sciendo

TATRA MOUNTAINS
Mathematical Publications

DOI: 10.2478/tmmp-2019-0005
Tatra Mt. Math. Publ. **73** (2019), 39–60

CRYPTANALYSIS OF THE COLUMNAR TRANSPOSITION USING META-HEURISTICS

Eugen Antal — Peter Javorka — Tomáš Hliboký

Institute of Computer Science and Mathematics
Slovak University of Technology in Bratislava
SLOVAKIA

ABSTRACT. The most commonly used methods for solving classical (historical) ciphers are based on global optimization (meta-heuristic methods). Despite the fact that global optimization is a well-studied problem, in the case of classical ciphers, there are still many open questions such as the construction of fitness functions or efficient transformation of the cryptanalysis (breaking attempt) to an optimization problem. Therefore the transformation of a cryptanalytical task to an optimization problem and the choice of a suitable fitness function form an important part of the topic. In this paper, we focus on the simple columnar transposition in depth. Our main contribution is a detailed analysis and comparison of different fitness functions, fitness landscape analysis and solving experiments.

1. Columnar transposition

The columnar transposition is a cipher where the order of plain-text letters is modified based on some selected permutation. The plain-text is divided into blocks of size b (the length of the permutation). The letters in each block are reordered based on the permutation. These blocks can be written into a matrix. There are two variants of this cipher. The cipher-text can be constructed by reading the blocks in rows or by reading the blocks in columns. We investigate the variant of the cipher with reading the re-ordered letters in rows.

2010 Mathematics Subject Classification: 94A60, 68P25.
Keywords: historical ciphers, columnar transposition, fitness function, grid, hill climbing, genetic algorithm, meta-heuristic, cryptanalysis.
This work was partially supported by grants VEGA 1/0159/17. We are grateful to the HPC center at the Slovak University of Technology in Bratislava, which is a part of the Slovak Infrastructure of High Performance Computing (SIVVP project, ITMS code 26230120002, funded by the European region development funds, ERDF), for the computational time and resources made available.

As an example, we encrypted the *hidden notice* plain-text, using the key *secret*. First, we obtain a permutation from the key phrase using the Bellaso's method [3] (Table 1).

Table 1. Permutation obtained from the key-phrase.

S	E	C	R	E	T
5	2	1	4	3	6

The plain-text is divided into blocks (Table 2).

Table 2. Transposition table.

5	2	1	4	3	6
h	i	d	d	e	n
n	o	t	i	c	e

In the next step, we apply the permutation on the text blocks.

Table 3. Reordered cipher-text.

1	2	3	4	5	6
d	i	e	d	h	n
t	o	c	i	n	e

The resulting cipher-text is: DIEDHN TOCINE.

2. Meta-heuristics

The main principle of a meta-heuristic method is an iterative search process, where the initial solution is mostly randomly generated. The saved solution(s) in a specific computation cycle (iteration) represent(s) the actual state of the method instance. In each iteration, a changing process is performed on these saved solution candidates. The goal is to improve these solutions [13]. In fact, this general description also describes one of the most commonly used meta-heuristic method: the *hill-climbing* (HC, see Algorithm 1).

The goal of a meta-heuristic method is to reach the global optimum of the fitness function. Unfortunately these methods may get stuck in local optimum while searching the solution space. The number of local optima depends on the fitness landscape. The fitness landscape [15] describes the global geometry of our search space in a more detailed way.

Algorithm 1 Hill-climbing algorithm

1: Generate an initial random solution candidate S iteration counter $k = 0$.
2: Find the neighbours $N(S)$ of S.
3: Randomly pick an element S' from $N(S)$.
4: Set S' as a new solution candidate $S = S'$, if S' has better fitness than S.
5: Next iteration $k \leftarrow k + 1$.
6: While the termination criteria is not satisfied, jump to step 2.

If we check all the neighbours (step 3 of Algorithm 1) and select the best solution candidate, the method is called "steepest ascent" or "gradient ascent" hill-climbing [10,15]. The gradient ascent version is commonly used in case of *fitness landscape* analysis.

2.1. Meta-heuristics in cryptanalysis

Meta-heuristics are mostly used as an approximation of the solution, based on some kind of intelligent search of a large solution space. The main idea of heuristic cryptanalysis is to transform the key space search to an optimization problem. The goal is to find the correct key of a selected cryptosystem. The quality of a solution candidate is evaluated with a fitness function - in our case, it is based on measuring the "meaningfulness" of the text [9, 14]. We can make an assumption that the correct result should be a meaningful text, and model this meaningfulness based on some statistical distance/similarity of quantitative characteristic of the text elements[1] (e.g. letter frequency).

The state of the art from this topic is presented in [1,2,4–8,11,12,16–18]. The prior work covers different types of classical ciphers.

Using this methodology we were able to successfully solve the simple substitution (described in [1, 2]). In this paper, we used this methodology to solve the columnar transposition[2]. We focused on analysis of the most commonly used

[1]The reference values were obtained from the OANC corpus [19].
[2]In both cases we have a combinatorial problem (permutation), only the key length differs.

fitness functions types:

- *Distance* - Manhattan distance (L1 distance) of the measured n-grams from the reference values.
$$fitness = \sum_{0}^{n} \big(|REF(n) - MES(n)| \big), \tag{1}$$
where n is the number of different values for a given n-gram, REF contains the reference values and MES contains the measured values.
- *Rank* - is based on the ranking order of n-grams. All the reference n-grams with the measured ones are sorted by their frequency in a descending order. For each n-gram we calculate the distance of their position in measured set from the position in the reference set.
- *Weighted sum* - we calculate the weighted occurrence of all n-grams. For each n-gram we multiply the count with the estimated frequency from the reference values.
$$fitness = \sum_{0}^{n} \big(REF(n) * COUNT(n) \big). \tag{2}$$

In our case the solution S (from Algorithm 1) is a permutation representing the key (see Table 2 and 3). Since the decryption algorithm is known and the solution candidate is directly the parameter of the decryption function, we can obtain the plain text candidate for each S. Instead of evaluating the quality of the key, we can evaluate the quality of the plain text with fitness functions described above.

In Algorithm 1 step 1, we are generating a random initial permutation. We defined $N(S)$ as a set of permutations obtained by swapping all possible pairs of elements of permutation S.

3. Experiments

We can logically divide our research into three parts:

- Preliminary fitness function analysis.
- Empirical analysis and comparison of different meta-heuristic methods and fitness functions.
- Fitness landscape analysis of selected fitness functions.

