TECHNICAL TRANSACTIONS | CZASOPISMO TECHNICZNE
AUTOMATIC CONTROL | AUTOMATYKA
4-AC/2013

ALEX TORMÁSI*, LÁSZLÓ T. KÓCZY**

IDENTIFICATION OF THE INITIAL RULE-BASE OF A MULTI-STROKE FUZZY-BASED CHARACTER RECOGNITION METHOD WITH META-HEURISTIC TECHNIQUES

IDENTYFIKACJA POCZĄTKOWEJ BAZY REGUŁ W METODZIE ROZPOZNAWANIA WIELOLINIOWEGO PISMA ODRĘCZNEGO OPARTEJ NA LOGICE ROZMYTEJ Z WYKORZYSTANIEM TECHNIK METAHEURYSTYCZNYCH

A b s t r a c t

This paper summarizes the basic concept of the designed a fuzzy-based character recognition algorithm family and the results of the optimization of its rule-base with two various meta-heuristic methods, the Imperialist Competitive Algorithm and the bacterial evolutionary algorithm. The results are presented and compared with two other methods from literature after a short overview of the recognition algorithm.

Keywords: *fuzzy systems, character recognition, mate-heuristic optimization*

S t r e s z c z e n i e

W niniejszym artykule podsumowano podstawową koncepcję projektowania rodziny algorytmów rozpoznawania pisma odręcznego opartej na logice rozmytej oraz wyniki optymalizacji bazy reguł, z wykorzystaniem dwóch różnych metod metaheurystycznych: algorytmu ewolucyjnego ICA (*Imperialist Competitive Algorithm*) oraz ewolucyjnego algorytmu bakteryjnego. Przedstawiono krótkie podsumowanie algorytmu rozpoznawania pisma, a także wyniki porównawcze z dwoma innych metodami dostępnymi w literaturze.

Słowa kluczowe: *systemy rozmyte, rozpoznawanie pisma, optymalizacja metaheurystyczna*

* M.Sc. Alex Tormási, e-mail: tormasi@sze.hu, Department of Information Technology, Faculty of Engineering Sciences, Széchenyi István University, Györ.

** Prof. D.Sc. Ph.D. László T. Kóczy, Department of Automation, Faculty of Engineering Sciences, Széchenyi István University Györ; Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics.

1. Introduction

At present, there are several handwriting solutions used which are based on various techniques. Most of these techniques are also implemented on smart phones and tablet PCs, but the default text entry methods are still physical or virtual keyboards and only a few devices come with a stylus to support handwriting recognition.

There are several reasons why handwriting recognition techniques are still not used as main text entry methods, such as usability issues caused by the latency of the recognition as a result of the use of complex mathematical transformations. The other reason is the lack of flexibility of recognition methods; there are many alphabets and various writing styles which make recognition more difficult, and most of the methods are able to recognize only a few of these.

The most important property of handwriting recognizers is accuracy; the user-acceptance threshold is a 97% recognition rate, determined by LaLomia [1]. Recognition methods usually apply various complex geometric transformation methods on the input to reach this level of accuracy; even recent devices do not have the necessary hardware capacity to provide near real-time recognition with such complexity of these methods.

Researchers must deal with all these problems to design an acceptable, wide-spread handwriting recognizer. According to previous results, the properties of fuzzy logic [2] may provide an acceptable solution for the mentioned problems. Meta-heuristic methods might be able to optimize the fuzzy rule-bases [3, 4] to reach the user-acceptance threshold of the recognition system.

After the introduction, the basic features and concept of the Fuzzy-Based Character Recognition (FUBAR) algorithm-family are overviewed [5, 6, 7]. In section 3, concepts of the investigated meta-heuristics are summarized. In section 4, results are presented and analyzed for the optimization of a multi-stroke FUBAR rule-base. Results are compared to the accuracy of other recognition methods and the possible orientations of future work are also investigated in the last section.

2. Basic Concept of Multi-Stroke Fuzzy-Based Character Recognition

2.1. Features and Limitations

The following important key features were kept in mind during the development of the recognition engine:

1. Accuracy: The algorithm has to reach an acceptable recognition rate or at least the same as other accepted methods.
2. Efficiency: The designed methods must fit user's requirements in response time and in resources of the currently used hardware. This means that complex geometrical transformations and other mathematical functions should be avoided.
3. Flexibility of the alphabet: The model of the alphabet must be easily modifiable to support various alphabets and context-sensitive recognition.
4. Learning: The designed system should be able to learn user-specific writing styles.

