Computer Science Journal of Moldova, vol.**26**, no.2(77), 2018

Efficiency and Penalty Factors on Monoids of Strings

Mitrofan Choban, Ivan Budanaev

Abstract

In information theory, linguistics and computer science, metrics for measuring similarity between two given strings (sequences) are important. In this article we introduce efficiency, measure of similarity and penalty for given parallel decompositions of two strings. Relations between these characteristics are established. In this way, we continue the research from [3], [4].

Keywords: invariant distance, measure of similarity, Levenshtein distance, Hamming distance, Graev method, penalty.

1 Introduction

Let G be a semigroup and d be a metric on G. The metric d is called:

- *left* (respectively, *right*) *invariant* if $d(xa, xb) \leq d(a, b)$ (respectively, $d(ax, bx) \leq d(a, b)$) for all $x, a, b \in G$;
- *invariant* if it is both left and right invariant;
- *strong invariant* if $d(xa, xb) = d(ax, bx) = d(a, b)$ for all $x, a, b \in G$;
- *stable* if $d(xy, uv) \leq d(x, u) + d(y, v)$ for all $x, y, u, v \in G$.

Example 1.1. *Let G be the additive semigroup of non-negative real numbers and G^+ be the subsemigroup of positive real numbers. We put $d(x, x) = 0$ for each $x \in G$, $d(x, y) = 1$ for any distinct numbers $x, y \in G^+$ and $d(0, x) = d(x, 0) = 2$ for any $x \in G^+$. Then d is an invariant metric on G. Since* $1 = d(2, 5) = d(0 + 2, 3 + 2) < d(0, 3) = 2$*, the metric d is not strong invariant.*

The following assertion is well known.

Proposition 1. *Let d be an invariant metric on a group G. Then the metric d is strong invariant and $d(x^{-1}, y^{-1}) = d(x, y)$ for all $x, y \in G$.*

A monoid is a semigroup with an identity element. Fix a non-empty set A. The set A is called an alphabet. We put $\bar{A} = A \cup \{\varepsilon\}$. Let $L^*(A)$ be the set of all finite strings $a_1a_2 \dots a_n$ with $a_1, a_2, \dots, a_n \in \bar{A}$. Let ε be the empty string. Consider the strings $a_1a_2 \dots a_n$ for which $a_i = \varepsilon$ for some $i \leq n$. If $a_i \neq \varepsilon$, for any $i \leq n$ or $n = 1$ and $a_1 = \varepsilon$, the string $a_1a_2 \dots a_n$ is called an *irreducible string* or *canonical string*. The set $Supp(a_1a_2 \dots a_n) = \{a_1, a_2, \dots, a_n\} \cap A$ is the support of the string $a_1a_2 \dots a_n$ and $l(a_1a_2 \dots a_n) = |\{i \leq n : a_i \neq \varepsilon\}|$ is the length of the string $a_1a_2 \dots a_n$. For two strings $a_1 \dots a_n$ and $b_1 \dots b_m$, their product (concatenation) is $a_1 \dots a_n b_1 \dots b_m$. If $n \geq 2$, $i < n$ and $a_i = \varepsilon$, then the strings $a_1 \dots a_n$ and $a_1 \dots a_{i-1}a_{i+1} \dots a_n$ are considered equivalent. In this case any string is equivalent to one unique canonical string. We identify the equivalent strings. The set $L(A)$ of all canonical strings is the family of all classes of equivalent strings. In this case $L^*(A)$ is a semigroup and $L(A)$ becomes a monoid with identity ε. The set $L(A)$ is not a subsemigroup of $L^*(A)$. Only the set $L(A) \setminus \{\varepsilon\}$ is a subsemigroup of the semigroup $L^*(A)$.

Let $Supp(a, b) = Supp(a) \cup Supp(b) \cup \{\varepsilon\}$, and $Supp(a, a) = Supp(a) \cup \{\varepsilon\}$. It is well known that any subset $L \subset L(A)$ is an abstract language over the alphabet A.

Let a, b be two strings. For any two representations $a = a_1a_2 \dots a_n$ and $b = b_1b_2 \dots b_m$ we put

$$\begin{aligned} d_H(a_1a_2 \dots a_n, b_1b_2 \dots b_m) = & |\{i : a_i \neq b_i, i \leq min\{n, m\}\}| \\ & + |\{i : n < i \leq m, b_i \neq e_i\}| \\ & + |\{j : m < j \leq n, a_j \neq e_i\}|. \end{aligned}$$

The function d_H is called the Hamming distance on the space of strings [3], [4], [7].

Now we put:

$$d_G(a, b) = inf\{d_H(a, b) : a = a_1a_2 \dots a_n, b = b_1b_2 \dots b_n\}.$$

The function d_G is called the Graev – Markov distance on the space of strings [6], [9].

The V. I. Levenshtein's distance $d_L(a, b)$ between two strings $a = a_1a_2 \ldots a_n$ and $b = b_1b_2 \ldots b_m$ is defined as the minimum number of insertions, deletions, and substitutions required to transform one string into the other [4], [8].