3.1. Fitness function analysis

In our first experiment, we investigated if the fitness functions met the most important criteria: evaluating the correct solution with the best fitness value. In this experiment we selected only a subset of our dataset: 10 texts of length l.

For keys of length[3] k we generated all possible permutations and calculated the number of false positive keys[4].

In this experiment we used the following parameters:

- Text lengths:
  - $l = 50, 500, 2000$.
- Key lengths:
  - $k = 5, 10$.
- Language models:
  - $n = 2, 3, \ldots, 7$[5].
- Fitness functions:
  - *distance*,
  - *rank*,
  - *weighted sum*.

The experiment itself tests all possible keys from the key space and evaluate them (the resulting plain-text candidate) one-by-one with the corresponding fitness function. The number of keys producing better score than the real one is stored. In Tables 4 and 5 the average values are visible.

We figured out that fitness functions based on measuring the statistical distance of obtained n-grams from the reference values are:

- not usable in case of $n > 4$ for any l,
- for $k = 5$ and $n < 4$ usable only in case of $l >= 500$,
- for $k = 10$ and $n < 4$ usable only in case of $l >= 2000$.

The best performing fitness function was based on weighted sum of n-grams. In case of $k = 5$ there are no false positive keys up to tested $n = 7$ for any text length. For $k = 10$ there are no false positive keys up to tested $n = 5$, for text long at least $l = 500$.

3.2. Comparison of meta-heuristic methods and fitness functions

We investigated the success rate of different meta-heuristics (variants of hill-climbing with restarts and genetic algorithm) with a combination of different fitness functions (based on n-gram statistics for different n). In our experiments, we selected various texts of different length from a reference corpus (100 different texts for each length). Each text was encrypted by a columnar transposition with a random key (random permutation of length k). We also used the corresponding plain-texts, to measure the match rate of the result with the correct solution. We analysed the success rate of various fitness functions - the match rate of the

[3]Due to the computational requirements we focused only on smaller keys.
[4]Keys evaluated with a better score than the correct solution.
[5]For $k = 10$ we tested only $n = 2, 3, \ldots, 5$ due to performance restrictions.

resulting solution with the correct solution and its dependence on the parameters of used meta-heuristics.

In this experiment we used the following parameters:

- Text lengths:
  - $l = 50, 100, 150, \ldots, 2000$.
- Key lengths:
  - $k = 5, 10, 25$ and 50.
- Fitness functions (with different language models):
  (1) L1 distance of 2-grams from the reference values,
  (2) L1 distance of 3-grams from the reference values,
  (3) weighted sum of 1000 most frequent 3-grams frequencies,
  (4) weighted sum of 1000 most frequent 4-grams frequencies,
  (5) weighted sum of 1000 most frequent 5-grams frequencies.
- Meta-heuristic methods:
  (1) hill-climbing,
    - termination criteria: 10000 and 100000 iterations.
  (2) hill-climbing with restarts,
    - $r = 10, 100$ restarts,
    - termination criteria: 10000 and 100000 iterations,
    - final result selected by the best score from all restarts.
  (3) steepest-ascent hill-climbing with restarts,
    - $r = 10, 100, 1000, 5000$ restarts,
    - termination criteria: nearest local/global optimum,
    - final result selected by the best score from all restarts.
  (4) genetic algorithm (the same configuration as described in [2]).

In case of hill-climbing (HC) with restarts, we obtain very notable results (near 100 % match rate) in case of $k = 5, 10$ for almost all investigated fitness functions and n. As it can bee seen on Figures 1 and 2, restarting the HC highly increase the success rate. On the other hand, the result is affected by the fitness function only slightly (compare the previous figures with Figure 3).

In case of $k = 25$ we can obtain near 70 % match rate with only 100 restarts, but only in case of fitness function based on weighted sum of most frequent 4-grams and 5-grams frequencies (Figure 4). This can be increased with gradient version of HC (SAHC) up to 90 % with 5000 restarts using fitness function based on weighted sum of most frequent 3-grams frequencies (Figure 5).

For $k = 50$ we were unable to reach any usable solution.

The obtained results depend also on the text length and on the number of restarts. Please note, that the average result on figures (the purple line) indicates the average result that can be obtained without restarts. The maximum

result (the green line) is selected as the best one from all of the results/restarts (average for the measured 100 texts for a specific text length).

In case of genetic algorithm (GA), we tested several schemes. Surprisingly the success rate was worse than expected. In case of $k = 5$, we can still achieve 100 % match rate for several GA schemes using 3-grams (Figure 7 and 8). For $k = 10$ the match rates of all GA schemes were below of those for HC. The best result for $k = 10$ (near 80 % match rate) can be obtained only with one GA scheme (scheme C[6]) and with the maximal value of tested population size ($p = 100$) with 3-grams only (Figures 9 and 10). For $k = 25, 50$ we obtained unusable results. From the overall results we can say, that in case of GA, the resulting match rate is affected mainly on the used GA scheme, also on the used n-gram.

3.3. Fitness landscape analysis of selected fitness functions

To measure the difficulty of our optimization problem, we also investigated the fitness landscape for selected fitness function. We used a gradient hill-climbing[7] with random restarts and the same parameters than before.

We focus on the number of local optima (Figures 11, 13, 12, 14 and 15). For large k it is impossible to find all the local optima, but we can guess it from the number of unique results obtained from restarts. The presented result is normalized with the number of restarts. The value 1 means, that each restart ends with a different result. The best scenario is to reach the $1/r$ value. From the obtained result it is clear that for smaller k we have less local optima. Increasing the k also increase the local optimum count. From $k = 25$ we reach the worst scenario.

Next we measured the average number of gradient hill-climbing iterations required to reach the first local optimum (Figures 16, 17 and 18). From the results it is clear that it depends mainly on the used n-gram, also on the k.

The last part of the experiment is to measure if the correct key represents a local optimum. If not, we measured how many iterations are needed to obtain the nearest local optimum. The obtained results (Figures 19, 20 and 21) indicate that in case of fitness function *distance* it depends mainly on the size of k and on the used n-gram. The results also shown that the most suitable function is the *weighted sum*, where the correct solution is an optimum and it does not depend on the key size.

[6]Half of the chromosomes are selected with tournament selection without any change. The second half of the chromosomes are selected with tournament selection, using one swap as mutation.