The characteristics and properties of fuzzy systems are able to satisfy all the considered features, which led to the use of the fuzzy inference method for the recognition method. Fuzzy-Based Character Recognizer (FUBAR) is a family of algorithms of various single-stroke and multi-stroke hand printed (handwritten, non-cursive, capital letters) character recognition engines. The basic concept of the designed method is shown in Fig. 1 [5].

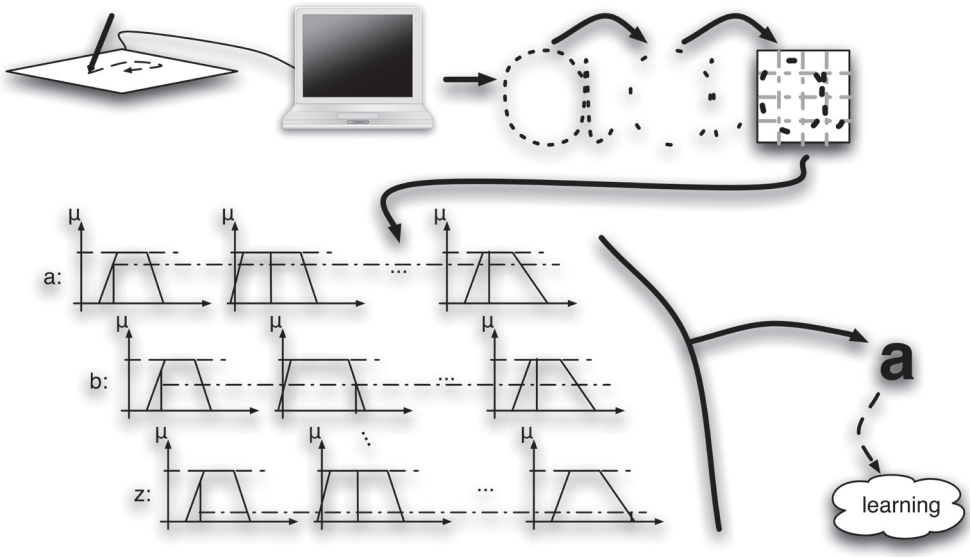

Fig. 1. The concept of FUBAR algorithms

The designed system is an online and personalized recognizer, which means that it processes digital ink and the recognition uses user-specific information.

2.2. Input Handling

The input signal of the algorithm consists of two-dimensional (x, y) coordinates in chronological order, representing the pen-movement (called (pen)stroke) as seen in Fig. 2.

In unistroke (or single-stroke) recognizers, letters are represented by one single stroke; in multi-stroke recognizers each symbol is represented by various numbers of strokes (sub-strokes). The FUBAR algorithms handle multi-strokes as one stroke by merging all the sub-strokes.

Usually, the received signal is non-continuous as a result of the bottlenecks of hardware, such as the bandwidth of interfaces and the available CPU resources. This causes information loss in recording the pen-movement; this information-loss causes difficulties in the processing because the positions of the missing coordinates are non-deterministic as seen in Fig. 3.

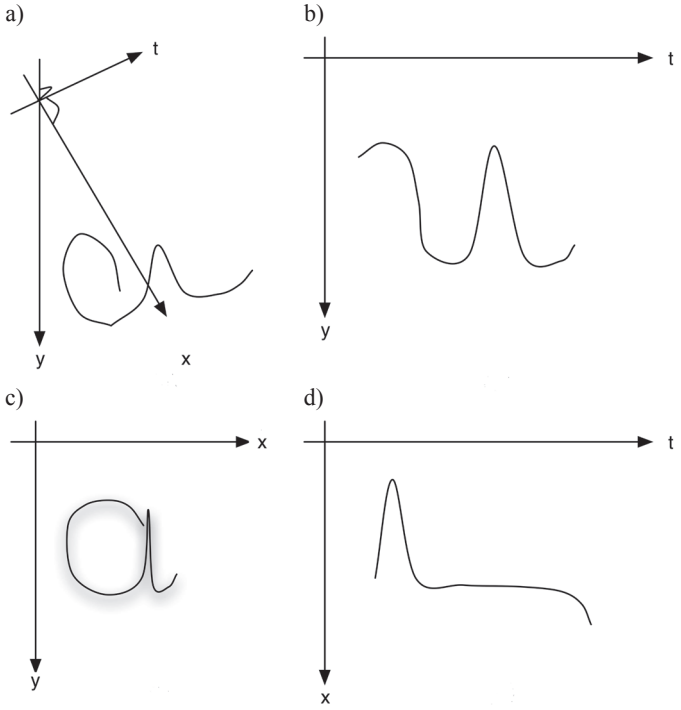

Fig. 2. A received 3-D stroke from various views: a) in 3-D, b) changes in the value of *y* by time, c) the 2-D projection, d) changes in the value of *x* by time

