The 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
Vol. 2, pp. 590-595, September 21-23, 2017 Bucharest, Romania

Increasing Speed of Genetic Algorithm-based Instance Selection

Mirosław Kordos and Aleksandra Kłos-Witkowska
Department of Computer Science and Automatics,
University of Bielsko-Biala, Poland,
corresponding author's e-mail: mkordos@ath.bielsko.pl

***Abstract* – We discuss several ways to accelerate genetic algorithm-based instance selection, where the two objectives are a minimal number of training instances and maximal accuracy of the classifier (we use neural networks) on the test data. We discuss several ways to accelerate the process, but we especially focus on two parameters: fitness function and chromosome length reduction. We evaluate different fitness functions, discuss their performance and propose the guidance for choosing the optimal one in respect to the process speed and stability. We also discuss the possibility of reducing the chromosome length during the optimization by excluding from further optimization these positions that are unlikely to change. We verify our method experimentally as well on several real-world datasets.**

***Keywords* – Genetic algorithms, fitness function, instance selection**

I. Introduction and State of the Art

A. *instance selection*

Good data preprocessing, including data selection is frequently the most important part of the prediction process, even more important than the optimal model parameters. This is because the models can be only as good as the data used for their learning. The list of data selection purposes begins with reduction of the data size and noise elimination, which both improve the data quality. They allow for achieving higher accuracy, shorter learning time and lower complexity of predictive models as well as easier data interpretation.

Data selection comprises feature selection and instance selection. Much more research has been done so far in feature selection. Since feature selection and instance selection are equally important. To bridge the gap our research focuses on instance selection.

There are two objectives in instance selection (which are sometimes contradictory): to obtain a low number of instances (vectors) in the training set and to obtain high classification accuracy. Thus the classical instance selection methods fall into two categories: noise filters and condensations methods. A good idea is to apply noise filter first and then a condensation algorithm. Description and experimentally evaluation of many instance selection methods can be found in [1]–[3].

In most of the instance selection methods, the decision is based on some distance or neighborhood concept. Further the input space can be divided into clusters to achieve still higher speed, but also to ensure good local data representation by the selected instances [4].

A frequently used noise filter is ENN (Editted Nearest Neighbor) [5]. ENN uses k-NN to predict the class of each instance and marks the instances for which the predicted class is different than the real class. In the next step the marked instances are removed from the training set, as they are considered noise. The data size reduction obtained by ENN is usually very little at the level of 10% and higher compression indicates poor data quality.

Much more instances can get removed because of their redundancy and their irrelevance for the classification process. However, removing too many instances from this group can result in decrease of the classification capabilities of the models trained on the reduced dataset.

DROP-3 and the used in our experiments IB3 belong to the best classical condensation algorithms. IB3 (Instance Based Learning v.3) [6] first selects the instances misclassified by k-NN (as the correctly classified instances are believed not to provide any additional information) and then it removes from the selected set each instances if it does not cause accuracy loss.

Instance selection finds application in almost every domain where the problem can be expressed by data and many of the classical methods can be also adjusted to regression tasks [7], [8]. We have already successfully applied our research on instance selection in metallurgical problems and we are currently considering an application in optical immunosensors [9], [10].

Usually higher compression can be achieved for bigger datasets. A linear complexity method has been proposed in [11], what makes the application quite well adapted to big data also in terms of computational complexity.

First were developing classical [8], [12] and neural-network embedded [13], [14] instance selection methods. But once we tried an evolutionary approach, we were astonished by the excellent results, but equally disappointed with the computational cost.

B. genetic algorithms

Genetic algorithms (GA) can be applied to many optimization problems. Their good description can be found in [15]–[17]. We will discuss their application to instance selection. The main advantage of genetic algorithms and other evolutionary optimization methods is their ability to exploit the data space in a much detailed level than traditional methods based on gradient or search heuristics and thus to find better solutions. However, their main disadvantage is the computational cost, which is several order of magnitude higher than that of classical instance selection methods. Our goal is to keep the high quality of the solutions obtained with genetic algorithms but to decrease the computational cost as far as possible.

