyphase components, the reconstructed image looks even better, as it is smoother, and quantization noise is less noticeable, even though the objective measure suggests the opposite.

Our experimental data also confirm the known fact that the HEVC performs better than the H.264, regardless of whether an image is represented as a single frame or as a sequence of polyphase components.

6. Summary

The primary aim of our research has been achieved. We have developed a functional system that is really helpful in configuring and running the reference video codecs, and then in analyzing their output data. In particular, the system has allowed us to thoroughly test whether the HEVC reference software is able to efficiently compress images represented as multi-frame video sequences using polyphase decomposition.

Unfortunately, even equipped with such a handy tool, we were unable to tune settings of the HEVC encoder so as to prove that it is better to compress a sequence of polyphase components than to encode an image as a single frame. In general, the latter approach performs slightly better, but sometimes both techniques work equally well. This suggests that only a video codec with carefully optimized internals, especially entropy codes, would be able to handle decimated images effectively. We have recently studied the heart of such codec, a wavelet-like transform inspired by motion estimation/compensation [13].

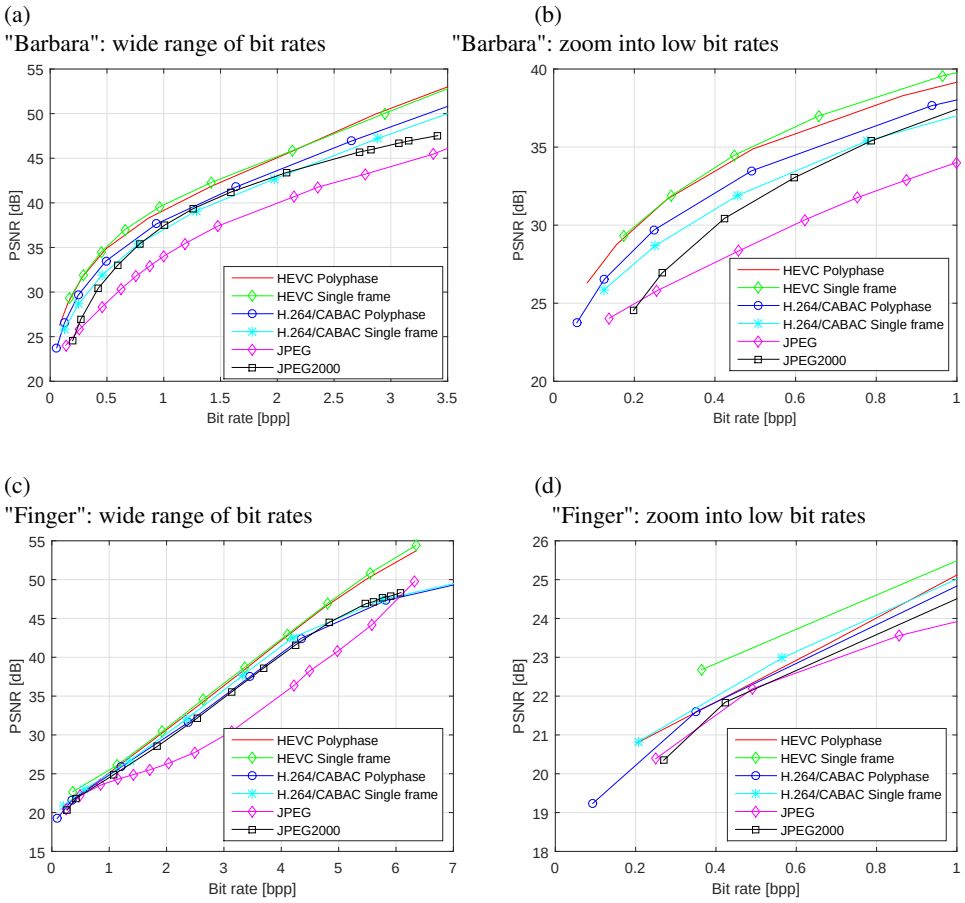


Fig. 6. Effectiveness of various standards in coding the ”Barbara” (512×512 pixels) and ”Finger” (256×256 pixels) images.