We put $A^{-1} = \{a^{-1} : a \in A\}$, $\varepsilon^{-1} = \varepsilon$, $(a^{-1})^{-1} = a$ for any $a \in A$ and consider that $A^{-1} \cap \bar{A} = \emptyset$. Denote $\check{A} = A \cup A^{-1} \cup \{\varepsilon\}$. Let $\check{L}(A)$ $= L(\check{A})$ be the set of all strings over the set $\check{A}$. The strings over the set $\check{A}$ are called *words*. A word $a = a_1a_2 \ldots a_n \in \check{L}(A)$ is called an irreducible string if either $n = 1$ and $a_1 \in \check{A}$, or $n \geq 2$, $a_i \neq \varepsilon$ for any $i \leq n$ and $a_j^{-1} \neq a_{j+1}$ for each $j < n$.

Let $a = a_1a_2 \ldots a_n \in \check{L}(A)$ and $n \geq 2$. Then:

- if $i \leq n$ and $a_i = \varepsilon$, then the words $a_1 \ldots a_n$ and $a_1 \ldots a_{i-1}a_{i+1} \ldots a_n$ are considered equivalent;

- if $i < n$ and $a_i^{-1} = a_{i+1}$, then the words $a_1a_2 \ldots a_n$ and $a_1 \ldots a_{i-1}\varepsilon a_{i+2} \ldots a_n$ are considered equivalent.

In this case, any word $a_1a_2 \ldots a_n \in \check{L}(A)$ is equivalent to one unique irreducible word from $\check{L}(A)$. We identify equivalent words. Classes of equivalence form free group $F(A)$ over A with unity ε. We have that $L(A)$ is a subsemigroup of the group $F(A)$.

Let $a = a_1a_2 \ldots a_n \in F(A)$ be an irreducible word. The representation $a = x_1x_2 \ldots x_m \in L^*(A)$ is called an *almost irreducible* representation of a if there exist $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $a_j = x_{i_j}$ for any $j \leq n$ and $x_i = \varepsilon$ for each $i \in \{1, 2, ..., m\} \setminus \{i_1, i_2, ..., i_n\}$. If $a = a_1a_2 \ldots a_n \in L^*(A)$ is a representation of the string a, then $a_1a_2 \ldots a_n$ is an almost irreducible word.

If $a = a_1a_2 \ldots a_n$, then $a^s = a_na_{n-1} \ldots a_1$ and $a^{-1} = a_n^{-1}a_{n-1}^{-1} \ldots a_1^{-1}$. The word a^s is the symmetric word of a and a^{-1} is the inverse word of a. If a and b are equivalent words, then the words a^{-1} and b^{-1} are equivalent, as well as the words a^s and b^s.

Hence the mappings $\cdot^s, \cdot^{-1} : F(A) \longrightarrow F(A)$ are the group automorphisms. Obviously that $L(A)^s = L(A)$.

Let $a, b \in A$ and $a \neq b$, then we put $d_H(a, b) = d_H(a^{-1}, b^{-1}) = d_H(a, \varepsilon) = d_H(\varepsilon, a) = d_H(a^{-1}, \varepsilon) = d_H(\varepsilon, a^{-1}) = 1$. If $a \in A$ and $b \in$

A^{-1}, then $d_H(a,b) = d_H(b,a) = 2$. For any $x \in \check{A}$ we put $d_H(x,x) = 0$. Thus d_H is a metric on $\check{A}$. For any two words $a_1a_2 \dots a_n, b_1b_2 \dots b_m \in \check{L}(A)$ we put:

$$\begin{aligned} d_H(a_1a_2 \dots a_n, b_1b_2 \dots b_m) =& \Sigma\{d_H(a_i,b_i) : i \leq min\{n,m\}\} \\ &+|\{i : n < i \leq m, b_i \neq e_i\}| \\ &+|\{j : m < j \leq n, a_j \neq e_i\}|. \end{aligned}$$

For $a, b \in F(A)$ we put:

$$\check{d}(a,b) = inf\{d_H(a,b) : a = a_1 \dots a_n \in \check{L}(A), b = b_1 \dots b_n \in \check{L}(A)\}.$$

Remark 1.1. *The function $\check{d}$ is called the Graev – Markov distance on the free group [6]. The method of extensions of distances for free groups, used by us, was proposed by A. A. Markov [9] and M. I. Graev [6]. For metrics on free universal algebras it was extended in [2], for quasimetrics on free groups and varieties of groups it was examined in [5], [12].*

M. I. Graev [6] has proved the following assertions:

G1. $\check{d}$ is an invariant metric on $F(A)$ and $\check{d}(a,b) = d_H(a,b)$ for all $a, b \in A^*$.

G2. If ρ is an invariant metric on $F(A)$ and $\rho(x,y) \leq d_H(x,y)$ for any $x, y\ inA^*$, then $\rho(x,y) \leq \check{d}(x,y)$ for any $x, y \in F(A)$.

G3. For any two words $a, b \in F(A)$ there exist $m \geq 1$ and two almost irreducible representations $a = x_1x_2 \dots x_m$ and $b = y_1y_2 \dots y_m$ such that $\check{d}(a,b) = d_H(x_1x_2 \dots x_m, y_1y_2 \dots y_m)$.