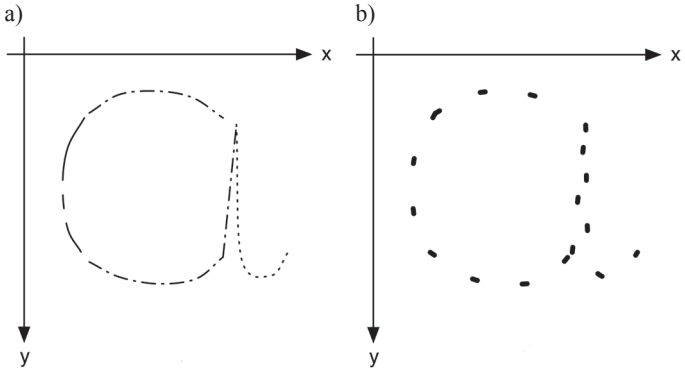

Fig. 3. Received stroke (a) and re-sampled stroke (b)

The received signal must be normalized for further processing and better recognition rate as seen in Fig. 3. In the FUBAR algorithm family, the points of the received signal are re-sampled, the points between a given (Euclidean-)distance from the reference point (the very first point of the stroke) are filtered out, the first and last points are always kept for reference. The following formula describes the re-sampling (filtering):

$$\underline{l'} = \{l_1\} \cup \left\{ l_j \middle| l \underset{\substack{j \in \{N_{n-1}-N_1\} \\ j>k}}{\arg\min_{j}} \left[\left| d\left(l_{k-1}, l_j\right) - \gamma \right| \right], k = \underset{i \in \{N_{n-1}-N_1\}}{\arg\min_{p}} \left[\left| d\left(l'_{i-1}, l_p\right) - \gamma \right| \right], i = \dim \underline{l'} \right\} \cup \{l_n\} \quad (1)$$

where:

- l – the received stroke,
- l' – the re-sampled stroke,
- l_j – a point in the stroke,
- γ – the threshold of distance between the kept points.

The re-sampling of the strokes also has an anti-aliasing property, which increases the degree of recognition as a result of the normalized stroke as seen in Fig. 4.

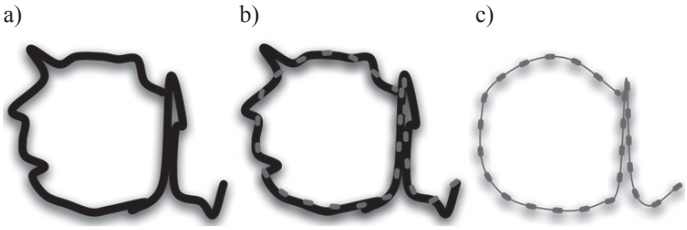

Fig. 4. Anti-aliasing properties of re-sampling: a) received stroke, b) received signal with selected points, c) selected points of the stroke

2.3. Character-Feature Extraction

FUBAR uses two kinds of stroke features for the recognition: 1) the width/height ratio of the stroke and 2) the average number of points in the rows and columns of the grid [5] drawn around the stroke.

The first member of the FUBAR family used crisp grid (classical grid with sharp borders) for the feature extraction, but the system reached a low average recognition rate as a result of the changing writing style of the test subjects. Some of the users started to write faster and use an italic writing style after creating a few samples, according to the collected data as seen in Fig. 5.

The sampled points of the strokes of oblique and normal characters could be located in completely different rows and columns of the grid. This caused huge overlap between the features of various letters. Other methods (like [18]) are rotating the input characters to avoid the negative effects of the italic writing style, but those methods are complex mathematical transformations, which would dramatically increase the computational complexity of the method.

Fuzzy grids were designed to resolve the problems caused by the italic writing style by the increased information provided by it. In fuzzy grids, the rows and columns of the grid are defined by fuzzy sets. It can be also considered as a transformation of the stroke into a fuzzy space.