[7]Gradient/steepest-ascent hill-climbing always ends in a local (global) optimum.

Summary

The most commonly used methods for solving classical ciphers are based on global optimization. In this paper, we focus on the simple columnar transposition in depth. Our main contribution is a detailed analysis and comparison of different fitness functions, fitness landscape analysis and solving experiments.

In our first experiment, we investigated if the fitness functions met the most important criteria: evaluating the correct solution with the best fitness value. The best performing fitness function was based on weighted sum of n-grams.

In our second experiment we investigated the success rate of different meta-heuristics with a combination of different fitness functions (based on n-gram statistics for different n). Investigated methods:

- hill-climbing,
- hill-climbing with restarts,
- steepest-ascent hill-climbing with restarts,
- genetic algorithm.

We analysed the success rate of various fitness functions—the match rate of the resulting solution with the correct solution and its dependence on the parameters of used meta-heuristics. The best performing meta-heuristic method was hill-climbing (also the steepest-ascent version) with restarts in combination with fitness functions based on 3-gram frequencies. We can obtain usable result up to $k = 25$. The result depends also on the text length and on the number of restarts. The genetic algorithm performed worst than hill-climbing in general.

In our third experiment we investigated the fitness landscape for selected fitness function. We used a gradient hill-climbing with random restarts. We focus on the number of local optima and on the average number of gradient hill-climbing iterations required to reach the first local optimum. We also measured if the correct key represents a local optimum, and how many iterations are needed to obtain the nearest local optimum. The obtained results indicate that fitness functions weighted sum of 3-grams is the most suitable evaluation method in case of columnar transposition.

From all the obtained results we can state that in case of transposition ciphers it is important to use fitness functions based on weighted frequency sum rather than based on statistical distance. The success rate may vary depending on the text length, used n-grams and also on the parameters of the meta-heuristic method. We recommend to use 3-gram statistics.

There Appendix A with Table 4 and Table 5, Appendix B, with Figures 1–21 and Bibliography follow on the next pages.

Appendix A. Tables

Table 4. Average false positives for fitness function, $k = 5$.

(a) weighted sum; $k = 5$.

n-gram	l = 50	l = 500	l = 2000
2, 3	0.2	0.0	0.0
4, 6	0.1	0.0	0.0
5	0.3	0.0	0.0
7	0.0	0.0	0.0

(b) distance; $k = 5$.

n-gram	l = 50	l = 500	l = 2000
2	7.8	1.4	0.7
3	0.3	0.4	0.0
4	106.4	14.1	0.0
5,6,7	119.0	119.0	119.0

(c) rank; $k = 5$.

n-gram	l = 50	l = 500	l = 2000
2, 3	0.2	0.0	0.0
4	0.0	0.0	0.0
5	0.4	0.0	0.0
6	56.5	58.1	67.8
7	114.4	118.2	119.0

TABLE 5. Average false positives for fitness function, $k = 10$.

(A) weighted sum; $k = 10$.

n-gram	l = 50	l = 500	l = 2000
2	550.4	0.0	0.0
3	2124.1	0.0	0.0
4	1667.2	0.0	0.0
5	5081.2	0.0	0.0

(B) distance; $k = 10$.

n-gram	l = 50	l = 500	l = 2000
2	118230.4	820.0	0.0
3	11157.9	0.0	0.0
4	3399382.7	575.0	0.0
5	3628798.0	3628799.0	3628799.0

(C) rank; $k = 10$.

n-gram	l = 50	l = 500	l = 2000
2	59.0	0.0	0.0
3	6.2	0.0	0.0
4	0.7	0.0	0.0
5	28043.3	654.8	0.0

Appendix B. Figures

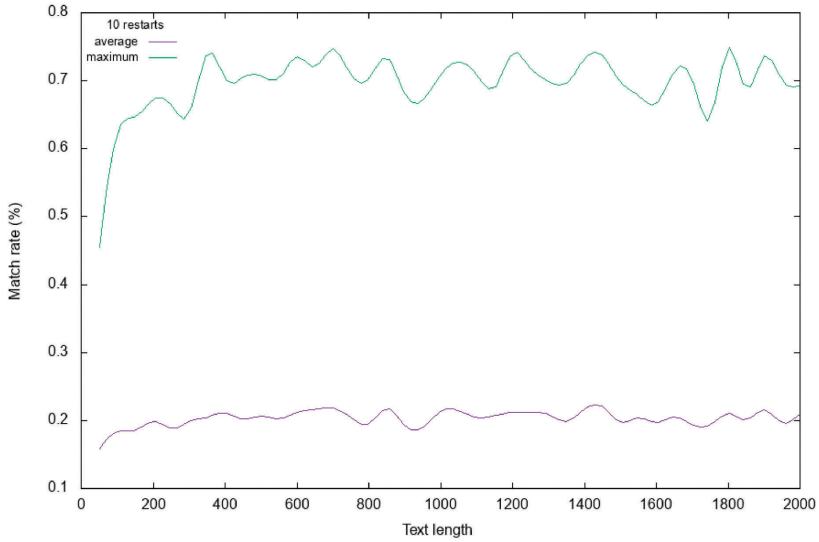

Figure 1. HC results for function 2; k=10; 10000 iterations and 10 restarts.

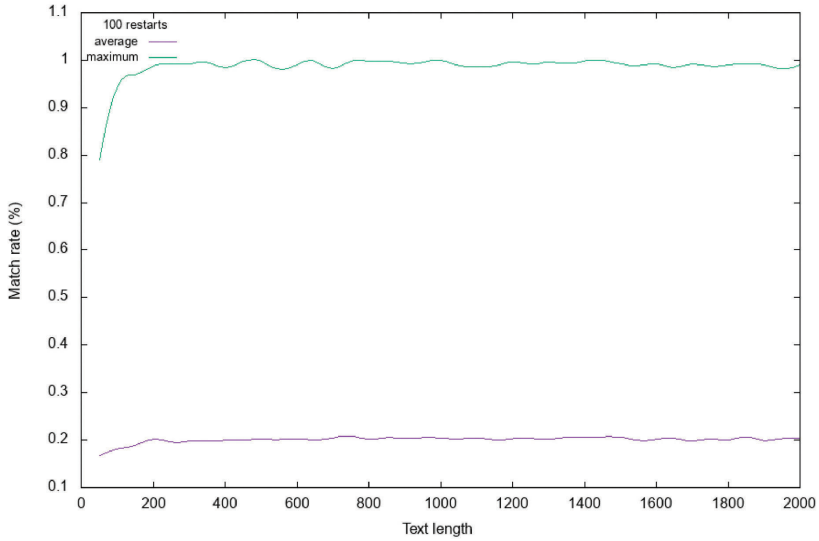

Figure 2. HC results for function 2; k=10; 10000 iterations and 100 restarts.