Theorem 1.1. *The distance d_G on a monoid $L(A)$ has the following properties:*

1. d_G is a strong invariant metric on $L(A)$ and $d_G(x,y) = d_G(zx,zy) = d_G(xz,yz)$ for all $x, y, z \in L(A)$.

2. $d_G(a,b) = d_G(a^s,b^s)$ for all $a, b \in L(A)$.

3. If ρ is an invariant metric on $L(A)$ and $\rho(x,y) \leq d_G(x,y)$ for all $x, y \in \bar{A}$, then $\rho(a,b) \leq d_G(a,b)$ for all $a, b \in L(A)$.

4. For any $a, b \in L(A)$ there exist $n \in \mathbb{N}$, $x_1, x_2, \dots, x_n \in Supp(a,a)$ and $y_1, y_2, \dots, y_n \in Supp(b,b)$ such that $a = x_1x_2 \dots x_n$,

$b = y_1y_2 \dots y_n$ *such that* $n \leq l(a) + l(b)$ *and* $d_G(a,b) = |\{i : i \leq n, a_i \neq b_i\}| = d_H(x_1x_2 \dots x_n, y_1y_2 \dots y_n)$.

5. $d_G(a,b) = d_L(a,b) = \check{d}(a,b) \leq d_H(a,b)$ *for all* $a, b \in L(A)$.

Proof. Fix $a, b \in L(A)$. Let $a = a_1a_2 \dots a_n$, $b = b_1b_2 \dots b_n$. If $n > l(a) + l(b)$, then there exists $i \leq n$ such that $a_i = b_i = \varepsilon$, $a = a_1a_2 \dots a_{i-1}a_{i+1} \dots a_n$, $b = b_1b_2 \dots b_{i-1}b_{i+1}, \dots b_n$ and $d_H(a_1a_2...a_n, b_1b_2...b_n) = d_H(a_1...a_{i-1}a_{i+1}...a_n,\ b_1...b_{i-1}b_{i+1}...b_n)$. Hence $d_G(a,b) = inf\{d_H(a_1a_2 \dots a_n,\ b_1b_2 \dots b_n) : a = a_1a_2 \dots a_n$, $b = b_1b_2 \dots b_n,\ n \leq l(a) + l(b)\}$. Since we have finite pairs of parallel representations $a = a_1a_2 \dots a_m$, $b = b_1b_2 \dots b_m$ of length $m \leq l(a)+l(b)$, there exist $n \in \mathbb{N}$, $x_1, x_2, \dots, x_n \in Supp(a,a)$ and $y_1, y_2, \dots, y_n \in Supp(b,b)$ such that $a = x_1x_2 \dots x_n$, $b = y_1y_2 \dots y_n$ with $n \leq l(a)+l(b)$ and $d_G(a,b) = |\{i : i \leq n, a_i \neq b_i\}| = d_H(x_1x_2 \dots x_n, y_1y_2 \dots y_n)$. Thus, *Assertion 4* is proved. *Assertion 2* is obvious.

Fix $a, b \in L(A)$ and $c \in A$. It is clear that $d_G(ca, cb) \leq d_G(a,b)$. Assume that $d_G(ca, cb) < d_G(a,b)$. Then there exist representations $ca = x_1x_2 \dots x_n$ and $cb = y_1y_2 \dots y_n$ such that $n \leq l(a) + l(b) + 2$ and $d_G(ca, cb) = d_H(x_1x_2 \dots x_n, y_1y_2 \dots y_n)$, where $A \cap \{x_i, y_i\} \neq \emptyset$ for each $i \leq n$. If $x_1 = y_1$, then $x_1 = y_1 = c$. In this case $a = x_2 \dots x_n$, $b = y_2 \dots y_n$ and $d_G(a,b) \leq d_H(x_2 \dots x_n, y_2 \dots y_n) = d_H(x_1x_2 \dots x_n, y_1y_2 \dots y_n) = d_H(ca, cb) < d_H(a,b)$, a contradiction. Hence $x_1 \neq y_1$. In this case we have two possibilities: $x_1 = c$, $y_1 = \varepsilon$ or $x_1 = \varepsilon$, $y_1 = c$. We can assume that $x_1 = c$ and $y_1 = \varepsilon$. Let $1 < j$, $y_j = c$ and $y_i = \varepsilon$ for each $i < j$. We put $u_1 = v_i = \varepsilon$ for each $i \leq j$, $u_i = x_i$ for each $i \geq 2$ and $v_k = y_k$ for each $k > j$. Then $b = u_1u_2 \dots u_n$, $b = v_1v_2 \dots v_n$, $0 = d_H(u_1, v_1) < d_H(x_1, y_1) = 1$, $d_H(x_j, y_j) \leq 1$, $d_H(u_j, v_j) \leq 1$ and $d_H(u_i, v_i) = d_H(x_i, y_i)$ for $i \in \{2, 3, ..., j-1, j+1, ..., n\}$. Hence $d_G(a,b) \leq d_H(u_1u_2...u_n, v_1v_2...v_n) \leq d_H(x_1x_2...x_n, y_1y_2...y_n) = d_G(ca, cb) < d(a,b)$, a contradiction. Hence $d_G(ca, cb) = d(a,b)$. From *Assertion 2* it follows that $d_G(ac, bc) = d_G(a,b)$. *Assertion 1* is proved.