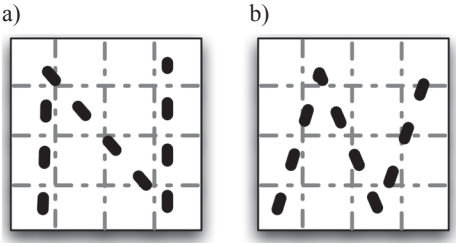

Fig. 5. Straight (a) and italic letters (b) in a grid

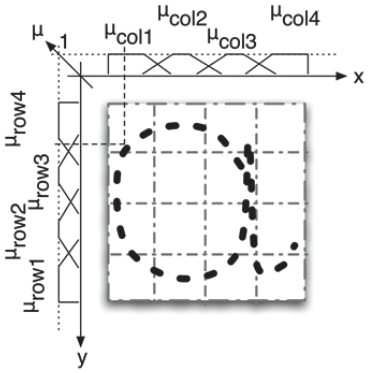

Fig. 6. Concept of the fuzzy grid

The optimal sizes of fuzzy grids were determined in [8, 9], which is 6 × 6 for single-stroke and 4 × 3 for multi-stroke letters. If the computational cost and recognition rate are both considered, then the optimal size of fuzzy grid for single-stroke letters is 6 × 4, while for the multi-strokes it does not change. The points in a fuzzy grid may belong to two different columns or rows at the same time with various membership values as seen in Fig. 6.

2.4. Inference

In the designed recognition engine a Takagi-Sugeno method [4] is used for inference. Each symbol in the alphabet is represented by a single rule associated to it. The input parameters of the rules are the features described in the previous sub-section. The output parameter of the rules is the degree of matching between the features of the input stroke and the stored rules as seen in Fig. 7, where R_i is the *i*-th rule, si is the degree of matching between the candidate stroke and the one represented by the *i*-th rule.

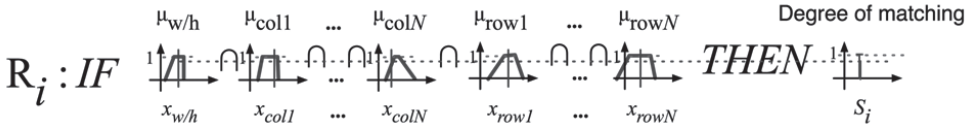

Fig. 7. A rule describing a letter

FUBAR returns with the character associated to the best matching rule (the one with the highest si value) after the rule evaluation phase. The initial rule-base was determined statistically from 60 samples per character, collected previously from test subjects.

There are two main ways to reduce the complexity of a rule-based fuzzy system: reducing the number of input parameters (antecedent) or reducing the number of rules evaluated. Hierarchical rule-bases are used to reduce the number of rules evaluated during the inference by partitioning the problem domain. Each partition contains a subset of the rules. Meta-rules are added to select which subset of the rules should be evaluated in a particular case; the input

parameters of these rules might be completely independent from the ones used in the original rules. Hierarchical rule-bases [10, 11] were used in some members of the FUBAR algorithm to decrease the computational complexity [12]. The average (fuzzified) number of points in the third row of a stroke was used as meta-rules in the single-stroke, while the number of strokes in the multi-stroke FUBAR.

3. The Applied Optimization Techniques

The initial rule-base for multi-stroke characters was determined the same way as described in [5] for single-stroke characters. The multi-stroke FUBAR algorithm reached 100% recognition rate for the 60 training sample per characters with the initial rule-base.

The fitness-function of each optimization algorithm was defined to maximize the average recognition rate for the training set. The optimization algorithms could not modify the initial rule-base because it had already reached the 100% accuracy. It does not mean that it can not be improved at all; it just reflects that the used initial rule-base is already optimal for the training set, so it can not be used during the meta-heuristic optimization and it must be changed to reach a lower recognition rate in order to start the optimization. The fuzzy sets in the initial rule-base for multi-stroke characters were modified with random values before the start of the optimization algorithms; the modified initial rule-base reached 79.17% recognition rate for the same training set and 74.7% for the validation sample set (120 samples per character) before the optimizations. The same randomly modified rule-base was used during each test.

Each entity (bacterium or country) in the algorithms represents a rule from the rule base. Each entity is coded into a vector, where the elements are the breakpoints of the fuzzy sets describing the antecedent of the rule. However, this technique requires validating the correctness of the new entities after any changes. The vector should be partially ordered for each represented membership function (it should maintain a trapezoid shape). Each algorithm had to be extended to use multiple populations (without migration), a further loop to say. These modified algorithms represent each letter by a single population, this was required to avoid the mixture between the data of various types of symbols.