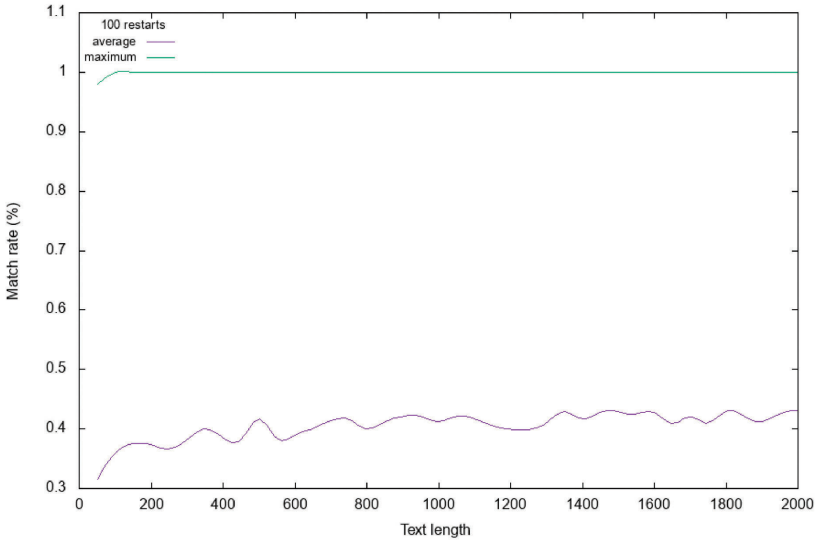

FIGURE 3. HC results for function 5; k=5; 10000 iterations and 100 restarts.

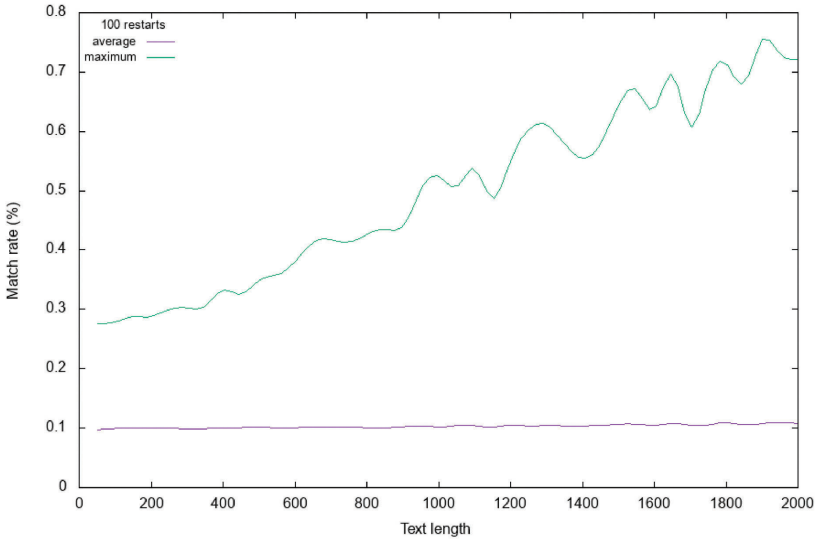

FIGURE 4. HC results for function 5; k=25; 10000 iterations and 100 restarts.

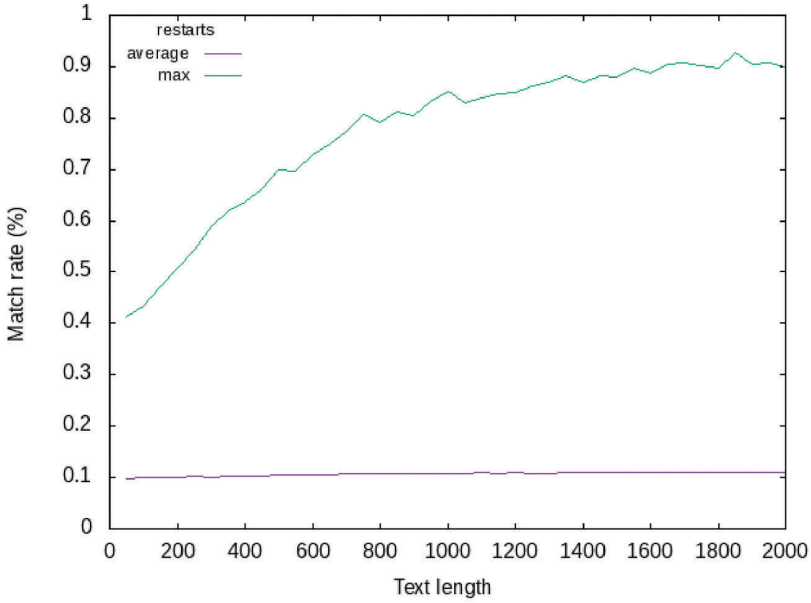

FIGURE 5. SAHC results for function 3; k=25; 5000 restarts.

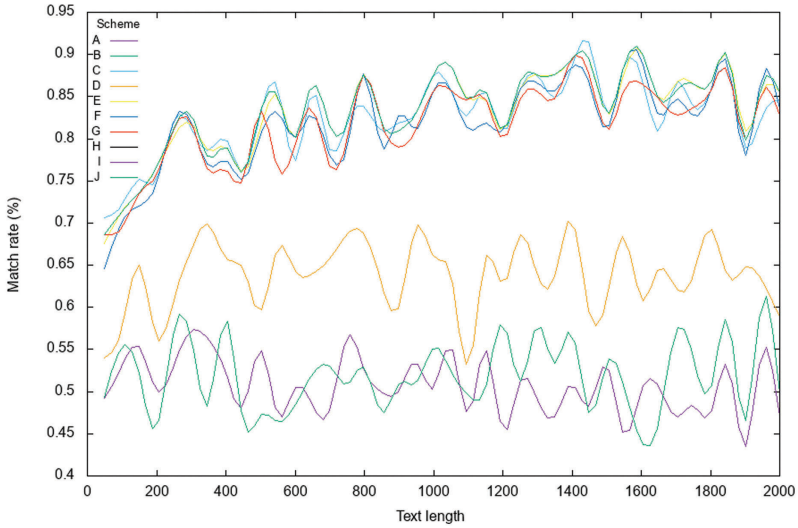

FIGURE 6. GA results for function 1; k=5; population size 100; 10000 iterations.