We put $d(x,x) = 0$ and $d(x,y) = 1$ for any distinct strings $x, y \in L(A)$. Let $ID(A)$ denote the family of all invariant metrics ρ on $L(A)$ with the property: $\rho(x,y) \leq d(x,y)$ for all $x, y \in \overline{(}A)$. Since $d \in ID(A)$, the set $ID(A)$ is non-empty. Now we put $d^*(a,b) = sup\{\rho(a,b) : \rho \in$

$ID(A)\}$. One can easily observe that $d^* \in ID(A)$, $d(a,b) \leq d^*(a,b)$ for any $a,b \in L(A)$ and $d(x,y) = d^*(x,y) = 1$ for all distinct $x,y \in \overline{(}A)$.

Property 1. If $\rho \in ID(A)$, then

$$\begin{aligned}\rho(x_1x_2 \dots x_n, y_1y_2 \dots y_n) \leq & |\{i \leq 1 : x_i \neq y_i\}| \\ = & d_H(x_1x_2 \dots x_n, y_1y_2 \dots y_n)\end{aligned}$$

for any two strings $(x_1x_2 \dots x_n, y_1y_2 \dots y_n) \in L(A)$.

This property follows from the conditions of invariance of metric d.

Property 2. $d_G = d^* = d_L$.

Since d_G and d^* are invariant distances on $L(A)$ and they are constructed with the conditions of extremity

$$d^*(a,b) = sup\{\rho(a,b) : \rho \in ID(A)\},$$
$$d_G(a,b) = inf\{d_H(a,b) : a = a_1a_2 \dots a_n \ , \ b = b_1b_2 \dots b_n\},$$

we have $d_G = d^*$. In [3], [4] it was proved that $d^* = d_L$. The equality $d_G(a,b) = \check{d}(a,b)$ for all $a,b \in L(A)$ follows from the Graev's assertion $G3$ in the above Remark. This completes the proof of the theorem. $\square$

Example 1.2. *The metrics d, $d_G = d_L = d^*$ are strong invariant on $L(A)$. On $L(A)$ there exists a metric $d_r \in ID(A)$ which is invariant, but not strong invariant. Fix a real number r for which $2^{-1} \leq r < 1$. We put $d_r(x,x) = 0$ for each $x \in L(A)$, $d(x,y) = r$ for any distinct strings $x,y \in L(A) \setminus \{\varepsilon\}$ and $d(0,x) = d(x,0) = 1$ for any $x \in L(A) \setminus \{\varepsilon\}$. Then d is an invariant metric on G. Fix $a \in A$. Since $r = d(a,aa)$ $= d(\varepsilon \cdot a, a \cdot a) < d(\varepsilon, a) = 1$, the metric d_r is not strong invariant.*

Remark 1.2. *For the metric d_H we have $d_H(a,b) \leq max\{l(a,l(b)\}$ for any strings $a,b \in L(A)$. The Hamming distance d_H is left invariant: $d_H(xa,xb) = d(a,b)$ for all strings $x,a,b \in L(A)$. Assume now that $x,y,z \in A$, $a = xyzxyz$, $b= yzxy$ and $c = xyz$. Then $d_G(a,b) = 2$ and $6 = d_H(a,b) < d_H(ac,bc) = 9$. Therefore, metric d_H is not right invariant.*

2 Efficiency and Penalty of Two Strings

The longest common substring and pattern matching in two or more strings is a well known class of problems. For any two strings $a, b \in L(A)$ we find the decompositions of the form $a = v_1u_1v_2u_2 \dots v_ku_kv_{k+1}$ and $b = w_1u_1w_2u_2 \dots w_ku_kw_{k+1}$, which can be represented as $a = a_1a_2 \dots a_n$, $b = b_1b_2 \dots b_n$ with the following properties:

- some a_i and b_j may be empty strings, i.e. $a_i = \varepsilon$, $b_j = \varepsilon$;
- if $a_i = \varepsilon$, then $b_i \neq \varepsilon$, and if $b_j = \varepsilon$, then $a_j \neq \varepsilon$;
- if $u_1 = \varepsilon$, then $a = v_1$ and $b = w_1$;
- if $u_1 \neq \varepsilon$, then there exists a sequence $1 \leq i_1 \leq j_1 < i_2 \leq j_2 < \dots < i_k \leq j_k \leq n$ such that:

$u_1 = a_{i_1} \dots a_{j_1} = b_{i_1} \dots b_{j_1}$, $u_2 = a_{i_2} \dots a_{j_2} = b_{i_2} \dots b_{j_2}$, $u_k = a_{i_k} \dots a_{j_k} = b_{i_k} \dots b_{j_k}$;

- if $v_1 = w_1 = \varepsilon$, then $i_1 = 1$;
- if $v_{k+1} = w_{k+1} = \varepsilon$, then $j_k = n$;
- if $k \geq 2$, then for any $i \in \{2, \dots, k\}$ we have $v_i \neq \varepsilon$ or $w_i \neq \varepsilon$.