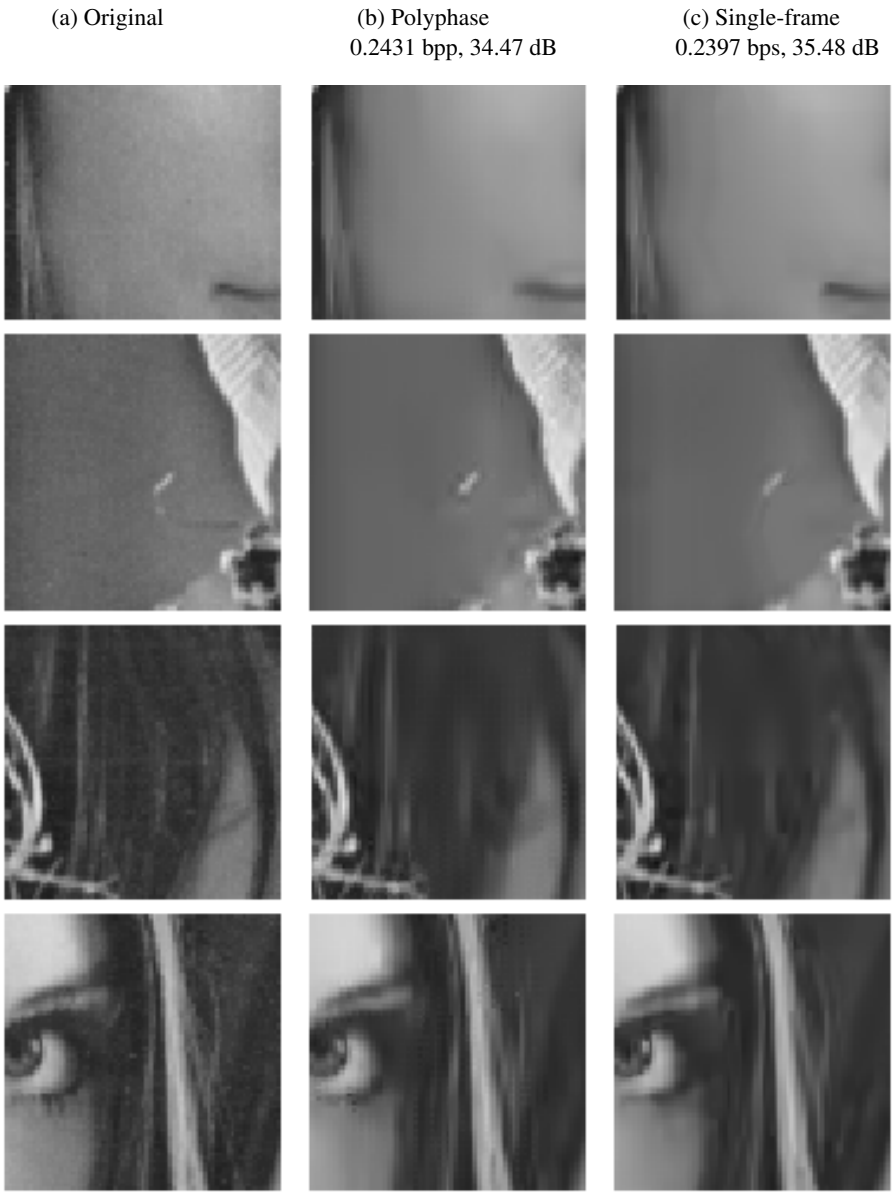


Fig. 7. Artifacts in images compressed using the reference HEVC codec: (a) fragments of the original "Lena" image, (b) fragments reconstructed from a bitstream obtained using the polyphase approach, and (c) fragments reconstructed from a bitstream obtained by encoding a single frame.

Acknowledgment

This work was supported by the Polish National Science Centre under Decision No. DEC-2012/07/D/ST6/02454.

References

[1] A. Al, B. P. Rao, S. S. Kudva, S. Babu, S. David, and . Ajit V. Rao. Quality and complexity comparison of H.264 intra mode with JPEG2000 and JPEG. In *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pages 525–528, Singapore, 24-27 Oct. 2004.

[2] Q. Cai, L. Song, G. Li, and N. Ling. Lossy and lossless intra coding performance evaluation: HEVC, H.264/AVC, JPEG 2000 and JPEG LS. In *Proc. Asia-Pacific Signal & Information Process. Association Annual Summit and Conf. (APSIPA ASC)*, pages 1–9, Hollywood, CA, 3-6 Dec. 2012.

[3] P. Hanhart, M. Rerabek, P. Korshunov, and T. Ebrahimi. Subjective evaluation of HEVC intra coding for still image compression. In *Proc. 7th Int. Workshop Video Process. Quality Metrics Consumer Electronics (VPQM)*, pages 58–63, Scottsdale, Arizona, USA, Jan. 30-Feb. 1 2013.

[4] M. M. Hannuksela, J. Lainema, and V. K. M. Vadakital. The High Efficiency Image File format standard. *IEEE Signal Process. Mag.*, 32(4):150–156, July 2015.

[5] F. Liang, X. Peng, and J. Xu. A light-weight HEVC encoder for image coding. In *Proc. Visual Communications Image Process. (VCIP)*, pages 1–5, Kuching, Malaysia, 17–20 Nov. 2013.

[6] D. Marpe, V. George, H. Cycon, and K. Barthel. Performance evaluation of Motion-JPEG2000 in comparison with H.264/AVC operated in pure intra coding mode. *Proc. SPIE*, 5266 (Wavelet Applications in Industrial Process.):129–137, 2004.

[7] D. Marpe, V. George, and T. Wiegand. Performance comparison of intra-only H.264/AVC HP and JPEG2000 for a set of monochrome ISO/IEC test images: Contribution JVT ISO/IEC MPEG and ITU-T VCEG. Technical Report JVT M-014, Oct. 2004.

[8] L. Milić. *Multirate filtering for digital signal processing: MATLAB applications*. Information Science Reference / IGI Global, Hershey, PA, 2009.

[9] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand. Transform coding techniques in HEVC. *IEEE J. Sel. Topics Signal Process.*, 7(6):978–989, Dec. 2013.

[10] T. Nguyen and D. Marpe. Performance analysis of HEVC-based intra coding for still image compression. In *Proc. Picture Coding Symp. (PCS)*, pages 233–236, Cracow, Poland, 7-9 May 2012.

[11] T. Nguyen and D. Marpe. Objective performance evaluation of the HEVC Main Still Picture Profile. *IEEE Trans. Circuits Syst. Video Technol.*, 25(5):790–797, May 2015.

[12] M. Parfieniuk. Polyphase components of an image as video frames: a way to code still images using H.264. In *Proc. 2012 Picture Coding Symposium (PCS)*, pages 189–192, Cracow, Poland, 7-9 May 2012.

[13] M. Parfieniuk. *Wavelet-Like Lifting-Based Transform for Decomposing Images in Accordance with the Inter-prediction Principles of Video Coding*, volume 9280 of *Lecture Notes in Computer Science (LNCS)*, pages 162–171. Springer, 2015.

[14] K. R. Rao, D. N. Kim, and J. J. Hwang. *Video coding standards: AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1*. Springer, 2014.

[15] I. E. G. Richardson. *The H.264 advanced video compression standard*. Wiley, 2 edition, 2010.

[16] G. Sullivan and S. Estrop. Recommended 8xbit YUV formats for video rendering. Technical report, Microsoft, 2008.

[17] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) standard. *IEEE Trans. Circuits Syst. Video Technol.*, 22(12):1649–1668, Dec. 2012.

[18] V. Sze, M. Budagavi, and G. J. Sullivan, editors. *High Efficiency Video Coding (HEVC): algorithms and architectures*. Springer, 2014.

[19] T. Tran, L. Liu, and P. Topiwala. Performance comparison of leading image codecs: H.264/AVC Intra, JPEG2000, and Microsoft HD Photo. *Proc. SPIE*, 6696 (Applications of Digital Image Processing XXX):66960B, 2007. doi:10.1117/12.775472.

[20] P. P. Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proc. IEEE*, 78(1):56–93, Jan. 1990.

[21] M. Wien. *High Efficiency Video Coding: coding tools and specification*. Springer, 2015.

[22] A. Zaghetto and R. L. de Queiroz. Segmentation-driven compound document coding based on H.264/AVC-Intra. *IEEE Trans. Image Process.*, 16(7):1755–1760, July 2007.