Two different meta-heuristic algorithms were selected to optimize the modified initial rule-base, the Imperialist Competitive Algorithm (ICA) and the Bacterial Evolutionary Algorithm (BEA). Detailed description of Imperialist Competitive Algorithm can be found in [13] and of the Bacterial Evolutionary Algorithm in [14]. Maximizing the average recognition rate for the whole training set was used as a fitness function of the optimization algorithms, which means that the results were evaluated for all the strokes in the training set.

The ICA was inspired by politic and strategic processes instead of biological analogy. In the algorithm, countries are points in the search domain, the countries are determined randomly in the first step. Each country can be a conqueror or a colony, the strongest countries (those with the highest fitness values) are the conquerors.

The colonies are approaching the conqueror countries in the assimilation phase. The conqueror countries are fighting for the colonies, which are providing the possibility to increase their strength which is calculated by the aggregated fitness values of the empires and colonies. This assimilation phase is repeated until only one empire stands. The colonies can uprise against the conqueror countries (revolution phase), this step is included to prevent the algorithm from sticking in local optimums.

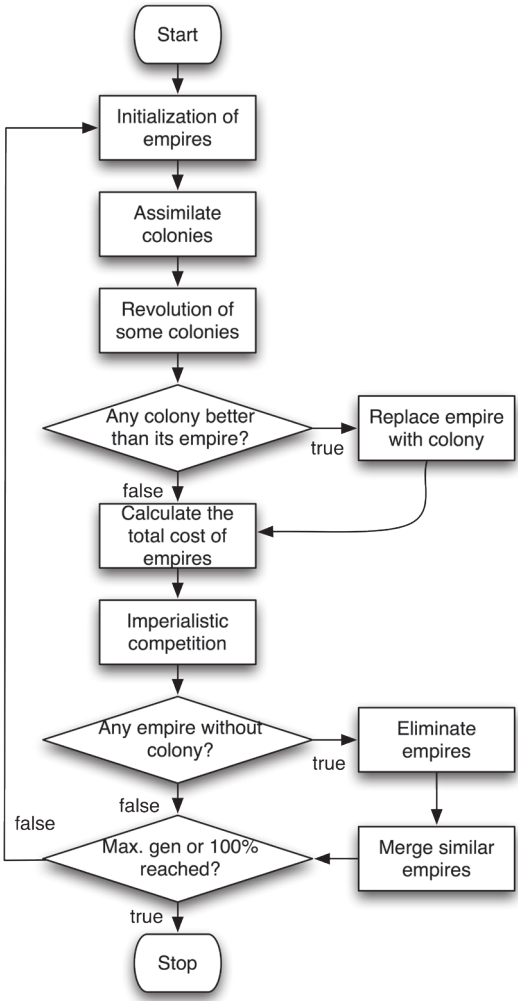

Fig. 8. The diagram of the Imperialist Competitive Algorithm

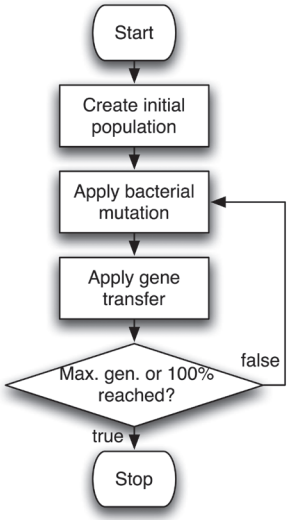

Fig. 9. The diagram of the Bacterial Evolutionary Algorithm

The BEA is based on the evolution process of bacteria. Each bacterium represents a point in the search domain. The first step of the algorithm is the bacterial mutation, which is applied to each bacterium individually. Each selected bacterium is cloned in this phase, then each allele of the clones are changed (only one at a time). The clone (or the original bacterium) with the best fitness value transfers its allele to the other clones.

The mutation phase is followed by the gene transfer (or infection) step. In this step, the bacteria population is divided into the groups of good and bad bacteria by their fitness values. A randomly selected good bacterium transfers a randomly selected allele to a randomly selected bad bacterium. All steps are repeated until the algorithm reaches the maximal number of generations.