In this case $l(u_1) + l(u_2) + \dots + l(u_k) = |\{i : a_i = b_i\}|$.

The above decompositions forms are called *parallel decompositions* of strings a and b [3], [4]. For any parallel decompositions $a = v_1u_1 \dots v_ku_kv_{k+1}$ and $b = w_1u_1 \dots w_ku_kw_{k+1}$ the number

$$E(v_1u_1 \dots v_ku_kv_{k+1}, w_1u_1 \dots w_ku_kw_{k+1}) \\ = \sum_{i \leq k+1} \{\max\{l(v_i), l(w_i)\}\} = d_H(x_1x_2 \dots x_n, y_1y_2 \dots y_n)$$

is called the efficiency of the given parallel decompositions. The number $E(a, b)$ is equal to the minimum of efficiency values of all parallel decompositions of the strings a, b and is called the *common efficiency of the strings a,b*. It is obvious that $E(a, b)$ is well determined and $E(a, b) = d_G(a, b)$. We say that the parallel decompositions $a = v_1u_1v_2u_2 \dots v_ku_kv_{k+1}$ and $b = w_1u_1w_2u_2 \dots w_ku_kw_{k+1}$ are optimal if the following equality holds:

$$E(v_1u_1v_2u_2 \dots v_ku_kv_{k+1}, w_1u_1w_2u_2 \dots w_ku_kw_{k+1}) = E(a, b).$$

This type of decompositions are associated with the problem of approximate string matching [10]. If the decompositions $a = v_1u_1 \dots v_ku_kv_{k+1}$ and $b = w_1u_1 \dots w_ku_kw_{k+1}$ are optimal and $k \geq 2$, then we may consider that $u_i \neq \varepsilon$ for any $i \leq k$.

Any parallel decompositions $a = a_1a_2 \dots a_n = v_1u_1v_2u_2 \dots v_ku_kv_{k+1}$ and $b = b_1b_2 \dots b_n = w_1u_1w_2u_2 \dots w_ku_kw_{k+1}$ generate a common subsequence $u_1u_2 \dots u_k$. The number $m(a_1a_2 \dots a_n, b_1b_2 \dots b_n) = l(u_1) + l(u_2) + \dots + l(u_k)$ is the *measure of similarity* of the decompositions [1], [11]. There exist parallel decompositions $a = v_1u_1v_2u_2 \dots v_ku_kv_{k+1}$ and $b = w_1u_1w_2u_2 \dots w_ku_kw_{k+1}$ for which the measure of similarity is maximal. The maximum value of the measure of similarity of all decompositions is denoted by $m^*(a, b)$. The maximum value of the measure of similarity of all optimal decompositions is denoted by $m^\omega(a, b)$. We can note that $m^\omega(a, b) \leq m^*(a, b)$. For any two parallel decompositions $a = a_1a_2 \dots a_n$ and $b = b_1b_2 \dots b_n$ as in [4], we define the *penalty factors* as

$$p_r(a_1a_2 \dots a_n, b_1b_2 \dots b_n) = |\{i \leq n : a_i = \varepsilon\}|,$$

$$p_l(a_1a_2 \dots a_n, b_1b_2 \dots b_n) = |\{j \leq n : b_j = \varepsilon\}|,$$

$$p(a_1a_2 \dots a_n, b_1b_2 \dots b_n) = |\{i \leq n : a_i = \varepsilon\}| + |\{j \leq n : b_j = \varepsilon\}|$$

$$= p_r(a_1a_2 \dots a_n, b_1b_2 \dots b_n) + p_l(a_1a_2 \dots a_n, b_1b_2 \dots b_n)$$

and

$$\begin{aligned} M_r&(a_1a_2 \dots a_n, b_1b_2 \dots b_n) \\ &= m(a_1a_2 \dots a_n, b_1b_2 \dots b_n) - p_r(a_1a_2 \dots a_n, b_1b_2 \dots b_n) \\ M_l&(a_1a_2 \dots a_n, b_1b_2 \dots b_n) \\ &= m(a_1a_2 \dots a_n, b_1b_2 \dots b_n) - p_l(a_1a_2 \dots a_n, b_1b_2 \dots b_n) \\ M&(a_1a_2 \dots a_n, b_1b_2 \dots b_n) \\ &= m(a_1a_2 \dots a_n, b_1b_2 \dots b_n) - p(a_1a_2 \dots a_n, b_1b_2 \dots b_n) \end{aligned}$$

as the *measures of proper similarity*.

The number $d_H(a_1a_2\dots a_n, b_1b_2\dots b_n) = |\{i \leq n : a_i \neq b_i\}|$ is the Hamming distance between decompositions and it is another type of penalty: we have that $p(a_1\dots a_n, b_1\dots b_n) \leq d_H(a_1\dots a_n, b_1\dots b_n)$.

The assertions from the following theorem establish the main results.