Genetic algorithms use a population of individuals (binary vectors) of the length equal to the number of instances (vectors) in the original dataset. Each position at the individual denotes one instance from the dataset. 1 (or true) at this position means that the corresponding instance is selected and 0 (or false) means that it is rejected from the final dataset.

Genetic algorithms iteratively use the following operations:

- calculation of the quality (fitness function)
- selection of parents with probability proportional to their fitness function, of recombination points and creating offspring
- evaluating the quality (fitness) of the parents and children and deciding which individuals will enter into the next generation
- applying mutation with some probability by randomly changing single positions in the chromosome of randomly selected individuals

Depending on the version of genetic algorithms, particular operations can be performed in different ways. Here we shortly describe the operations in the Generational GA, Steady State GA and CHC GA. In the next section we introduce our improvements based on the three methods, where our main goal is to shorten the optimization time or (what is frequently equivalent) to obtain a better results (in terms of classification accuracy and data compression) in the same amount of time.

The next division of genetic algorithms is into single-objective and multi-objective. In multi-objective optimization the aim is to find the Pareto-front of non-dominated solutions, what corresponds to a simultaneous optimization for various α parameters in equations 1-5 [18], [19]. We are currently investigating the usability of multi-objective evolutionary optimization (the objectives are dataset size and classification accuracy) to instance selection. Our preliminary research shows that this can be a good solution for instance selection in regression problems. However, in this work we use single-optimization version, because in most of the experiments presented in this paper, the reasonable Pareto front was very short and sometimes it was reduced to a single point (changing α very little influences accuracy - see Table. 1).

It was verified [20], [21] that the steady state genetic algorithms display faster convergence than generational ones. The explanation of that fact is that in steady state algorithms, the offspring immediately replaces the worst individual in the population, what at this moment makes the population better as a whole, while in generational genetic algorithms we have to wait for that improvement until the next iteration.

In CHC [22] genetic algorithms the parent population is used to generate the intermediate population and then the best individuals from both population enter the next population. This is to some degree an extension of the elitism principle in the generational GA, when some number of the best individuals (the elite) from the parent population enter the next population. CHC uses a different recombination operation, so called HUX, which exchanges half of the bits that differ between parents, with the crossover point being randomly chosen. However, if the selected parents are too similar, the recombination is not performed. In CHC the mutation is applied after the recombination phase and the mutation is usually quite strong (up to 35%, comparing with the typical mutation rates of about 1% in classical generational GA). The CHC used for instance selection in [23] proved to be a very efficient method.

The method we use combines the best features of generational, steady-state and CHC genetic algorithms. The first difference between classical and evolutionary instance selection is that evolutionary algorithms optimize globally the whole training dataset, unlike the classical methods, which deal with each instance separately. The optimization can be either single-objective or multi-objective, binary or real-value in cease of instance weighting [19], [24].

There have been some propositions in the literature to use genetic or evolutionary algorithms for instance selections [25]–[27]. Evolutionary optimization is usually able to find the subset of instances, which is much more closer to Pareto-optimal or even is Pareto-optimal in the terms of compression and accuracy. Classical methods are unable to obtain such good results.

II. The proposed solution

A. The starting point: our previous work

In our previous work [28] we analyzed the influence of population size and crossover scheme on the genetic algorithms convergence applied to instance selection. Our general conclusions were that the population size of about 100 individuals and the multi-parent, multi-point reproduction scheme, where the number of parents is about 1/10th of the chromosome length were close to optimal parameters for a broad range of the instance selection for the datasets from 200 up to 20,000 instances. The population initialization is an important issue in genetic algorithms and especially the hybrid methods composed of more than one initialization scheme are efficient [29]. Thus in the instance selection based on evolutionary algorithms, our optimization starts from weights randomly distributed around the instances selected by a classical instance selection method. That allowed to start from a point corresponding to about 1/3 of the whole optimization. Next if the k-NN algorithm is used for instance evaluation, the distances between training and test set instances can be cashed, not calculated every time. The distances are sorted and matrices with the classes of the corresponding training set instances are stored and provided for k-NN if the for the first kk selected instances. The caching does not change the number of fitness function evaluations, but makes calculating the fitness function value much faster.