4. Results of the Applied Optimization Techniques

The parameters of the bacterial evolutionary algorithm are the number of clones, the number of infections and the number of generations. The parameters of the imperialist competitive algorithm are the number of countries, the imperialist and the revolution factor.

The parameters of optimization algorithms were set according to preliminary results from formerly published works (where the single-stroke FUBAR was optimized), the ones with the best results are presented in this paper. The number of countries was constantly 120 and the number of imperialists was constantly 48 for the ICA, while the revolution factor was varying between 10 and 14 (steps were 0.2). All the parameters, number of clones, maximum generations and the number of infections were set to 10 in each BEA experiments.

The two best recognition rates achieved with the ICA for the validation set are 93.48% and 94.1%, as you may see in Fig. 10, the first average recognition rate was achieved with 120 countries, 48 imperialists and 12.4 revolution factors in 2 generations. The second result was reached by 120 countries, 48 imperialists and a revolution factor of 12.6 in 1 generation.

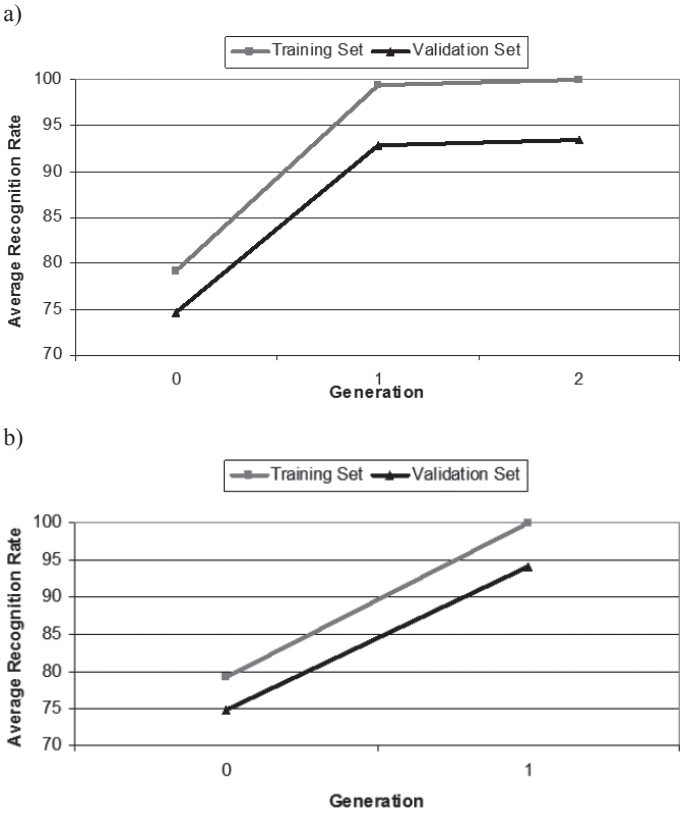

Fig. 10a) the second highest average recognition rate after the imperial competitive optimization, b) the highest average recognition rate after the imperial competitive optimization

The best two results of BEA are average recognition rates of 96.65% and 97.54% in 3 and 4 generations, with the same parameter settings, which are 10 clones, 10 infections and 10 generations as seen in Fig. 11.