Theorem 2.1. *Let a and b be two non-empty strings, $a = a_1a_2\dots a_n$ and $b = b_1b_2\dots b_n$ be the initial optimal decompositions, and $a = a'_1a'_2\dots a'_q$ and $b = b'_1b'_2\dots b'_q$ be the second decompositions, which are arbitrary. Denote by*

$$\begin{aligned} m &= m(a_1a_2\dots a_n, b_1b_2\dots b_n), & m' &= m(a'_1a'_2\dots a'_n, b'_1b'_2\dots b'_q),\\ p &= p(a_1a_2\dots a_n, b_1b_2\dots b_n), & p' &= p(a'_1a'_2\dots a'_n, b'_1b'_2\dots b'_q),\\ p_l &= p_l(a_1a_2\dots a_n, b_1b_2\dots b_n), & p'_l &= p_l(a'_1a'_2\dots a'_n, b'_1b'_2\dots b'_q),\\ p_r &= p_r(a_1a_2\dots a_n, b_1b_2\dots b_n), & p'_r &= p_r(a'_1a'_2\dots a'_n, b'_1b'_2\dots b'_q),\\ r &= d_H(a_1a_2\dots a_n, b_1b_2\dots b_n), & r' &= d_H(a'_1a'_2\dots a'_n, b'_1b'_2\dots b'_q), \end{aligned}$$

$$\begin{aligned} M &= m - p, & M' &= m' - p',\\ M_l &= m - p_l, & M'_l &= m' - p'_l,\\ M_r &= m - p_r, & M'_r &= m' - p'_r. \end{aligned}$$

The following assertions are true:

1. *$p' - p = 2(m' - m) + 2(r' - r)$.*

2. *If the second decompositions are non optimal, then $M_l > M'_l$ and $M_r > M'_r$.*

3. *If the second decompositions are optimal, then $M_l = M'_l$ and $M_r = M'_r$ and the measures M_l and M_r are constant on the set of optimal parallel decompositions.*

4. *If $m' \geq m$ and the second decompositions are non optimal, then $p' > p$, $p_l\prime > p_l$, $p'_r > p_r$ and $M > M'$.*

5. *If $m' = m$ and the second decompositions are optimal, then $p' = p$, $p_{l'} = p_l$, $p'_r = p_r$ and $M' = M$.*

6. *If $m' \leq m$ and the second decompositions are non optimal, then $m' - r' < m - r$.*

The proof of Theorem 2.1 follows from the next lemmas.

Lemma 1.

$$p_r(a'_1a'_2 \ldots a'_q, b'_1b'_2 \ldots b'_q) = q - l(a),$$
$$p_l(a'_1a'_2 \ldots a'_q, b'_1b'_2 \ldots b'_q) = q - l(b),$$
$$p(a'_1a'_2 \ldots a'_q, b'_1b'_2 \ldots b'_q) = 2q - l(a) - l(b).$$

Proof. Follows immediately from the definitions of penalty factors and parallel decompositions. □

Lemma 2. $p' - p = 2(m' - m) + 2(r' - r)$.

Proof. From Lemma 1 it follows that $p' - p = (2q - l(a) - l(b)) - (2n - l(a) - l(b)) = 2(q - n)$. Since $q = m' + r'$ and $n = m + r$, the proof is complete. □

Lemma 3. $p'_l - p_l = p'_r - p_r = (m' - m) + (r' - r)$.

Proof. We can assume that $l(a) \leq l(b)$. Then $p_l = (l(b) - l(a)) + l_r$ and $p_l - p_r = l(b) - l(a)$. Hence $p_l - p_r = p'_l - p'_r$ and $p'_l - p_l = p'_r - p_r$. The equality $p' - p = (p'_r - p_r) + (p'_l - p_l)$ and Lemma 2 complete the proof. □

Lemma 4. *Assume that $m' > m$. Then:*
1. *$M > M'$, $M_l \geq M'_l$ and $M_r \geq M'_r$.*
2. *$M_l > M'_l$ and $M_r > M'_r$ provided that the second decompositions are non optimal.*
3. *$M_l = M'_l$ and $M_r = M'_r$ provided that the second decompositions are optimal.*

Proof. Since the initial decompositions are optimal, we have $r' \geq r$. Moreover, we have $r' = r$ if and only if the second decompositions are optimal as well. By virtue of definitions, we have $n = m + r$ and $q = m' + r'$. Therefore $n < q$. From Lemma 2, it follows that $p' - p = 2(m' - m) + 2(r' - r)$ and $p < p'$. Thus $p' - p > m' - m$ and $M = m - p > m' - p' = M'$.

Also, from Lemma 3, it follows that $p'_l - p_l = p'_r - p_r = (m' - m) + (r' - r)$. Hence, $M_l = m - p_l = (m' - p'_l) + (r' - r) = M'_l + (r' - r)$ and $M_r = m - p_r = (m' - p'_r) + (r' - r) = M'_r + (r' - r)$. Since $r' \geq r$ and $r' = r$ if and only if the second decompositions are optimal, the proof is complete. □

Corollary 2.1. *The measures M_l and M_r are constant on the set of optimal parallel decompositions.*

Lemma 5. *Let $m' = m$. Then:*

1. $M \geq M'$, $M_l \geq M'_l$ and $M_r \geq M'_r$.