B. Search space reduction

In this paper we propose two further improvements to the instance selection with genetic algorithms, which also can be applied to other optimization problems. The first proposition is reducing the chromosome length based on the majority voting at a certain point of the optimization and the second one is selection of the optimal fitness function. The purpose of both is to make the optimization faster. The whole instance selection process is shown in the pseudocode in Algorithm 1.

It can be observed that because of the decreasing diversity in the population during the optimization progresses, there are positions in the chromosome, at which after some number of iterations there is the same value (0 or 1) in almost every individual. If the optimization continues, the positions that take at this moment the same values in the great majority of individuals will finally take the majority value with a very high probability (a prevailing most of them in most runs of the experiments did). Thus this positions can be at this moment assigned the majority values and removed from the further optimization. That will leave us fewer parameters to optimize and thus the further optimization will be faster. This is depicted in the example below:

```
ind1 0010101010101010010
ind2 0001101010101010110
ind3 0111011001111011110
ind4 0010101010101010010
ind5 1010101010101010010
ind6 0001111110011110000
ind7 0010101010101010010
ind8 0011011111101110110
vote 00????1?1???1?10??0
```

The number in the last row shows the majority position (in our case the position represented in at least 7 out of the 8 individuals). If neither 0 nor 1 is present in at least 7 individuals on the same position, then this position is denoted by a question mark. Thus only the question mark positions will be further optimized and the chromosome can be shortened at this moment. This reduces the number of further iterations as well as the computational complexity of a single iteration (fewer parents and fewer crossover points have to be generated to produce offspring). Also the population size can be decreased, because the optimal population size depends on the chromosome length. There are two parameters, which we adjusted experimentally: the iteration at which the reduction is performed and the voting threshold to eliminate a given position. Even if it could introduce occasionally some local minimum, it is not a great concern, because the difference is only in one or a few instances and there is a little chance that this affects the classifier. Moreover, this feature can be switched off, if any possible loss of classification accuracy is unacceptable.

Algorithm 1 The genetic instance selection

```
run the ENN followed by IB3 instance selection to get the set $S$ of selected instances
if Instance Selection then
  for $i = 0 \ldots populationSize$ do
    for $v = 0 \ldots originalNumVectors$ do
      $initialPopulation[i][v] = 1$ with probability $p1 = 0.1$ if the instance $v$ is in not $S$
      $initialPopulation[i][v] = 0$ with probability $p2 = 1 - p1$ if the instance $v$ is in $S$
    end for
  end for
else
  generate initial $currentPopulation$ of $P$ individuals
end if
for $n = 0 \ldots numIterations$ do
  calculate fitness for $newPopulation$ individuals
  adjust the fitness function to the standard deviation
  apply the multiparent multipoint crossover operation to generate the $newPopulation$ of $N$ individuals $(N <= P)$
  if $diversity < Threshold$ and $reducedSEarchSpace = false$ then
    Reduce the search space
    $reducedSEarchSpace = true$
  end if
  if the best solution is kown to be found then
    STOP
  end if
  sort together $currentPopulation$ and $newPopulation$ individuals by fitness
  select the best $P$ individuals into $currentPopulation$
  apply the mutation operation
end for
```

C. *Fitness function*

As the genetic optimization progresses, the variability among individuals is getting smaller and thus in order to promote the best individuals the fitness function should get steeper. On the other hand too steep fitness function at the beginning of optimization can cause that only the best individuals will take part in generating the offspring and thus the genetic diversity will shortly be very limited with possibly lacking the good chromosome regions that were contained in the generally poor individuals. We designed several different fitness functions, to find out how they perform and to propose the optimal solutions depending on the optimization progress.