SYSTEM DO BADANIA SKUTECZNOŚCI KODEKÓW WIDEO W KODOWANIU OBRAZU

Streszczenie Artykuł prezentuje system do badania sprawności kodeków wideo w kompresowaniu obrazu. Ponieważ bezpośrednie konfigurowanie i uruchamianie referencyjnych kodeków H.265 (HEVC) i H.264 (MPEG4 AVC) jest pracą czasochłonną i żmudną, omawiane narzędzie pozwala przeprowadzać eksperymenty łatwiej i szybciej. System jest wyposażony w graficzny interfejs użytkownika do wygodnego i sprawnego przygotowywania plików wejściowych, do wielokrotnego uruchamiania kodeków wideo i do analizowania danych wyjściowych. Interfejs opiera się na podprogramach do formowania sekwencji wideo i plików konfiguracyjnych, do wsadowego uruchamiania koderów i do wydobywania informacji z plików wynikowych. Głównym celem pracy nad systemem było przyśpieszenie eksperymentów nad kodowaniem obrazu reprezentowanego jako sekwencja wideo złożona z jego składowych polifazowych. System umożliwił eksperymentalne sprawdzenie, że podobnie jak H.264, standard HEVC może skutecznie przetwarzać obrazy zdecymowane, ale wymaga to zoptymalizowania kodów entropijnych.

Słowa kluczowe: wideo, obraz, kodowanie, kompresja, kodek, HEVC, H.264, dekompozycja/składowa polifazowa

SCREEN KEYBOARD ARRANGEMENT OPTIMIZATION FOR POLISH LANGUAGE

Michał Wołosik, Marek Tabędzki

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: The aim of this work was to find screen keyboard arrangement optimal for Polish language. This study adopted a standard shape and organization of the keyboard, the task is therefore only for identifying the best permutations of keys. Only the alphabet keys and five selected punctuation marks were permutated. In order to accomplish this task, machine learning methods were used: genetic algorithms and simulated annealing. Fitness function is based on two literary works and one technical document. The following criteria were used: of distance, the writing direction and row weights. The application prepared for the experiments was developed in Java. The paper describes used algorithms and obtained results. Best found arrangement would shorten the time to input sample texts by about 30% (assuming adequate accustom of the new layout by the writer).

Keywords: keyboard arrangement problem, genetic algorithms, simulated annealing

1. Introduction

The most popular input device, that among other things is used to enter the text, is a computer keyboard. Common alternative for a classic physical keyboard is its screen counterpart, that usually can be displayed on the monitor and managed with a mouse. Almost every country has its own key arrangement which is optimal for the local language. What's interesting, for some reason it's different in Poland so this subject is worth looking into and perhaps there is something that could be done about it.

The goal of this work is to develop keyboard arrangement that would be optimal for Polish language. The task is connected to minimizing lengths of moves that need to be performed by mouse pointer in order to rewrite given text. There are also other less important criteria that might be taken under consideration that define difficulty of typing the text. In this work keyboard that is being optimized is of standard rectangular shape and the same set of keys that are present in QWERTY "programmer" version arrangement.

Advances in Computer Science Research, vol. 13, pp. 75-93, 2016.

In order to achieve mentioned goal it is necessary to use machine learning methods like evolutionary algorithms or simmulated annealing, which would train keyboard arrangement with sample text blocks that are well representants of given language characteristics. For instance, it could be some classic novels. Polish texts that have been used in this work are among the others: "With fire and sword" by Henryk Sienkiewicz (*Ogniem i mieczem*) and "Sir Thaddeus" by Adam Mickiewicz (*Pan Tadeusz*).

In the next section you can find a short review of typical approaches and algorithms that are usually used to solve keyboard arrangement problem (KAP). Then, the details about permutational representation of the chromosome are explained. Following section is dedicated to the topic of multicriterial evaluation function. Then, the optimization methods that were implemented and used for experiments are described. Final sections contain results of the tests, and present and describe optimized keyboard arrangement – achieved goal of this work. It is followed by a short summary of what have been done and some loose conclusions.

2. KAP

In Poland, the most popular arrangement is called by its first letters of upper row keys – QWERTY keyboard in so called "programmer" version, or QWERTZ in older devices. In French-speaking countries most commonly used layout is AZERTY [16] while in English-speaking countries many kinds of QWERTY and its rival DVORAC [2, 4] can be encountered.

Keyboard arrangement problem (KAP) for certain language is a task in which evolutionary algorithms are usually applied. The key issue here is to determine appropriate individual representation as chromosome. One way of doing this, what was proposed in [6], is permutation. Next thing, that has to be established is an evaluation function, which would measure optimization level of current keyboard arrangement. This requires answering the question what ergonomic factors should be considered. It mostly depends on input that keyboard operates on. Wheter all fingers of both hands can be used, only two thumbs like in [9] or maybe text can be typed with just one pointer.

In literature, in conjunction with problem that is being discussed in this work, beside usage of genetic algorithms, applications of other methods can also be found. Very common are simulated annealing [1, 3, 8] and ant colony algorithms [14].

2.1 Chromosome representation and typing simulation

Experiments were conducted for PC screen keyboard with standard rectangular shape and the same set of keys that can be found in QWERTY "programmer" arrangement. Similarly how it has been done by Julstrom, Goettl and Brugh in [6], chromosome were encoded as permutation of keys. Index of each element is connected to its position on keyboard. Permutation includes 26 keys that represents latin alphabet characters and 5 punctuation keys (in [6] there was only 4). Size of problem space to search is: $31! \approx 8.22 \times 10^{33}$

Application created for experiments, in order to rewrite exemplary text block coded in UTF-8, is able to use keys that are highlighted in Figure 1. Light grey ones are part of chromosome and the darker ones have fixed positions.

Fig. 1. Movable and fixed keys

Each following recognizable character has to be mapped on a proper keys sequence. For instance for digit or small letter it will simply be one key that represents this digit or letter, and for capital letter it would be sequence of Shift key and this letter key.

3. Evaluation function for SFKL problem

Experiments were conducted for keyboard that is controlled with mouse pointer. In literature [3, 8] this approach is referred as single-finger keyboard layout problem (SFKL). This is important regarding cost calculation and ergonomics of typing. Chromosome evaluation function should estimate difficulty of rewriting given text block. Therefore, its global minimum has to be found.