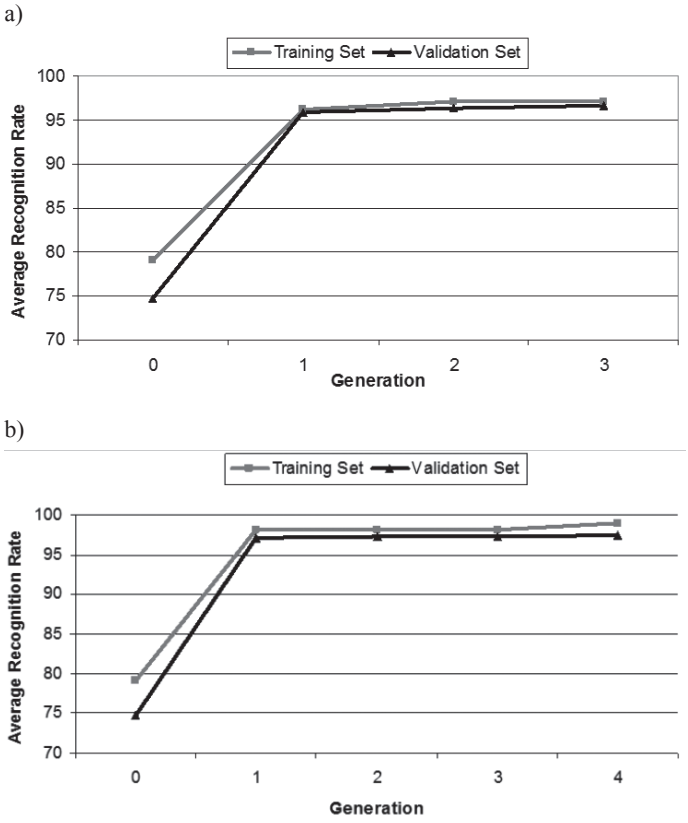

Fig. 11a) the second highest average recognition rate after the bacterial evolutionary optimization, b) the highest average recognition rate after the bacterial evolutionary optimization

The best accuracy reached by the ICA decreased the average error rate by only 10.6%. The highest average recognition rate of 97.54% was achieved by the BEA, which is a 62.72% decrease in the average error rate.

The average of the results from the experiments with the same parameters for BEA is 96.145%. There is no significant variance between the results for ICA with the previously described parameters; the average of the results is 93.67%.

It is also important to highlight the fact that only the ICA could reach 100% average recognition rate for the training set during the optimization in the first few (1–5) generations, while the bacterial evolutionary algorithm could not, the BEA finished only after the number of the generations reached the maximum. It is important, because it reflects that ICA is able to find (local)optimum solutions faster.

Despite the better convergence and lower resource consumption (computational cost), the imperialist competitive algorithm reached a lower increase in the recognition rate compared

to the bacterial evolutionary algorithm, which has a moderate convergence (compared to the imperialist competitive algorithm), a higher computational cost and a much higher increase in the accuracy.

The BEA may give better results, but it takes more time to compute the results. While BEA can process 0.0667 generations, ICA can evaluate 0.667 generations in a second. Both algorithms used 99% of 7 CPU cores from 8 during the experiments.

5. Conclusion and Future Work

The imperialist competitive algorithm had a better convergence during the optimization of the recognizer's rule-base, while the BEA reached the highest average accuracy (97.54%) for the validation sample set. The achieved average recognition rate reaches the 97% user acceptance threshold.

The most similar recognizer to the FUBAR is the Palm's Graffiti2 [15] and the $N recognizers [16]. The Graffiti2 is the multi-stroke version of the Palm's Graffiti unistroke recognizer [17]. The accuracy of the algorithm was determined at 86.03% by Költringer and Grechenig.

The $N recognizer is the multi-stroke successor of the $1 single-stroke recognizer [18], which was developed by J.O. Wobbrock, A.D. Wilson and Y. Li. The $N was developed by L. Anthony and J.O. Wobbrock.

The FUBAR algorithm with multi-stroke support reached a higher average recognition rate with 26 different characters after the optimization, compared to the Graffiti2 algorithm, which reached 86.03% accuracy and supports only 3 multi-stroke symbols, and compared to the $N algorithm, which achieved a 96.7% average recognition rate for 16 single-stroke symbols, as seen in Fig. 12.

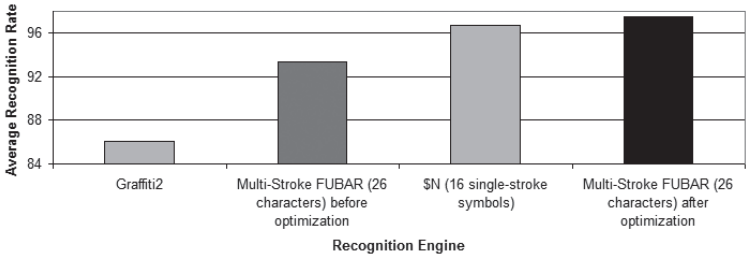

Fig. 12. Average recognition rates of various multi-stroke recognition engines

The in-depth analysis of these and other meta-heuristic techniques in model identification is important. Much more detail and further directions should be investigated such as a wider range of parameter values and as other type of rule-bases.

Future research includes the investigation of modeling symbols with multiple rules in the rule-base, this might increase the recognition rate by supporting various types of the same symbols. The increased number of the rules in the rule-base causes higher resource

requirements by the algorithm. The hierarchical structure of the fuzzy rule-base might reduce the resource requirements without any significant reduction in the average accuracy of the FUBAR algorithms.

The support of off-line character recognition will be included in the FUBAR algorithm family to extend the application areas of the method such as form processing.