In equation (1) the expression before raising to the power p is simply proportional to the accuracy A obtained on the test set and inversely proportional to the number if instances Vin the training set. The average accuracy for the population $avgA$ and average number of vectors $avgV$ only play the role of a normalizing factor. So this is the most intuitive approach.

$$fitness = \left(\alpha \frac{A}{avgA} + (1-\alpha)\frac{avgV}{V}\right)^p \tag{1}$$

In equation (2) the expression in the brackets itself makes the fitness function steeper, as now the individual with the minimal accuracy $minA$ has a fitness of zero in the accuracy part and the individual with the maximum number of vectors $maxV$ has a fitness of zero in the vector part.

$$fitness = \left(\alpha \frac{A - minA}{stdA} + (1-\alpha)\frac{maxV - V}{stdV}\right)^p \tag{2}$$

Equation (3) presents the strongest selection, as only the individuals with the quality above the average will have non-zero fitness. Also other cut-off points, situated wherever between zero and $avgA$ (eq. 3) can be used. However, we limited the experiments to the three cases, because it seemed sufficient for drawing the conclusions.

$$\begin{aligned} fitness = \Big(\alpha \cdot max(0, \frac{A - avgA}{stdA}) \\ +(1-\alpha) \cdot max(0, \frac{avgV - V}{stdV})\Big)^p \end{aligned} \tag{3}$$

The exponent p (eq. 4 and 5) can be determined in various ways and we used two approaches. In both approaches it consisted of the constant part $p1$ and the variable part $p2$. In particular case $p1$ or $p2$ can be set to zero in eq. (4) and (5). In equation (4) the variable part increases linearly during the optimization, where i denotes the current iteration:

$$p = p1 + p2\frac{i}{numIterations} \tag{4}$$

In eq. (5) the variable part depends on the speed at which the standard deviation of the weighted sum of accuracy and the inverse of the number of vectors $\left(\alpha\frac{A}{avgA} + (1-\alpha)\frac{avgV}{V}\right)$ decreases, where $stdAV(i)$ is the standard deviation in the current $i - th$ iteration and $stdAV(i-1)$ and $stdAV(i-2)$ in one and in two iterations ago. This ensures that the fitness function steepness is close to proportional to the standard deviation of the population quality.

$$p = p1 + p2\frac{0.66 \cdot stdAV(i-1) + 0.33 \cdot stdAV(i-2)}{stdAV(i)} \tag{5}$$

In the experiments we evaluated 12 combinations (three formulas for the base, each with two formulas for the exponent), each with various $p1$ and $p2$ values and each with two α values ($\alpha = 0.96$ and $\alpha = 0.90$) for instance selection. We also performed the tests for randomly generated data and in this case the we just used $\alpha = 1$ and A represented the number of bits which are equal in the individual and in the target solution.

III. Advantages and disadvantages of our method

The advantage of our solution is the shorter optimization. When the time is a constraint and the problem is big and we must chose some point of stopping the optimization, then our method can find a better solution within the time limit.

The disadvantage of our method is higher complexity of the algorithm by the application of the chromosome length reduction, but that is not so complex that it could be an obstacle in implementing this. There is a very little chance that some position that should not be removed from the further optimization will get removed and the best solution will be missed. In order to minimize the chance, the selection parameters should be properly tuned. We cannot see any disadvantage of applying the proposed fitness function, as the application is very simple and it is quite stable and small changes in the parameters make also very small changes to the optimization process.

IV. Experimental Results

We performed the experiments on 10 classification datasets from the Keel Repository [30]: Ionosphere (351,33,2), Image Segmentation (210, 18, 7), Magic (19020, 20, 2), Thyroid (7200, 21, 3), Page-blocks (5472, 10, 5), WDBC (569, 30, 2), Sonar (208, 60, 2), Satellite Image (6435, 36, 6), Penbased (10992, 16, 10), Pima(768, 8, 2). The numbers in the brackets are (number of instances, number of features, number of classes). To perform the experiments we created software that can be obtained from *www.kordos.com/idaacs2017*.