3.1 Text blocks used for experiments

Texts that are going to be used to train keyboard arrangements should reflect characteristics of given language as well as possible. Set of three text blocks[1], that were used in this work contains following files: ogniem-i-mieczem-dwa-tomy.txt (With fire and sword), which includes 1719477 characters, pan-tadeusz.txt (Sir Thaddeus) with 503759 characters and wytyczne_informatyka-15.txt (Guidelines and advices on diploma work preparation) with 23850 characters. Chart in Figure 2 presents averaged probabilities of each key occurrence during rewriting of mentioned texts, and Figure 3 shows probabilities of pair of keys occurrences (one key after another). Charts do not include keys that are not part of chromosome.

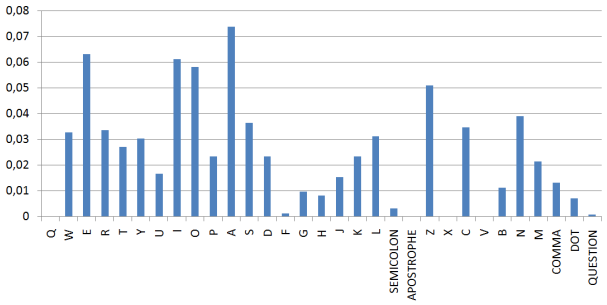


Fig. 2. Keys probabilities

Buttons which among the others refer to Polish diacritical characters have high frequency. For example key A has to be clicked each time when ‘a’ or ‘ą’ character occurs in the text. Therefore characteristics of keys typing frequency during writing Polish text with a keyboard should not be confused with characteristics of characters occurrences frequency in this language.

Four most probable pairs of keys that are clicked one after another are: $\langle \mathrm{I}, \mathrm{E} \rangle$, $\langle \mathrm{N}, \mathrm{I} \rangle$, $\langle \mathrm{O}, \mathrm{W} \rangle$, $\langle \mathrm{Z}, \mathrm{E} \rangle$ with probabilities of 0.01943, 0.01371, 0.00987, 0.0089 respectively.

[1] Texts “With fire and sword” by H. Sienkiewicz and “Sir Thaddeus” by A. Mickiewicz have been downloaded from `https://wolnelektury.pl/`. The third text block is a dokument that contains guidelines and advices on diploma work preparation which is available on `http://wi.pb.edu.pl/`.

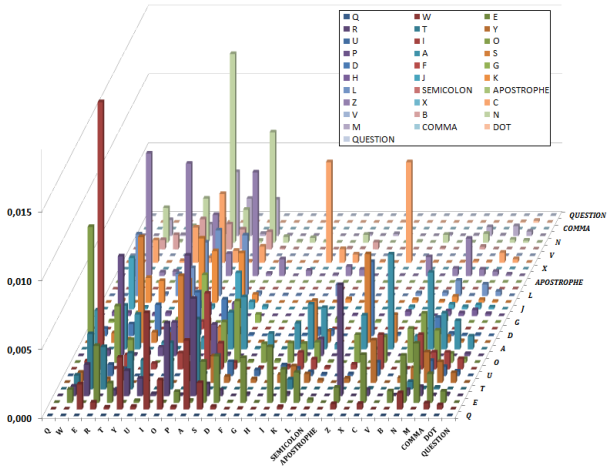


Fig. 3. Probabilities of pair of keys, first in row

For each of analyzed text blocks, proportions of concrete keys and pairs of keys frequencies are almost identical. In all cases key A is the most commonly clicked button and pair $\langle \mathrm{I}, \mathrm{E} \rangle$ is the most frequent.

3.2 Multicriterial evaluation function

Developed fitness function is composed of 3 different criteria, which are usually used for single-finger keyboard optimization problem. They are described below.

Distances criterion considers length of path that mouse pointer needs to travel by moving from previous to next keys. In [8] has pointed out, that "the s-finger keyboard layout problem can be modeled in terms of the Quadratic Assignment Problem (QAP)", which involves assigning n facilities to n locations in optimal way. In [3, 8] researchers mentioned two square matrixes: flow matrix, in which cell c_{ij} contains probability of flow between objects i and j, and also distances matrix in which d_{kl} is euclidean distance between positions k and l. Whole cost of rewriting the text for this criterion given in [8] and [3] is the sum of multiplications of cells corresponding to the same pairs of keys from both matrixes. Taking that n is the quantity of keys and $p(x)$ is the index of key x in permutation, formula looks like this:

$$D = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} d_{p(i)p(j)} \tag{1}$$

Criterion of hit direction – in [8] researchers concluded, that it is better if the direction of mouse pointer movement is corresponding to the direction of natural writing. For Polish language it is from left to right. The pair of keys I and E has the highest probability (0.01943) of occurrence one after another. Therefore, expected situation after optimization process for this criterion is arrangement where key corresponding to letter I is on the left from key corresponding to letter E. Formula below shows the penalty that is added:

$$S = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} v_{ij} \tag{2}$$

where: $v_{ij} = \begin{cases} 1, & \text{if } x_i - x_j > 0 \\ 0, & \text{otherwise} \end{cases}$

Row weights – the last criterion proposed in publication [8]. It is based on assumption that more frequent keys should be placed in rows which are more convenient to use. According to the most researchers opinion usage of the middle row is more comfortable than the other two, so the most frequent keys should be put in there. Nevertheless such solution would cause problems regarding distances criterion. Keys that correspond to Polish diacritical characters are the most frequent ones and putting them in the middle row their distances from fixed Alt key (which is below the bottom row) would be increased. Therefore it has been decided that penalties would be given for placing frequent keys in upper row only:

$$R = \sum_{i=1}^{n} f_i w_i \tag{3}$$

where:

- f_i – probability of pressing key i
- $w_i = \begin{cases} 2, & \text{if key } i \text{ is placed in the upper row} \\ 1, & \text{otherwise} \end{cases}$

The final form of the fitness function which is based on previously mentioned criteria:

$$F = aD + bS + cR \tag{4}$$

where: a, b and c are factors (weights of importance) of each criteria.

For cosmetic reason, these factors should sum up to 1. Optimal weights of criteria importance were found experimentally in the analogical way to what has been

described in [8]. Test keyboard arrangement optimization algorithms has been run iteratively, each time for different set of discussed factors, starting from $a = 1$ after which factors b and c were increased by small values subtracted from a. Expected results for each criterion were met with $a = 0.8$, $b = 0.15$ and $c = 0.05$. Fitness function values will later on be presented in proportion to the fitness value of QWERTY programmer arrangement.