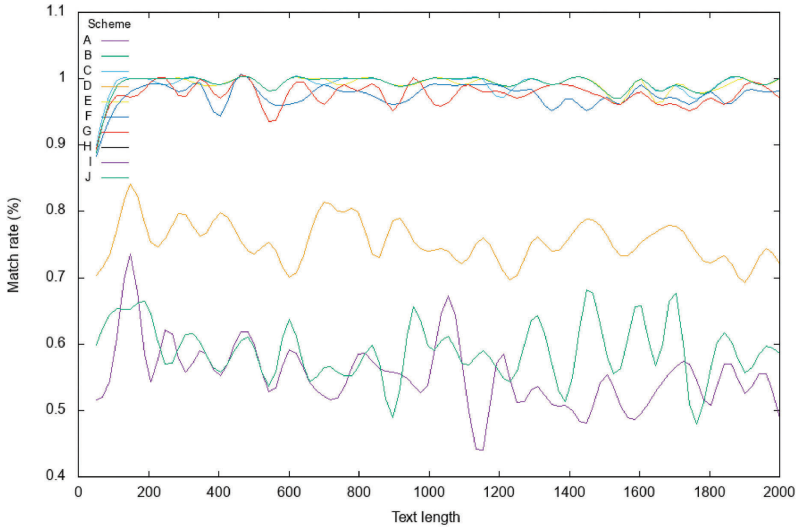

FIGURE 7. GA results for function 2; k=5; population size 100; 10000 iterations.

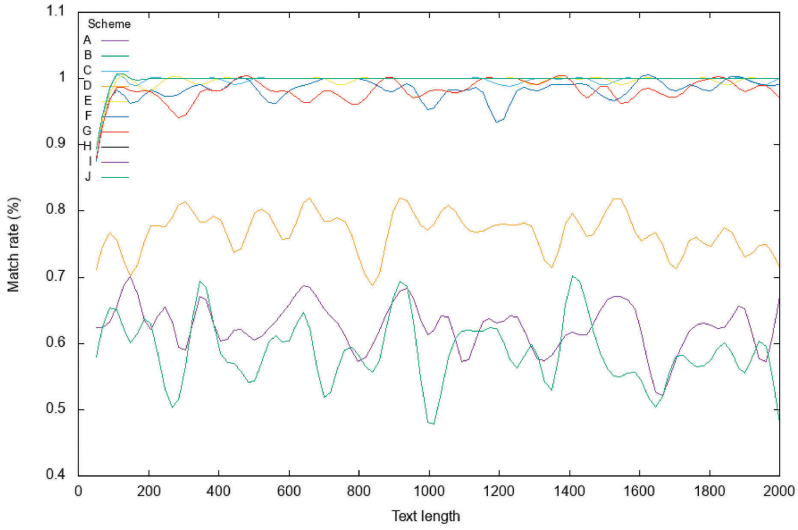

FIGURE 8. GA results for function 3; k=5; population size 100; 10000 iterations.

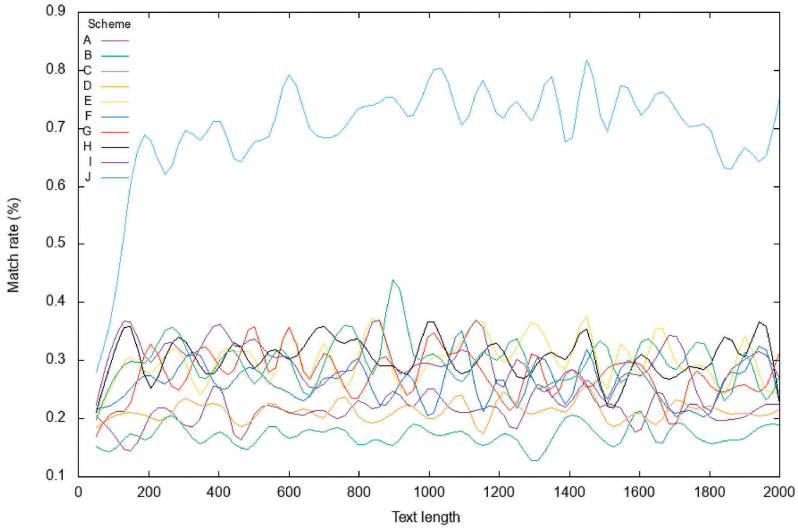

FIGURE 9. GA results for function 2; k=10; population size 100; 10000 iterations.

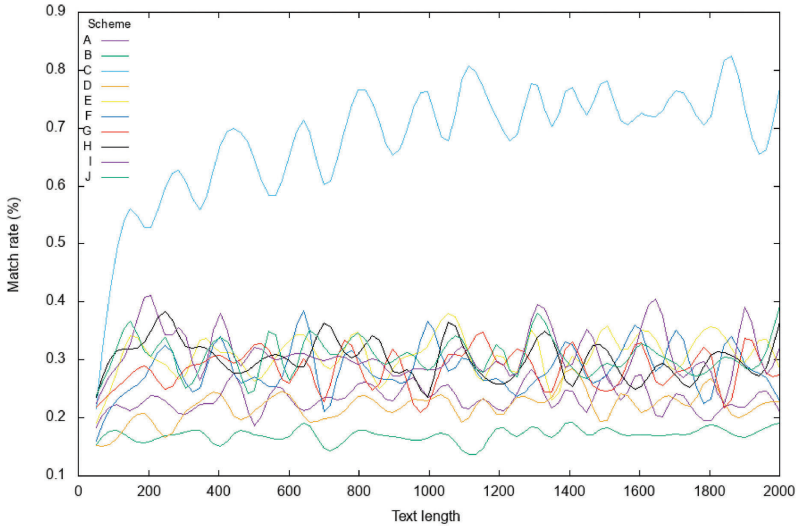

FIGURE 10. GA results for function 3; k=10; population size 100; 10000 iterations.