The experiments with instance selection were run in 5-fold crossvalidation with two different values of the α parameter: 0.96 and 0.90 (eq. 1-3). The stopping criterion was instance reduction rate at the best individual on the training set slower that 0.2% of the original instances for three consecutive iterations (the maximum accuracy was always achieved by that time and after that point over-fitting starting to occur and the accuracy on test set stated to drop). The mutation probability was linearly increasing during the optimization from 0.002 to 0.02 for each position in each individual, as this was experimentally determined as close to the optimal probability.

In k-NN used for fitness function determination we used $k = 3$ or $k = 1$ if there were too few instances to use $k = 3$. For the final prediction we used two different classifiers: neural network and k-NN. The MLP neural network was trained for 30 epochs with the R-prop algorithm [31]. That was usually the optimal number of epochs before over-fitting starts to occur. We tried also some other neural network learning algorithms, but there was no noticeable difference between the performance of them, so we decided to use only Rprop for the experiments. We used networks with one hidden layer. The numbers of neurons in the hidden layer was set to the rounded up geometric mean of number of attributes (inputs) and number of classes (outputs of the network). We used a population of 96 individuals. This choice was dictated by the fact that there were 24 physical cores in our computer and 96 is a multiply of 24 so all the processing power could be effectively used in parallel computations.

The precise determination of how many times longer it took to perform evolutionary than classical instance selection depends on many implementation details and at some conditions it can be even faster (for instance DROP is $O(n^2)$) For example, for the magic dataset calculating the distance matrix for k-NN took as many times as 2000 evaluations of the fitness function with reading the classes from the sorted arrays (this is about 21 epochs). However, the CPU utilization while calculating distance matrices was 100% and while calculating the fitness functions below 30% due to thread synchronization issues. That however likely can be improved with better implementation. The computational complexity of ENN+IB3 is of the order of $O(n^2)$ - the same as calculating k-NN distances, where n is the number of instances. There also exist some methods of increasing k-NN speed (at the cost of possible precision loss), that were not used in this work.

We experimentally determined that the chromosome length reduction should take place when at least at 60% of positions, the majority vote ir at least 95% that is of 91 the same votes for the population size of 96 individuals. In case of instance selection it allowed for an average 3-fold chromosome length reduction because of the data sparsity (that is only a few instances need to be selected and there should be zeros on prevailing number of positions), especially for bigger datasets and it allowed to shorten the total optimization time by about 25% - 35% depending on the dataset size.

dataset	α	cI	cA	agI	agA	nf
	0.96	15.2	84.4	13.5	86.8	2980
Ionosph	0.90			13.5	86.8	±220
	0.96	13.8	91.0	13.8	91.6	2870
ImSegm	0.90			13.7	91.5	±120
	0.96	3.91	82.5	2.65	84.5	13200
Magic	0.90			1.97	83.3	±2300
	0.96	2.87	93.0	1.80	96.8	12200
Thyroid	0.90			1.32	96.1	±1100
	0.96	2.29	93.6	1.83	96.6	23900
PageBlk	0.90			0.98	96.3	±1700
	0.96	12.0	97.0	9.43	97.7	3470
WDBC	0.90			6.66	97.1	±80
	0.96	27.6	80.2	15.6	82.7	2100
Sonar	0.90			11.3	82.1	±10
	0.96	7.47	82.6	4.67	82.5	14800
SatImg	0.90			2.12	68.2	±1200
	0.96	3.60	90.3	3.20	91.3	11400
Penbas	0.90			3.25	88.2	±900
	0.96	6.76	95.3	3.56	97.8	3890
Pima	0.90			3.56	97.8	±200

Table I. EXPERIMENTAL RESULTS WITH INSTANCE SELECTION, CI, AGI - PERCENTAGE OF SELECTED INSTANCES BY ENN+IB3 AND EVOLUTIONARY METHOD RESPECTIVELY, CA, AGA - CLASSIFICATION ACCURACY OF MLP NETWORK TRAINED ON THIS SELECTION, NF - NUMBER OF FITNESS FUNCTION EVALUATIONS.