2. $M_l > M'_l$ and $M_r > M'_r$ provided that the second decompositions are non optimal.

3. $M_l = M'_l$ and $M_r = M'_r$ provided that the second decompositions are optimal.

Proof. We have that $n = m + r$ and $q = m' + r'$. Since $r \leq r'$, we have that $n \leq q$.

Assume that $M < M'$. Then $m - p < m' - p'$, $p' = 2q - l(a) - l(b)$ and $p = 2n - l(a) - l(b)$. Hence $m - 2n + l(a) + l(b) < m - 2q + l(a) + l(b)$, or $-2n < -2q$ and $n > q$, a contradiction.

From Lemma 3 it follows that $p'_l - p_l = p'_r - p_r = r' - r$. Hence $p'_l \geq p_l$ and $p'_r \geq p_r$. If the second decompositions are non optimal, then $p'_l > p_l$ and $p'_r > p_r$. Assertions are proved. □

Lemma 6. *Assume that $m' < m$ and the second decompositions are non optimal. Then $M_l > M'_l$ and $M_r > M'_r$.*

Proof. Since the initial decompositions are optimal, we have $r' > r$. By virtue of Lemma 3, we have $p'_l - p_l = p'_r - p_r = (m' - m) + (r' - r)$. Hence, $M_l = m - p_l = (m' - p'_l) + (r' - r) = M'_l + (r' - r)$ and

$M_r = m - p_r = (m' - p'_r) + (r' - r) = M'_r + (r' - r)$. Since $r' - r > 0$, the proof is complete. $\square$

Remark 2.1. *From Assertions 1 and 3 of Theorem 2.1 it follows that on the class of all optimal decompositions of given two strings:*

- *the maximal measure of proper similarity is attained on the optimal parallel decomposition with minimal penalties (minimal measure of similarity);*
- *the minimal measure of proper similarity is attained on the optimal parallel decomposition with maximal penalties (maximal measure of similarity).*

3 Computing algorithms

The algorithm of computing the Levenshtein distance for the case of a discrete metric was presented in [4]. Below we show a well known algorithm (see Algorithm 1, Annex 1) that permits to calculate the Graev-Markov-Levenshtein distance between two irreducible strings for any metric.

For any two non-empty strings there exist parallel decompositions with maximal measure of similarity and optimal decompositions on which measure of similarity is minimal. Pseudo-code of such algorithm is presented in Algorithm 2 (see Algorithm 2, Annex 2).

Algorithm 2 makes calls to functions $LevenshteinDistance$ and $BuildOPD$. The first function computes distance function and builds the memoization matrix. The function $BuildOPD$ uses the memoization matrix to generate optimal parallel decompositions. The pseudocode for these functions was presented in [4].

4 Conclusions

For any two non-empty strings there exist parallel decompositions with maximal measure of similarity and optimal decompositions on which measure of similarity is minimal. The following example shows that there exist some exotic non optimal parallel decompositions $a =$

$a'_1a'_2\dots a'_q$ and $b = b'_1b'_2\dots b'_q$, such that for optimal decompositions $a = a_1a_2\dots a_n$ and $b = b_1b_2\dots b_n$ we have $m' < m$, $p' < p$ and $M' > M$.

Example 4.1. *Let* $a = ABCDEF$ *and* $b = CDEFED$ *be trivial non optimal decompositions of strings* a, b, *and* $a = ABCDEF\varepsilon\varepsilon$ *and* $b = \varepsilon\varepsilon CDEFED$ *be their optimal decompositions. Then* $m' = 1$, $r' = 5$, $p' = p'_l = p'_r = 0$ *and* $m = 4$, $r = 4$, $p = 4$, $p_l = p_r = 2$. *In this example we have that* $M'_l = M'_r = M' = m' - p' = 1 - 0 = 1 > 0 = 4 - 4 = m - p = M$, $m' - r' = -4 < 0 = m - r$, $M_l = 4 - 2 = 2 > 1 = M'_l$, $M_r = 4 - 2 = 2 > 1 = M'_r$.

Example 4.2. *Let* $a = AAAACCC$ *and* $b = CCCBBBB$ *be trivial optimal decompositions of strings* a, b, *and* $a = AAAACCC\varepsilon\varepsilon\varepsilon\varepsilon$ *and* $b = \varepsilon\varepsilon\varepsilon\varepsilon CCCBBBB$ *be their non-optimal decompositions. Then* $m' = 3$, $r' = 8$, $p' = 8$ *and* $m = 0$, $r = 7$, $p = p_l = p_r = 0$. *In this example we have that* $-5 = m' - r' > m - r = -7$ *and* $-5 = m' - p' < m - p = 0$.

The above examples show that Theorem 2.1 cannot be improved in the case of $m' < m$.

Decompositions with minimal penalty and maximal proper similarity are of significant interest. Moreover, if we consider the problem of text editing and correction, the optimal decompositions are more favorable. Therefore, optimal decompositions are the best parallel decompositions and we may solve the string matching problems only on class of optimal decompositions.