3.3 Typing time estimation, Fitts's law

Typing time is very important in keyboard optimization evaluation. Theoretically the distances criterion could be substituted with it. In [10] many different Fitts's law – based methods of mouse pointer travel time estimation (form one key to another) were described. Taking that A is the distance between centers of source and the target keys and W is the width of the target key, this is the formula that were used in this work:

$$T = \begin{cases} \frac{10}{49} \log_2 \left(\frac{A}{W} + 1 \right), & \text{if } A > 0 \\ 0.127, & \text{if } A = 0 \end{cases} \tag{5}$$

4. Used optimization algorithms

Within this work, experiments were conducted with genetic algorithms with and without adaptive methods of mutation operator. Simulated annealing algorithm also has been used.

4.1 Genetic algorithms

Usage of evolutionary algorithms is very common in context of KAP. These methods operate on some population of chromosomes simultaneously. In each iteration (generation) to selected parents individuals from population, crossover and mutation operators are applied. Children population is being created in this way. Next step involves so called succession which is replacing some old individuals from actual population with the new ones. These steps are repeated till the specified stop condition is satisfied, with hope that the next generations will "produce" better specimens. Due to probabilistic nature of this algorithms they should be executed many times.

Selection and succession operators Some basic models of succession and selection operators are described by Wierzchoń in [15]. For experiments in this work selection SUS and rank succession were used.

Proportional selection, which extension is SUS [12], chooses individuals to parents population with probability $P_i = f_i/F$, where F is the sum fitness of whole population and f_i is the fitness of i-th individual. It should be remembered that according to chosen take on the problem, fitness function determines the cost, so before performing selection this function need to be transformed for each specimen: $f_i' = f_{AVG}/f_i$, where f_i' is the new fitness and f_{AVG} is average fitness.

SUS (Stochastic Universal Sampling) instead of using only one selection point on fitness sum space, it uses N equally spaced points like this, where N is a quantity of parents population. Method randomly chooses only one value from range $[0, 1/N]$, which becomes the first point. After that it generates next $N-1$ points of selection, starting from the first, incrementing value by interval $1/N$. Finally, N individuals addressed by these points being placed in their fitness range, are chosen to parents population. Figure 4 shows example situation where population size is 10 and $N = 6$.

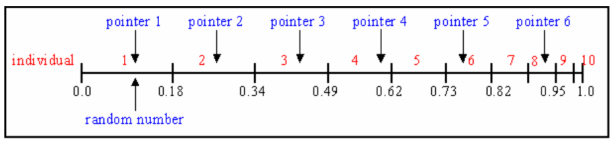


Fig. 4. Stochastic Universal Sampling, source: [12]

Thanks to this strategy there is lesser probability of premature convergence of evolutionary algorithm, because there is a big chance for weaker individuals to be chosen to parent population. In this way SUS maintains a large diversity of population.

Rank succession on the other hand, which was used for experiments, sorts population by individuals fitness values with descending order. For each specimen concrete ranks are being assigned – the first gets value 1, next one gets previous value increased by 1, etc. Ranks play the role of a new fitness function values. Following replacement goes on by the means of roulette wheel method. Individuals for replacement are chosen with probability $P_i = f_i/F$. The use of ranks makes it easier to distinguish better individuals from worse ones in population of low diversity.

Crossover and mutation operators The role of crossover is to exchange features between particular individuals – children are created by exchanging parents genes. Mutation, by contrast, makes small modifications of single individuals [15] and is usually performed with very small probability.

In experiments, for solutions coded as permutations, following crossover operators were used:

- UX (*Uniform crossover*) – at first, it iterates through both parents and copies the i-th element from the first or second one (randomly selected) on the i-th position of child chromosome. If on this position both elements from both parents are already in the child, an empty space is left. After that, all blank places are randomly filled with the rest available elements [6, 8, 11].
- PMX (*Partially mapped crossover*) – in detail, method was described in [11].
- OX (*Order crossover*) – at first a random segment is being copied from the first parent to a child. After that, missing elements are taken from second parent, starting right after the area of chosen segment and then continuing from the beginning of permutation [11].

In experiments, for chromosomes coded as permutations, following mutation operators were used:

- SWAP – it is based on swapping places of two randomly selected keys of permutation [8, 11]
- SCRAMBLE – elements on k randomly selected positions are randomly permuted (within selected positions only) [11]
- INSERTION – random element is put on a new randomly selected position [11]. In [8] this mutation has been used In slightly different form.
- SHIFT – random segment of permutation is moved to a different location [13].
- INVERSION – the order of random segment elements is reversed [11].

Diversity measurements of population The need of taking measurements of population diversity has many reasons of diagnostic nature. As [5] states, diversity measurements can be divided on: genotypic – which informs about differences between chromosomes, and phenotypic – that deals with differences in fitness values of population individuals. The first ones are usually more time consuming and the second ones are less accurate.

In this work, genotypic measure that were computed during the experiments is modified version of linear diversity measure for permutations that has $O(n)$ time complexity. Its original version was designed by Dudek M. and presented in [5].

$$LMRDP_{POP} = \frac{1}{n+1}\left(\frac{PZ_{dif}-1}{n-1} + \sum_{i=1}^{n}\frac{LP_i-2}{n-3}\right), \quad LMRDP_{POP} \in [0,1] \qquad (6)$$

where:

- n – number of nodes – problem size (for example, number of keys in chromosome of coded keyboard)
- LP_i – number of different connections from node i, or 2 if all are identical
- PZ_{dif} – number of different first nodes of permutations within a population, possible values are in range $[1,n]$

Diversity in population is equal to 0 if all chromosomes are identical. On the other hand, the measure takes the value of 1 when starting nodes of permutations do not repeat and each node has connection to every other nodes within a population permutations. Maximal number of such connections for one node is $n-1$.

4.2 Adaptive methods for mutation

Publication [13] describes mechanisms, inter alia, AOC and MOS, which are based on usage of many mutation operators during one evolutionary algorithm. It suppose to increase the chance of leaving the local optimum.

AOC (*Adaptive operator cycling*) – mechanism of exchange is run periodically, after each execution of specific number of generations – so called adaptive period, and at least one mutation has been performed since previous period has finished. Probability of operator exchange is based on success rate factor r_s, which is the ratio of success mutations to all mutations of current adaptive period. Formula looks like this:

$$P_{OC} = \frac{1}{2}\left(1+\cos\left(r_s \cdot \pi\right)\right) \qquad (7)$$

AOC2 slightly differs from metod described above. Attempt to exchange mutation operator occurs when in actual adaptive period all mutation were not successful.