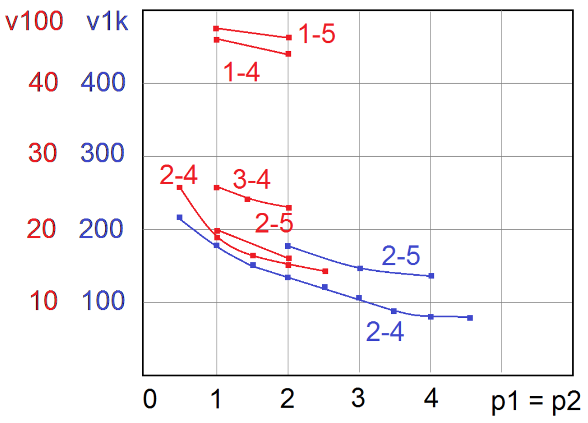

Figure 1. How the number of fitness function evaluation on vertical axis for 100 (red) and 1000 (blue) chromosome length) depends on the exponents (eq. 4, 5) of the fitness function for different functions (the numbers, e.g 2-4 show the number of equations with the function formula: first number:accuracy, second:exponent).

As it can be seen from Fig. 1., the best fitness function is given by eq. 2 and the exponent p by equation 4. For longer chromosomes higher exponents are better than for shorter. However, too high exponents are not stable and the process may not converge at all. For that reasons the lines 2-4 (in Fig. 1.) are not shown further to the right as it was already in the unstable area.

V. CONCLUSIONS

There are several factors, which influence the speed of convergence of genetic optimization. In this work we discussed two ways to accelerate the instance selection with genetic algorithms: optimized fitness function and chromosome size reduction, which allowed us to significantly accelerate the process. The accuracy shown in Table 1 is the same as we obtained in our previous work, but the optimization time is on average cut by half (more for larger datasets due to stronger chromosome size reduction).

In the future work we are going to extend the research by: including joint instance and feature selection as well for classification as for regression tasks. The results of our research with joint instance and feature selection with classical (non evolutionary) methods showed, that not simultaneous, but sequential feature and then instance selection was the best option [32], so it is not so obvious at this moment how the problem should be optimally approached with evolutionary optimization. Our preliminary research with regression tasks showed that in this case it is a good idea to apply not only instance selection but also instance weighting, so in case of evolutionary optimization, the chromosome should contain not binary, but real or at least discrete multi-valued positions. And finally the last topic is a thorough study of setting all the parameters of evolutionary instance selection together and their dependence on each other and on the properties of the depicted task, as the length of the chromosome and the possible sparsity of the data in case of instance (or instance and feature) selection for big datasets in classification tasks.

REFERENCES

[1] Garcia, S., Derrac, J., Cano, J. R., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34(3), pp. 417–435 (2012)

[2] Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., Kittler, J.: A review of instance selection methods. Artificial Intelligence Review, vol. 34(2), pp. 133–143 (2010)

[3] Jankowski, N., M. Grochowski, M.: Comparison of instance selection algorithms. Lecture Notes in Computer Science, vol. 3070 pp. 598–603, ICAISC (2004)

[4] Czarnowski, I.: Cluster-based instance selection for machine classification, Knowledge and Information Systems, vol. 30, pp. 113–133 (2012)

[5] Wilson, D. R., Martinez, T. R.: Reduction techniques for instance-based learning algorithms. Machine Learning, vol. 38, pp. 257–286 (2000)

[6] Aha, D.W., Kibler, D., Albert, M.K.: Instance-based Learning Algorithms. Machine Learning, vol. 6, pp. 37–66 (1991)

[7] Kordos, M., Blachnik, M., Bialka, S.: Instance Selection in Logical Rule Extraction for Regression Problems. Lecture Notes in Artificial Intelligence, vol. 7895, pp. 167–175, ICAISC (2013)

[8] Kordos, M., Blachnik, M.: Instance Selection with Neural Networks for Regression Problems, Lecture Notes in Computer Science, vol. 7553, pp. 263-270, ICANN (2012)

[9] Kordos, M. Blachnik, M., Kozłowski, J., Perzyk, M., Bystrzycki, O., Gródek, M., Byrdziak, A., Motyka, Z.: A Hybrid System with Regression Trees in Steelmaking Process, Lecture Notes in Artificial Intelligence, vol. 6678, HAIS, pp. 222–229, (2011)