To summarize the results, we established that optimal decompositions:

- describe the proper similarity of two strings;
- permit to obtain long common sub-sequences;
- permit to calculate the distance between strings;
- permit to appreciate changeability of information over time.

References

[1] V. B. Barahnin, V. A. Nehaeva and A. M. Fedotov, “Similarity Determination for Textual Documents Clusterization,” *Vestnik*

Novosibirskogo Gos. Un-ta, Ser. Informactionnye tehnologii, vol. 6, no. 1, pp. 3–9, 2008. (in Russian).

[2] M. M. Choban, "The theory of stable metrics," *Math. Balkanica*, vol. 2, pp. 357–373, 1988.

[3] M. M. Choban and I. A. Budanaev, "Distances on Monoids of Strings and Their Applications," in *Proceedings of the Conference on Mathematical Foundations of Informatics MFOI2016*, (Chisinau, Republic of Moldova), 2016, pp. 144–159.

[4] M. M. Choban and I. A. Budanaev, "About Applications of Distances on Monoids of Strings," *Computer Science Journal of Moldova*, vol. 24, no. 3, pp. 335–356, 2016.

[5] M. M. Choban and L. L. Chiriac, "On free groups in classes of groups with topologies," *Bul. Acad. Ştiinţe Repub. Moldova, Matematica*, no. 2-3, pp. 61–79, 2013.

[6] M. I. Graev, "Free topological groups," *Trans. Moscow Math. Soc.*, vol. 8, pp. 303–364, 1962 (Russian original: *Izvestia Akad. Nauk SSSR*, vol. 12, 279–323, 1948).

[7] R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, vol. 29, no 2, pp. 147–160, 1952.

[8] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *DAN SSSR*, vol. 163, no 4, pp. 845–848, 1965 (in Russian) (English translation: *Soviet Physics – Doklady*, vol. 10, no. 8, pp. 707–710, 1966).

[9] A. A. Markov, "On free topological groups," *Trans. Moscow Math. Soc.*, vol. 8, pp. 195–272, 1962.

[10] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no 1, pp. 31–88, 2001.

[11] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no 3, pp. 443–453, 1970.

[12] S. Romaguera, M. Sanchis and M. Tkachenko, "Free paratopological groups," *Topology Proceed.*, vol. 27, no 2, pp. 613–640, 2003.

Mitrofan Choban, Ivan Budanaev, Received May 5, 2018

Mitrofan Choban
Tiraspol State University, Republic of Moldova
str. Iablochkin 5, Chisinau, Moldova
Phone: +373 69109553
E–mail: `mmchoban@gmail.com`

Ivan Budanaev
Doctoral School of Mathematics and Information Science
Institute of Mathematics and Computer Sciences
Tiraspol State University, Republic of Moldova
str. Academiei, 3/2, MD-2028, Chisinau, Moldova
E–mail: `ivan.budanaev@gmail.com`

ANNEX 1

Algorithm 1: Metric:
Given $x, y \in F(A)$ compute $\check{d}(x, y)$, for the case of metric.

Data: $x = x_1 x_2 \dots x_n$, $y = y_1 y_2 \dots y_m$, metric function $\check{d}$ on $\check{A}$.
Parameters: costs of insertion and removal operations - $cost_{insert}$ and $cost_{remove}$ respectively.
Result: $d_L(x, y)$, and matrix D.

`// initialize distance matrix`
for $i \leftarrow 1$ **to** m **do** D[i,0]=i;
for $j \leftarrow 1$ **to** n **do** D[0,j]=j;
`// initialize loop variables`
$i := 1, j := 1;$
for $j \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** m **do**
    if $dist(x_i, y_j) = 0$ **then**
      d[i, j] := d[i-1, j-1];
    else
      `// Dynamic Programming recursive function`
      d[i,j] := min(d[i-1, j] + $cost_{remove}$, min(d[i, j-1] + $cost_{insert}$, d[i-1, j-1] + $dist(x_i, y_i)$));
return D[m, n], D;

ANNEX 2

Algorithm 2: Maximal Measure of Similarity:
Finds maximum value of measure of similarity of $x, y \in L(\bar{A})$.

```
/* Helper functions to compute similarity and
   penalty factors                                   */
Function similarity (a,b)
    n = max(length(a), length(b))
    sim = 0
    for i ← 1 to n do
        if (a[i] == b[i]) sim = sim + 1
    return sim;
Function penalty (a,b)
    n = max(length(a), length(b))
    pen = 0
    for i ← 1 to n do
        if (a[i] == ε) pen = pen + 1
        if (b[i] == ε) pen = pen + 1
    return pen;
/* Main Algorithm Body                               */
Data: x = x_1 x_2 ... x_n, y = y_1 y_2 ... y_m.
Result: Maximal measure of similarity of x and y.
// Calling Metric or QuasiMetric Functions
d, D := LevenshteinDistance(x, y);
// Calling BackTracking function BuildOPD
S = BuildOPD(n,m,x,y,a,b,D);
max_sim = 0
for ((a,b): S) do
    sim = similarity(a,b) - penalty(a,b)
    max_sim = max_sim < sim ? sim : max_sim
return max_sim
```