MOS (*Mutation operators statstics*) – at the end of each adaptive period, this metod recalculates success rates r_{si} for each i-th mutation operator. Calculation is based on previous adaptive period. In each generation, with some r_{si} based probability, a choice is being made which mutation operator should be use on given parent individual. In order to give a chance for mutation with r_{si} equal to 0 to be selected,

in [13] usage of b_i parameter was proposed. It is calculated at the end of each adaptive period for each i-th mutation.

$$b_i = (\text{rand}(0;1) - 0.5) \cdot a \tag{8}$$

$$P_{chi} = \begin{cases} \frac{\max(0;r_{si}+b_i)}{\sum_{j=1}^{n}\max(0;r_{sj}+b_j)}, & \text{for } \sum_{j=1}^{n}\max(0;r_{sj}+b_j) \neq 0 \\ \frac{1}{n}, & \text{for } \sum_{j=1}^{n}\max(0;r_{sj}+b_j) = 0 \end{cases} \tag{9}$$

where:

- $\text{rand}(0;1)$ – random value from $[0,1]$ uniform distribution
- a – parameter which was found experimentally in [13] as 0.02
- P_{chi} – probability of i-th mutation selection
- n – mutations quantity

4.3 Simulated annealing algorithm

One of the approaches used in the keyboard arrangement problem, is simulated annealing algorithm and hybrid solutions combining it with genetic algorithms [1,3,8]. This algorithm operates on one individual, every step trying to replace it with a neighbor that is created using chosen mutation operator. The method introduces a parameter called the temperature, which is gradually reduced at a certain number of iterations (epoch). At higher temperature the probability that an individual will be replaced by a worse neighbor is higher.

Other parameters of simulated annealing approach are: α – thermal melting point, i.e. the rate at which the temperature decreases (usually in the range 0.95–0.98), L_{min} – initial length of the epoch, i.e. the number of iterations at the same temperature, L_{max} – maximum epoch length, β – the rate at which the length of the epoch grows.

This research is based on acceptance probability (the probability of substituting the current individual with the one obtained as a result of mutation) taken from [1] and [3]. Besides the temperature, it also takes into account the difference in the fitness values of the two individuals. Assuming that f is the fitness of the current individual, f' is the fitness of considered neighbor, and T is the current temperature, the formula looks as follows:

$$P_{subst} = \begin{cases} 1, & \text{if } f' < f \\ e^{-\frac{f'-f}{T}}, & \text{if } f' \leq f \end{cases} \tag{10}$$

5. Experiments and result analysis

Application for testing purposes was written in Java (JDK ver. 1.7) using NetBeans IDE (ver. 8.0.2). Some parts of the algorithms, including the fitness function calculation, has been parallelized. Tests were performed on a machine with quad-core processor Intel Core i7-2630QM 2.00GHz. Tests were conducted for a total of 15 optimization approaches: GA with 9 combinations of crossover and mutation operators, 3 adaptive mutation operators without crossover, and simulated annealing with 3 different mutation operators. Each of the approaches was run 10 times for each of the three text blocks, and the results were averaged.

5.1 Genetic algorithms

In experiments, the following parameters of GA were adopted: population size of 200, SUS selection with parental pool of 50, and rank-based selection. Crossover probability was 0.9 and mutation probability was 0.15. Termination condition: reaching 2,000 generations. The column "Operators" of the following table contains information about which crossover and mutation operators were used. For the SCRAMBLE operator, parameter k was equal to 3.

Table 1. Results for genetic algorithms

Operators	Best initial	Average initial	Best final (MBF)	Average final	Gain	Running time [s]
UX, SWAP	0.86457	0.99367	0.69951	0.70412	19%	24.26917
UX, SCRAMBLE	0.86611	0.99456	0.69882	0.69996	19%	24.80797
UX, INSERTION	0.86581	0.99394	0.69882	0.73335	19%	26.65170
PMX, SWAP	0.85778	0.99408	0.69500	0.70019	19%	25.11970
PMX, SCRAMBLE	0.86669	0.99438	0.69537	**0.69747**	20%	24.97160
PMX, INSERTION	0.86321	0.99415	**0.69460**	0.70439	20%	24.76830
OX, SWAP	0.86522	0.99287	0.70184	0.71601	19%	25.34267
OX, SCRAMBLE	0.86923	0.99431	0.70205	0.70547	19%	24.79710
OX, INSERTION	0.85807	0.99361	0.70567	0.72059	18%	25.32400

There were no major differences in average values of the reached fitting. Best MBF of 0.6946 was obtained using PMX crossover and INSERTION mutation. However, the top average of the final fitness (of 0.69747) was for the SCRAMBLE mutation with the same PMX crossover operator. The worst results were obtained with OX crossover, for which none of the final fitness drops below 0.7.

The sharp decline in the cost function took place in the initial phase of the algorithm – the first 250 generations, and then decreased gently. Population converged rapidly to the same time. After about 250 generations, the value LMRDP stabilized more or less on the value of 0.3.

5.2 Genetic algorithms with adaptive mutation

In experiments using adaptive mutation operator, an adaptation period was of 200 iterations, and the algorithm was stopped after $4,000$ iterations. Adaptations AOC and AOC2 were using only SWAP and INSERT mutation operators, while MOS – all of listed in section 4.1. For SCRAMBLE mutation value k was set to 3.

Table 2. Results for genetic algorithms with adaptive mutation

Adaptation method	Best initial	Average initial	Best final (MBF)	Average final	Gain	Running time [s]
AOC	0.86325	0.99440	0.69119	0.75307	20%	105.4112
AOC2	0.86239	0.99479	0.69061	**0.74204**	20%	104.0792
MOS	0.86009	0.99279	**0.69019**	0.76860	20%	103.3855

The final value of MBF are comparable for all types of adaptation. It is noteworthy that the results were somewhat better than in the case of not using adaptation algorithms. The best mean fitness of the best individuals was obtained for MOS adaptation (it equals 0.69019). Interestingly, this method also gave the worst mean fitness of average individuals (of 0.76860), which means poor stability. In this case, the best value was obtained for AOC2 approach (of 0.74204).