[10] Kłos-Witkowska, A.: The phenomenon of fluorescence in immunosensors. Acta Biochimica Polonica Vol. 63, pp. 215-221 (2016)

[11] Arnaiz-González, A., Díez-Pastor, J. F., Rodríguez, J., García-Osorio, C.: Instance selection of linear complexity for big data, Knowledge-Based Systems, vol. 107. pp. 83–95m (2016)

[12] Kordos, M., Rusiecki, A., Blachnik, M.: Noise Reduction in Regression Tasks with Distance, Instance, Attribute and Density Weighting. 2nd IEEE International Conference on Cybernetics, Gdynia, Poland, pp. 73–78 (2015)

[13] Kordos, M.: Instance Selection Optimization for Neural Network Training. Lecture Notes in Artificial Intelligence, vol. 9692, pp. 610–620, ICAISC (2016)

[14] Rusiecki, A., Kordos, M., Kaminski, T., Gren, K.: Training Neural Networks on Noisy Data, Lecture Notes in Artificial Intelligence, vol. 8467, pp. 131–142, ICAISC (2014)

[15] Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley (1989)

[16] Lobo, F. G., Lima, C. F., Michalewicz, Z.: Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol. 54, Springer (2007)

[17] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer (1992)

[18] Rudzinski, F.: An Application of Generalized Strength Pareto Evolutionary Algorithm for Finding a Set of Non-Dominated Solutions with High-Spread and Well-Balanced Distribution in the Logistics Facility Location Problem, LNAI, vol. 10245, pp. 439-450, ICAISC (2017)

[19] Konak, A., Coit, D., Smith, A.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering and System Safety 91, pp. 992—1007 (2006)

[20] Zavoianu, Z. C, et. al.: Performance comparison of generational and steady-state asynchronous multi-objective evolutionary algorithms for computationally-intensive problems. Knowledge-Based Systems, vol. 87, pp. 47-–60 (2015)

[21] Cano, J. R., Herrera, F., Lozano, M.: Instance Selection Using Evolutionary Algorithms: An Experimental Study. Advanced Information and Knowledge Processing, pp. 127–152 (2004)

[22] Eshelman, E. J.: The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, The First Workshop on Foundations of Genetic Algorithms, pp. 265–283, Morgan Kaufmann, (1991)

[23] Cano, J. R., Herrera, F., Lozano, M.: Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study, IEEE Transactions on Evolutionary Computations, Vol. 7, No. 6 (2003)

[24] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons (2001)

[25] Antonelli, M., Ducange, P., Marcelloni, F.: Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach. IEEE Transactions on Fuzzy Systems, vol. 20(2), pp. 276–290 (2012)

[26] Derrac, J. and Cornelis, C., Garcia, S., Herrera, F.: Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection. Information Sciences, vol. 186, pp. 73–92 (2012)

[27] Tsaia, C. F., and Eberleb, W., Chu, Chi-Yuan: Genetic algorithms in feature and instance selection. Knowledge-Based Systems, vol. 39, pp. 240–247 (2013)

[28] Kordos, M.: Optimization of Evolutionary Instance Selection. Lecture Notes in Artificial Intelligence, vol. 10245, pp. 359–369, ICAISC (2017)

[29] Łapa, K., Cpałka, K., Hayashi, Y.: Hybrid initialization in the process of evolutionary learning, Lectures Notes in Artificial Intelligence, vol. 10245, pp. 380–393, ICAISC (2017)

[30] Alcala-Fdez, J., et. al: KEEL Data-Mining Software Tool and Data Set Repository at http://sci2s.ugr.es/keel/datasets.php (2011)

[31] Riedmiller M., Braun, H.I: Rprop - A Fast Adaptive Learning Algorithm. Proceedings of the International Symposium on Computer and Information Science VII (1992)

[32] Kordos, M.: Data Selection for Neural Networks. Schedae Informaticae Vol. 25, pp. 153-–164 (2017)