Cursive handwriting is more general (especially in everyday uses), which makes its support a high priority task in the near future. The segmentation of characters must be also included in the FUBAR algorithms before the support of cursive writing recognition.

The FUBAR algorithm will be integrated into the HandSpy system [19], which is a collaborative environment for managing experiments in the cognitive processes in writing.

This paper was supported by the Hungarian Scientific Research Fund (Hungarian abbreviation: OTKA) K105529, K108405 and TÁMOP-4.2.2.A-11/1/KONV-2012-0012.
The used Meta-Heuristic algorithms were implemented by Márton Hevér as a part of his Master thesis (thesis advisor: Alex Tormási).

References

[1] LaLomia M.J., *User acceptance of handwritten recognition accuracy*, Companion Proc. CHI ’94, New York 1994, 107.
[2] Zadeh L.A., *Fuzzy sets*, Inf. Control, 83, 1965, 338-35.
[3] Mamdani E.H., Assilian S., *An experiment in linguistic synthesis with a fuzzy logic controller*, International Journal of Man-Machine Studies, Vol. 7, 1975, 1-13.
[4] Takagi T., Sugeno M., *Fuzzy identification of systems and its applications to modeling and control*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-15, 1985, 116-132.
[5] Tormási A., Botzheim J., *Single-stroke character recognition with fuzzy method*, New Concepts and Applications in Soft Computing SCI, Vol. 417, V.E. Balas et al. (eds.), 2012, 27-46.
[6] Tormási A., Kóczy T.L., *Improving the Accuracy of a Fuzzy-Based Single-Stroke Character Recognizer by Antecedent Weighting*, Proc. 2nd World Conference on Soft Computing, Baku 2012, 172-178.
[7] Tormási A., Kóczy T.L., *Fuzzy-Based Multi-Stroke Character Recognizer*, Preprints of the Federated Conference on Computer Science and Information Systems, Kraków 2013, 675-678.
[8] Tormási A., Kóczy T.L., *Comparing the efficiency of a fuzzy single-stroke character recognizer with various parameter values*, Proc. IPMU 2012, Part I. CCIS, Vol. 297, S. Greco et al. (eds.), 2012, 260–269.
[9] A. Tormasi, and Kóczy T.L., *Efficiency and accuracy analysis of a fuzzy single-stroke character recognizer with various rectangle fuzzy grids*, Proc. CSCS ’12, Szeged 2012, 54-55.
[10] Sugeno M., Griffin F.M., Bastian A., *Fuzzy hierarchical control of an unmanned helicopter*, Proc. IFSA ’93, Seoul 1993, 1262-1265.
[11] Kóczy T.L., Hirota K., *Approximate inference in hierarchical structured rule-bases*, Proc. IFSA ’93, Seoul 1993, 1262-1265.

[12] Tormási A., Kóczy T.L., *Improving the Efficiency of a Fuzzy-Based Single-Stroke Character Recognizer with Hierarchical Rule-Base*, Proc. 13th IEEE International Symposium on Computational Intelligence and Informatics, Óbuda 2012, 421-426.
[13] Atashpaz-Gargari E., Lucas C., *Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialistic competition*, Proc. 2007 IEEE Congress on Evolutionary Computation, 7, Singapore 2007, 4661-4666.
[14] Nawa N.E., Furuhashi T., *Fuzzy system parameters discovery by bacterial evolutionary algorithm*, IEEE Transactions on Fuzzy Systems, 7(5), 1999, 608-616.
[15] Költringer T., Grechenig T., *Comparing the Immediate Usability of Graffiti 2 and Virtual Keyboard*, Proc. CHI EA'04, New York 2004, 1175-1178.
[16] Anthony L., Wobbrock J.O., *A Lightweight Multistroke Recognizer for User Interface Prototypes*, Proc. GI 2010, Ottawa 2010, 245-252.
[17] Fleetwood M.D. et al., *An evaluation of text-entry in Palm OS – Graffiti and the virtual keyboard*, Proc. HFES'02, Santa Monica, CA, 2002, 617-621.
[18] Wobbrock J.O., Wilson A.D., Li Y., *Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes*, Proc. UIST '07. ACM Press, New York 2007, 159-168.
[19] Monteiro C., Leal J.P., *Managing experiments on cognitive processes in writing with HandSpy*, Computer Science and Information Systems, Vol. 10, No. 4, Novi Sad 2013, 1747-1773.