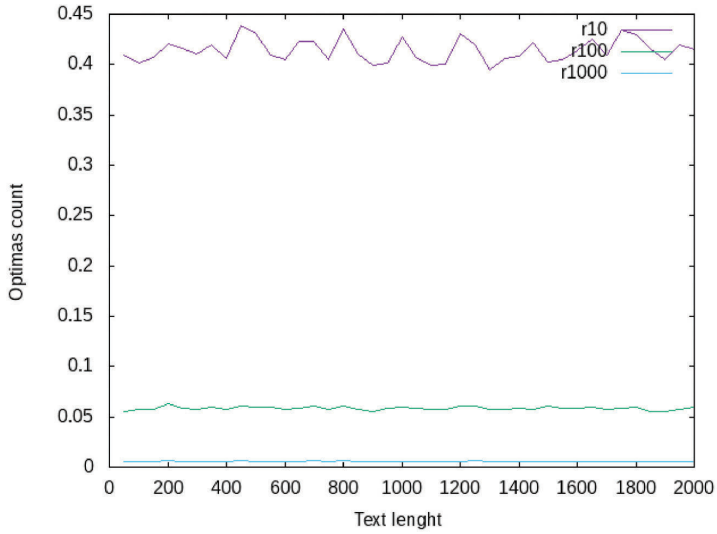

FIGURE 11. Number of unique results for fitness function 2; 3-grams; all restarts; $k = 5$.

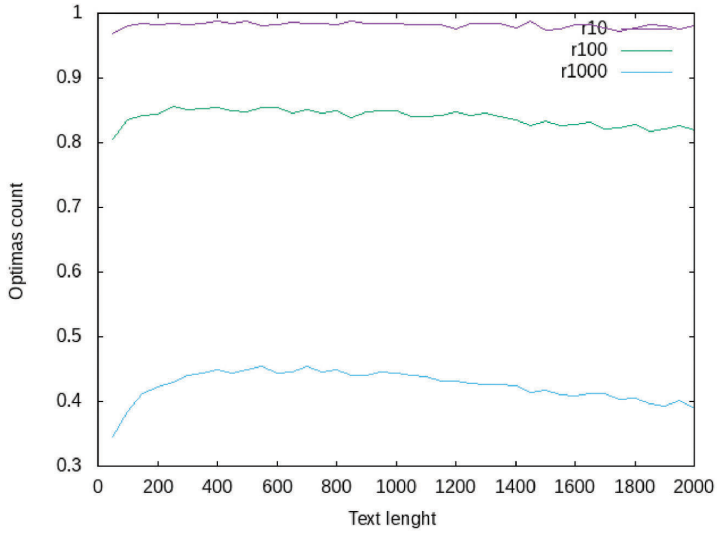

FIGURE 12. Number of unique results for fitness function 2; all restarts; $k = 10$.

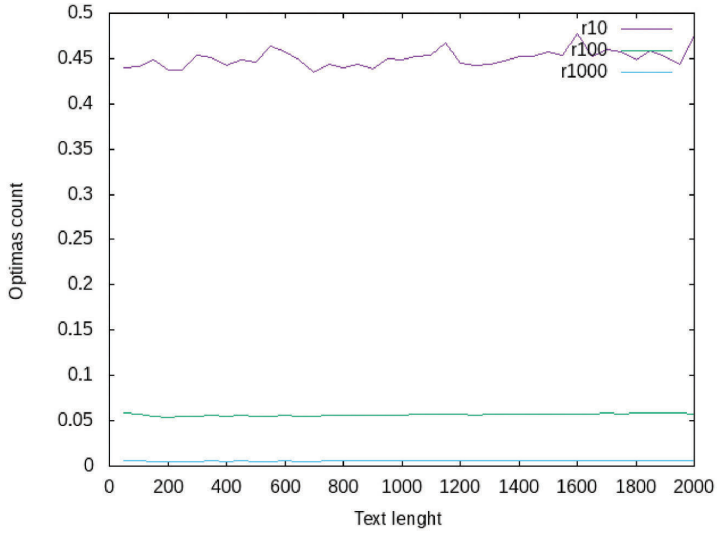

FIGURE 13. Number of unique results for fitness function 3; all restarts; $k = 5$.

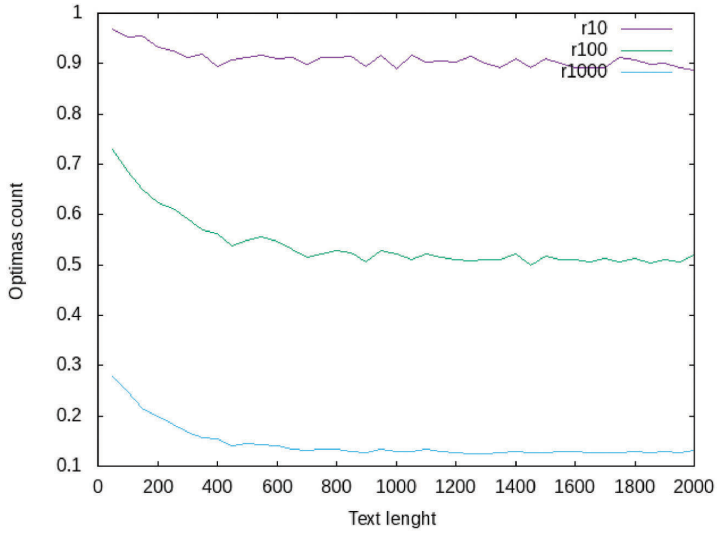

FIGURE 14. Number of unique results for fitness function 3; all restarts; $k = 10$.

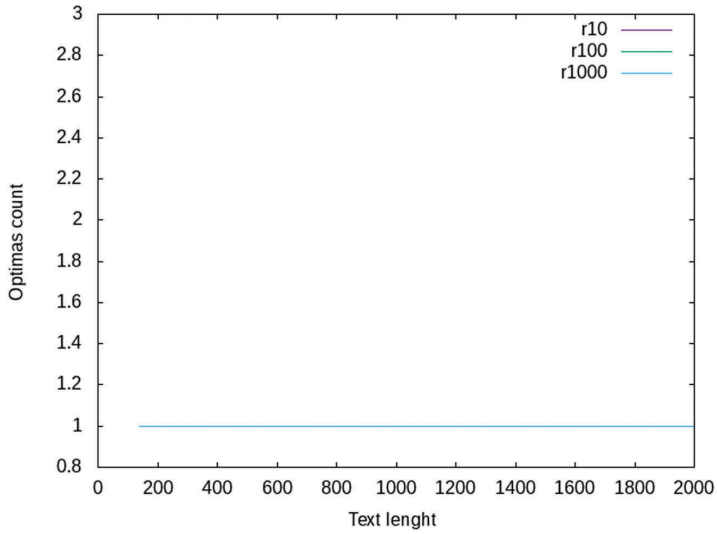

FIGURE 15. Number of unique results for fitness function 3; all restarts; $k = 25$.