Success rate of the GA with the AOC adaptation, usually reached a low value already at the end of the first adaptive era, and in successive epochs further decreases and stabilizes around the value of 0.1. Hence the probability of substitution one mutation operator to another was high all the time, and it was swapped at the end of almost every epoch (ie. every 200 generations).

In the case of AOC2, in the initial phase of the algorithm, when there is a sharp decline in the cost function, the mutation operator was not changed even once. In the later stage of evolution, as the fit function decreased more gently, the operator was changed more often. Using this method, substitution of the mutation operator could be observed after the 2000 generation.

With the MOS adaptation approach, all mutation operators implemented in the application were used. In each generation of evolutionary algorithm, the decision which mutation operator to use was taken repeatedly. This results in that new individuals are produced in a less stable manner – diagrams were more irregular.

5.3 Simulated annealing

For simulated annealing approach the following parameters were adopted: initial temperature of 0.1 and the final temperature of 10^{-5}, whose achievement was a termination condition. The initial size of epoch was 10 iterations, and the maximum was 500. The cooling coefficient was $\alpha = 0.95$, and the growth factor was $\beta = 1.2$. This gives a total of $81,183$ iterations.

Table 3. Results for simulated annealing

Mutation	Initial	Final	Gain	Average time [s]
SWAP	**0.98105**	**0.68947**	**30%**	**295.9947**
SCRAMBLE $k = 3$	1.00439	0.69017	31%	315.0103
INSERTION	1.01540	0.71162	30%	294.1847

The best mean fitness of final individuals for all tests was obtained for SWAP mutation. It equals 0.68947. Simulated annealing algorithm yielded slightly better results than adaptive mutation approach.

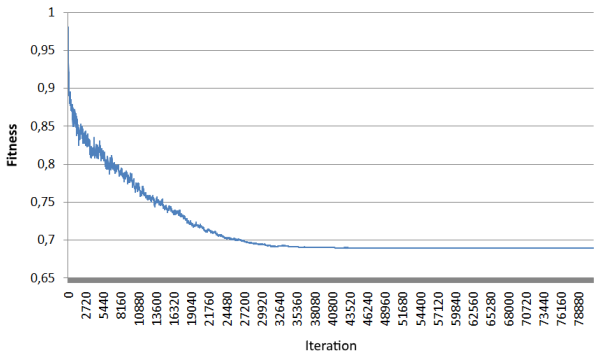


Fig. 5. Fitness for simulated annealing with SWAP mutation

As a result of the specificity of the simulated annealing algorithm, in the initial phase (at high temperatures) the probability to replace the current individual by worse one is high. This is reflected in the chart above – the beginning is more jagged.

5.4 The best keyboard arrangements found

The best keyboard arrangement was obtained using the simulated annealing algorithm. For the OM text ("With Fire and Sword"), the same arrangement was found using SA algorithm with SWAP and SCRAMBLE mutation operators. Its fitness value for this text was 0.68759101. Best keyboard for PT ("Sir Thaddeus") was obtained using SA with SWAP mutation (fitness of 0.69209257). The best fitness for the text WI (diploma guidelines) was 0.67895839. It was achieved by SA with SCRAMBLE mutation.

The following table shows the values of matching the best found keyboard arrangements for each of the test texts. Table rows correspond to the keyboards, and the column – blocks of text.

Table 4. Fitness of the top keyboards tested on different blocks of text

	OM	PT	WI	Average
OM keyboard	0.68759101	0.71119728	0.69419239	0.69766023
PT keyboard	**0.69542199**	**0.69209257**	**0.69207537**	**0.69319664**
WI keyboard	0.72413948	0.73202195	0.67895839	0.71170661

Keyboard trained on the WI text has poor fit for other blocks of text. Apparently it overfitted to the content of this particular document. The best average cost of rewriting all the texts was obtained with a keyboard optimized on the PT block. Its arrangement is shown on Fig. 6.

It is worth to compare the optimized keyboard with the other ones. Unfortunately, as noted by Herma in [7] there are no specific arrangements dedicated to the Polish language. Therefore, a comparison with to other popular arrangements was conducted – QWERTY and DVORAC. Table 5 summarizes the matching function (F_{FIT}) of said keyboards to that shown in Fig. 6 (PL_OPT). The table also includes time needed to rewrite each of the test blocks of text, estimated on the basis of the Fitts's law.

As can be seen, optimized keyboard has been rated as better than the QWERTY layout by about 30% from the perspective of the text of the novel "With Fire and

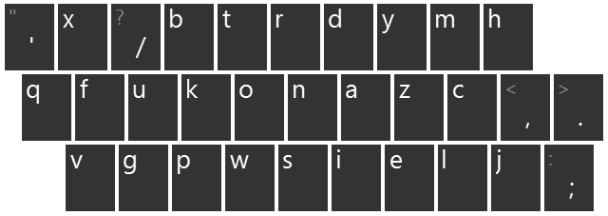


Fig. 6. The best keyboard arrangement for Polish language found

Table 5. Comparison results for different keyboard arrangements

	OM		PT		WI	
	F_{FIT}	time [h]	F_{FIT}	time [h]	F_{FIT}	time [h]
QWERTY	1.0000	197:29	1.0000	58:01	1.0000	2:47
DVORAC	1.0748	208:03	1.0923	61:11	1.1208	3:00
PL_OPT	0.6954	158:52	0.6921	46:50	0.6921	2:14

Sword" and about 31% from the perspective of the poem "Sir Thaddeus" and the document WI (diploma guidelines). Using keyboard arrangement tuned to the Polish language, the average time gain (in relation to the QWERTY keyboard) is 19.52%.

6. Conslusions and final thoughts

The goal of this work was to develop keyboard arrangement optimal for Polish language. Taking the standard, rectangular layout and set of keys of QWERTY keyboard (Polish programmer variant), optimal permutation was searched for, taking into account criteria of distance and position. For this purpose, authors used methods of machine learning such as evolutionary algorithms and simulated annealing.

This goal was fully achieved. Among numerous arrangements, found in experiments with different algorithms and sets of text, one was selected, featuring the best mean fitness. In other words, selected arrangement is well suited to the specifics of the Polish language, and is not overfitted to a specific block of text. The multicriterial function, used to evaluate keyboard arrangement, showed that the found keyboard gives more that 30% better fit over the standard QWERTY arrangement. Basing on the estimation of the typing time, calculated according to the Fitts's law, was shown that the time cost of writing text in is smaller than in the case of QWERTY and DVORAC keyboards. The results are very promising and encourage to the practical realization of such a keyboard arrangement.