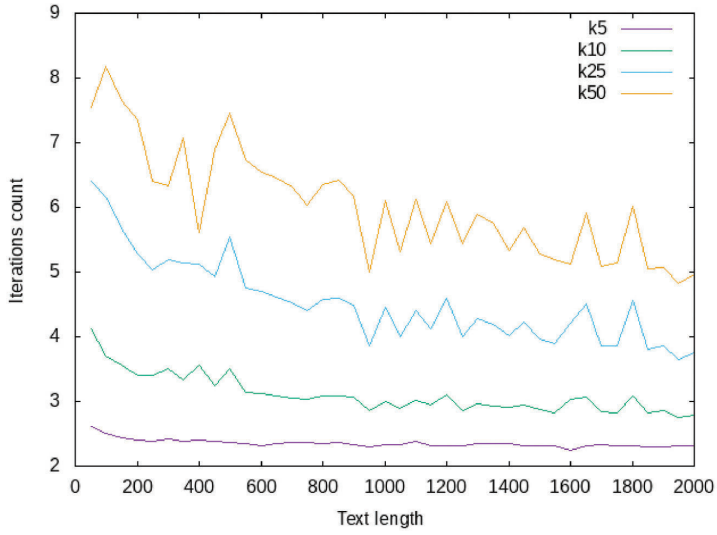

FIGURE 16. Average number of iteration count until the first local optima is reached for function 1;1000 restarts; $k = 5, 10, 25, 50$.

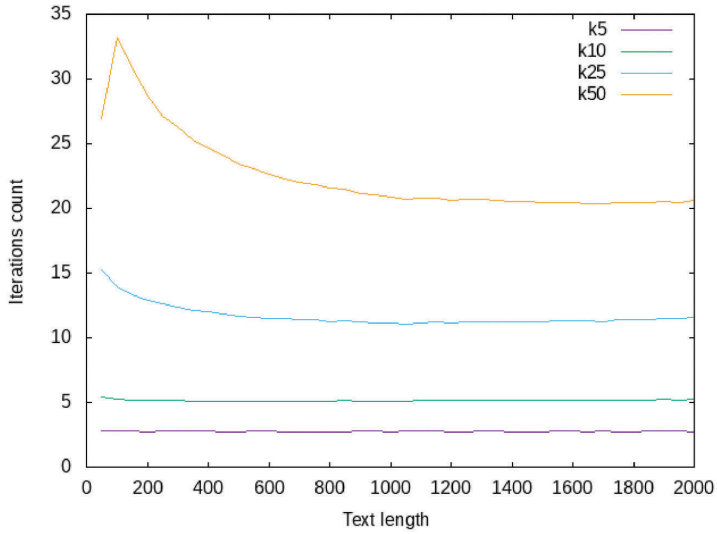

FIGURE 17. Average number of iteration count until the first local optima is reached for function 2; 1000 restarts; $k = 5, 10, 25, 50$.

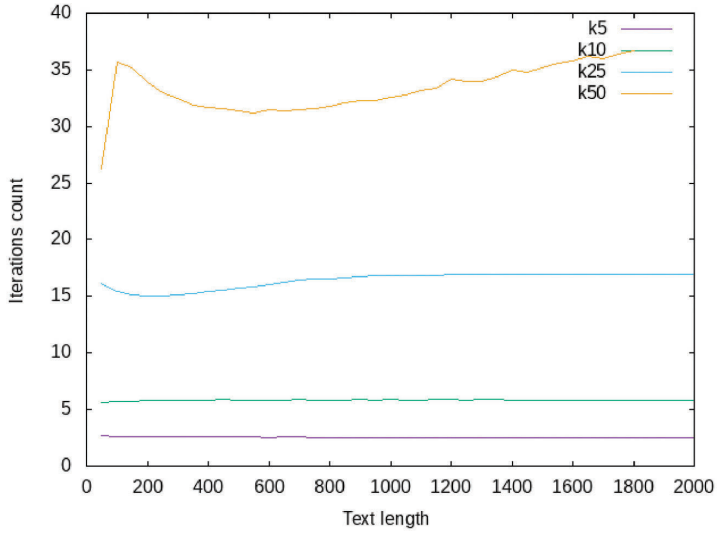

FIGURE 18. Average number of iteration count until the first local optima is reached for function 3; 1000 restarts; $k = 5, 10, 25, 50$.

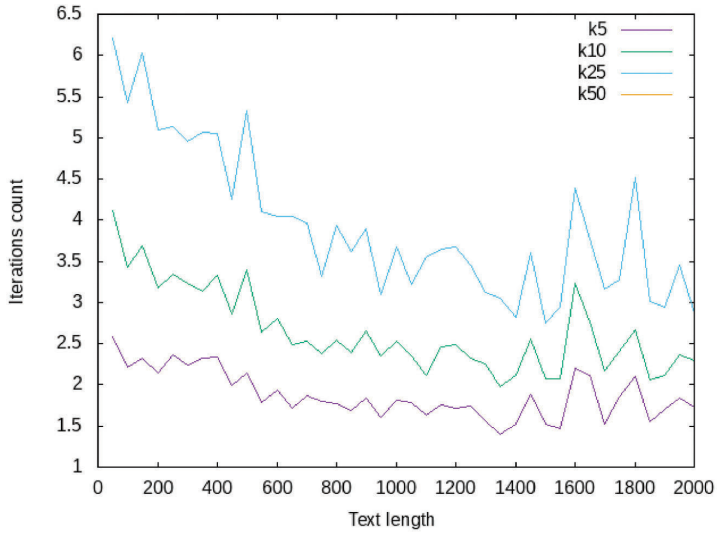

FIGURE 19. Average number of iteration count until the nearest local optima is reached (from the correct solution) for function 1; 1000 restarts; $k = 5, 10, 25, 50$.

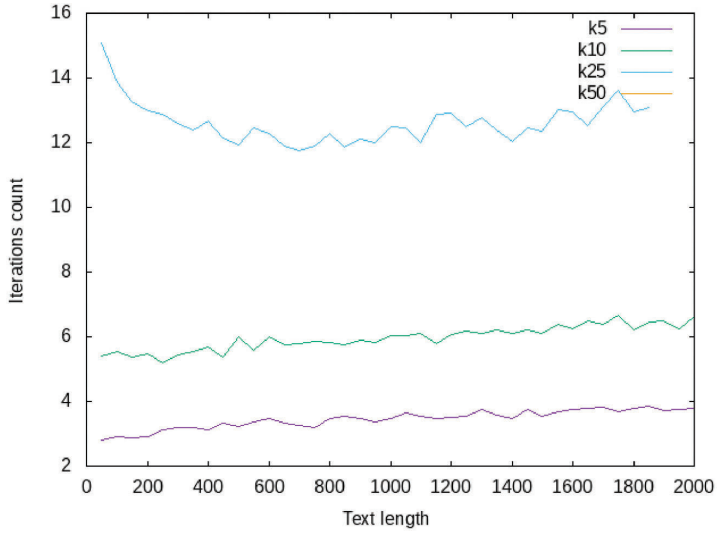

FIGURE 20. Average number of iteration count until the nearest local optima is reached (from the correct solution) for function 2; 1000 restarts; $k = 5, 10, 25, 50$.

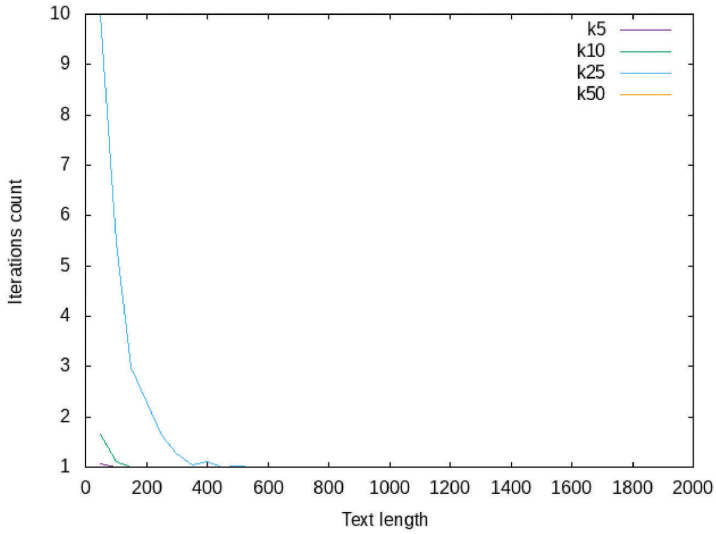

FIGURE 21. Average number of iteration count until the nearest local optima is reached (from the correct solution) for function 3; 1000 restarts; $k = 5, 10, 25, 50$.

REFERENCES

[1] ANTAL, E.: *Modern Cryptanalysis of Classical Ciphers*. PhD. Thesis, STU in Bratislave, 2017. (In Slovak)

[2] ANTAL, E.—ELIÁŠ, M.: *Evolutionary Computation in cryptanalysis of classical ciphers*, Tatra Mt. Math. Publ. **70** (2017), 179–197.

[3] ANTAL, E.—GROŠEK, O.—HORAK, P.: *On a mnemonic construction of permutations*, J. Math Cryptology. **11** (2017), no. 1, 45–53.

[4] BANKS, M. J.: *A Search-Based Tool for the Automated Cryptanalysis of Classical Ciphers*. The University of York, Department of Computer Science, May, 2008.

[5] CLARK, A. J.: *Optimisation Heuristics for Cryptology*. PhD. Thesis, Queensland University of Technology, 1998.

[6] GIDDY, J. P.—SAFAVI-NAINI, R.: *Automated cryptanalysis of transposition ciphers*, The Comput. Journal **37** (1994) no. 5, 429–436.

[7] FORSYTH, W. S.—SAFAVI-NAINI, R.: *Automated cryptanalysis of substitution ciphers*, Cryptologia, **17** (1993), n. 4, 407–418.

[8] JAKOBSEN, T.: *A fast method for the cryptanalysis of substitution ciphers*, Cryptologia, **19** (1995), n. 3, pp. 265–274.

[9] KULLBACK, S.: (Statistical Methods in Cryptanalysis). Aegean Park Press, Laguna Hills, California, 1976.

[10] KVASNIČKA, V.—POSPÍCHAL, J.—TIŇO, P.: *Evolučné algoritmy*. Vydavateľstvo STU, Bratislava, 2000.

[11] LASRY, G.: *A Methodology of Cryptanalysis of Classical Ciphers with Search metaheuristics*. PhD. Thesis, Kassel University, 2017.

[12] MATTHEWS, R. A. J.: *The Use of Genetic Algorithms in Cryptanalysis*, Cryptologia, **17** (1993), n. 2, 187–201.

[13] ÓLAFSSON, S.: *Metaheuristics*. In: *Handbooks in Operations Research and Management Science Vol. 13*, 2006, pp. 633–654.

[14] GANESAN, R.—SHERMAN, A.: *Statistical Techniques for Language Recognition: An Introduction and Guide for Cryptanalysts*. 1993, [cit: 2016-01-30]; `http://web.cecs.pdx.edu/~bart/decrypter/gs93.pdf`.

[15] REEVES, C. R.: *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Chapter: Fitness Landscapes. Boston, MA: Springer, New York, 2005, pp. 587–610.

[16] RUSSELL, M. D.—CLARK, J. A.—STEPNEY, S.: *Making the most of two heuristics: breaking transposition ciphers with ants*, In: CEC 03, Vol. 4, 2003, pp. 2653–2658.

[17] SPILLMAN, R.—JANSSEN, M.—NELSON, B.: *Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers*. Cryptologia, **17** (1993), no. 1, 31–44.

[18] VOBBILISETTY, R.—TROIA, F. D.—LOW, R. M.—VISAGGIO, A. C.—STAMP, M.: *Classic cryptanalysis using hidden Markov models*, Cryptologia **41** (2017), no. 1, 1–28; `http://dx.doi.org/10.1080/01611194.2015.1126660`

[19] *American National Corpus Project*, `http://www.anc.org/data/oanc/download/`

[20] *STU BA klaster - IBM iDataPlex*, `https://www.hpc.stuba.sk`

Received October 30, 2018

Institute of Computer Science and Mathematics
Slovak University of Technology
Ilkovičova 3
812 19 Bratislava
SLOVAKIA
E-mail: eugen.antal@stuba.sk