There are many possibilities of future research. Authors consider conducting experiments using other algorithms or different sets of text. The most interesting research direction is to give up standard keyboard layout and try to find a different one, of custom shape.

It should be noted however, that regardless of the calculated costs and fitness, a very important factor is the habits of the target users. If you are not accustomed to the new arrangement, you will have to spend some time relearning the keyboard layout to achieve greater typing speed than with QWERTY. Is the final profit from using the optimized keyboard is worth the time spent on getting used to it? It depends on the individual needs and expectations of each of us.

References

[1] Navid Samimi Behbahan. Optimization of farsi letter arrangement on keyboard by simulated annealing and genetic algorithms. *Majlesi Journal of Multimedia Processing*, 2012.

[2] Randy Cassingham. The dvorak keyboard. `http://www.dvorak-keyboard.com`. Retrived: 2016-01-23.

[3] Mauro Dell'Amico, José Carlos Díaz Díaz, Manuel Iori, and Roberto Montanari. The single-finger keyboard layout problem. *Computers & Operations Research*, 36(11):3002–3012, 2009.

[4] Richard Dickenson. Did sholes and densmore know what they were doing when they designed their keyboard? *ETCetera – Journal of the Early Typewriter Collectors Association*, 1989.

[5] Bogusław Filipowicz, Wojciech Chmiel, Maciej Dudek, and Piotr Kadłuczka. Efektywność wielopopulacyjnego algorytmu ewolucyjnego dla zagadnień permutacyjnych. *Automatyka/Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie*, 15:147–158, 2011.

[6] Jeffrey S Goettl, Alexander W Brugh, and Bryant A Julstrom. Call me e-mail: arranging the keyboard with a permutation-coded genetic algorithm. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 947–951. ACM, 2005.

[7] Mariusz Herma. Na tropie polskiej klawiatury. `http://www.polityka.pl`. Retrived: 2016-09-05.

[8] Manar I Hosny, Nourah Alswaidan, and Abir Benabid Najjar. An optimized single-finger arabic keyboard layout. In *Science and Information Conference (SAI), 2014*, pages 321–328. IEEE, 2014.

[9] Wojciech Kulik. Klawiatura ekranowa kalq przyspieszy o 34% pisanie na tabletach i smartfonach. `http://www.benchmark.pl/`. Retrived: 2016-09-01.

[10] Yanzhi Li, Lijuan Chen, and Ravindra S Goonetilleke. A heuristic-based approach to optimize keyboard design for single-finger keying applications. *International Journal of Industrial Ergonomics*, 36(8):695–704, 2006.

[11] Alberto Moraglio and Riccardo Poli. Geometric crossover for the permutation representation. *Intelligenza Artificiale*, 5(1):49–63, 2011.

[12] Tania Pencheva, Krassimir Atanassov, and Anthony Shannon. Modelling of a stochastic universal sampling selection operator in genetic algorithms using generalized nets. In *Proceedings of the Tenth International Workshop on Generalized Nets, Sofia*, pages 1–7, 2009.

[13] Aleksandar Prokopec and Marin Golub. Adaptive mutation operator cycling. In *Applications of Digital Information and Web Technologies, 2009. ICADIWT'09. Second International Conference on the*, pages 634–639. IEEE, 2009.

[14] Marc Oliver Wagner, Bernard Yannou, Steffen Kehl, Dominique Feillet, and Jan Eggers. Ergonomic modelling and optimization of the keyboard arrangement with an ant colony algorithm. *Journal of Engineering Design*, 14(2):187–208, 2003.

[15] Sławomir Tadeusz Wierzchoń. *Sztuczne systemy immunologiczne: teoria i zastosowania*. Akademicka Oficyna Wydawnicza EXIT, 2001.

[16] Wikipedia. Keyboard layout. `https://en.wikipedia.org/wiki/Keyboard_layout`. Retrived: 2016-08-23.

List of Abbreviations

AOC	Adaptive operator cycling
GA	Genetic algorithm
KAP	Keyboard arrangement problem
MBF	Mean best fitness
MOS	Mutation operators statistics
OX	Order crossover
PMX	Partially mapped crossover
QAP	Quadratic assignment problem
SA	Simulated annealing
SFKL	Single finger keyboard layout
SUS	Stochastic universal sampling
UX	Uniform crossover

OPTYMALIZACJA UKŁADU KLAWIATURY EKRANOWEJ DLA JĘZYKA POLSKIEGO

Streszczenie Celem niniejszej pracy było opracowanie układu klawiatury ekranowej przeznaczonej dla języka polskiego. Przyjęto standardowy kształt i organizację klawiatury, zatem jest to zadanie wskazania najlepszej permutacji klawiszy, przy czym permutacji podlegały jedynie klawisze znaków alfabetu oraz pięć wybranych znaków interpunkcyjnych. W celu realizacji tak określonego zadania, posłużono się metodami uczenia maszynowego: algorytmami genetycznymi oraz algorytmem symulowanego wyżarzania. Funkcja dopasowania opiera się na dwóch utworach literackich oraz jednym dokumencie technicznym. Zastosowano kryteria odległości oraz lokalizacji klawiszy (biorąc pod uwagę kierunek pisania oraz wagi rzędów). Aplikację przygotowaną w celu wykonania badań eksperymentalnych opracowano w języku Java. W pracy opisano zastosowane algorytmy oraz przedstawiono wyniki uzyskane na drodze eksperymentów. Najlepsze znalezione układy pozwoliłyby skrócić czas wprowadzania przykładowych tekstów o około 30% (zakładając odpowiednie opanowanie nowego układu przez piszącego).

Słowa kluczowe: optymalizacja układu klawiatury, algorytmy genetyczne, symulowane wyżarzanie

Artykuł zrealizowano w ramach pracy badawczej S/WI/2/2013 i sfinansowano ze środków na naukę MNiSW.

THE LIST OF REVIEWERS
(2016)

1. Piotr Artiemjew
2. Jarosław Bylina
3. Francisco Javier Díez
4. Joanna Domańska
5. Damian Karwowski
6. Adam Nowosielski
7. Grzegorz Pastuszak
8. Małgorzata Przybyła-Kasperek
9. Waldemar Rakowski
10. Mariusz Rybnik
11. Adam Zagórecki