Comparison of clustering optimization for classification with PSO algorithms

Klemen Berkovič
University of Maribor, Faculty
of Electrical Engineering and
Computer Science
Koroška cesta 46
Maribor, Slovenia
klemen.berkovic
@student.um.si

Uroš Mlakar
University of Maribor, Faculty
of Electrical Engineering and
Computer Science
Koroška cesta 46
Maribor, Slovenia
uros.mlakar@um.si

Borko Bošković
University of Maribor, Faculty
of Electrical Engineering and
Computer Science
Koroška cesta 46
Maribor, Slovenia
borko.boskovic@um.si

Iztok Fister
University of Maribor, Faculty
of Electrical Engineering and
Computer Science
Koroška cesta 46
Maribor, Slovenia
iztok.fister@um.si

Janez Brest
University of Maribor, Faculty
of Electrical Engineering and
Computer Science
Koroška cesta 46
Maribor, Slovenia
janez.brest@um.si

ABSTRACT

In this paper, we compare Particle Swarm Optimization (PSO) algorithms for classification based on clustering. Clustering is presented within the proposed PSO algorithms as an optimization problem, and captured in a so called fitness function. Clusters are presented with centers that should represent the classes hidden in data. With the help of PSO algorithms and proper fitness function, we optimize centers of the clusters. Because clustering belongs to a field of unsupervised learning methods, we redefine the clustering problem to supervised learning with a new fitness function that helps PSO algorithms in finding clusters centers that can classify instances of data to the right class. Two experiments are performed in our study. In the former, various fitness functions for classification based on clustering are tested. The latter measures the performance of PSO algorithms using the best fitness function from the first experiment. For testing the PSO algorithms, we use three different datasets. Friedman non-parametric statistical test, and Nemenyi and Wilcoxon post-hoc tests are conducted to properly compare the PSO algorithms.

Categories and Subject Descriptors

I.5.3 [Pattern recognition]: Clustering; G.1.6 [Numerical analysis]: Optimization

Keywords

clustering optimization, classification, particle swarm optimization, statistical comparison

1. INTRODUCTION

In many fields, including physics, bioinformatics, engineering and economics, we can encounter problems, where from a plethora of solutions we are interested in finding the most suitable. These so called optimization problems (OPs) can be sub-categorized into: discrete, continuous, and/or mixed-variable problems [18]. The difference between three mentioned groups of OPs are the type of variables in their solution space. Discrete problems work with discrete variables that are elements of $\mathbb{N}^+$ set, continuous problems have variables with elements of $\mathbb{R}$ set, while the mixed-variable problems can capture variables of both aforementioned sets.

Since we are usually limited by various computational resources (e.g., computational power and space, problem's constraints), the complexity of OPs are increased and therefore we are forced to be satisfied with a "good enough" solutions for the real-world application. Definition of a "good enough" depends exclusively on the type of a problem. Majority of the OPs have a huge, yet finite solution spaces, which can be solved approximately using meta-heuristics. In our work we are focused on meta-heuristics algorithms that are inspired by nature and are a part of Swarm Intelligence (SI) algorithms [4]. From the field of SI algorithms, we use be using PSO algorithms.

In this work, we are focused on minimization of single-objective OPs that can be defined mathematically, as follows:

$$f(\mathbf{x}^*) \leq \min_{\mathbf{x} \in \Omega} f(\mathbf{x}), \quad (1)$$

where $f: \mathbb{R}^D \mapsto \mathbb{R}$ represents the fitness function of the problem to be solved, $\mathbf{x}$ is D dimensional vector consisting of problem variables and $\mathbf{x}^*$ is a global minimum or the best solution of the problem. The fitness function f is realized as a mathematical formula or a set of empirical data that refer to the problem of interest. In order to make the problem simpler, we demand that the solutions should be in some

feasible space, written mathematically as $\mathbf{x} \in \Omega$. In our case, the feasible space Ω is determined by the upper and lower bounds of their corresponding problem variables.

Classification is a process of labeling data based on some attributes to the right type of information or to the right class. Clustering is a process of searching for centers of groups so called clusters that have very similar data instances. With clustering, we can find cluster centers that can be used for classification. The K-means algorithm is one of the most well known algorithms for clustering [9]. Unfortunately, it has two major defects: (1) the number of clusters must be known in advance, and (2) bad initial centers of clusters can degrade the performance. Because of those defects, many meta-heuristic algorithms have been used for clustering, like Big Bang–Big Crunch Algorithm (BBBCA) [8], Black Hole (BH) [7], Gravitational Search Algorithm (GSA) [6], and many others.

The rest of the paper is organized as follows: In Section 2 PSO algorithms used in our work are presented. Section 3 defines the clustering problem and presents some fitness function suitable for clustering with PSO algorithms. Section 4 shows the results of two experiments: The former based on usage of various fitness functions, while the latter compares different algorithms for classification based on clustering. Finally, in Section 5, we conclude our paper with summarizing the performed work and provide some directions for future work.

2. PARTICLE SWARM OPTIMIZATION

PSO is an optimization algorithm for solving problems which solutions variables are elements of $\mathbb{R}$. The basic PSO algorithm was presented by Kennedy and Eberhart in 1995 [10]. Individual in PSO algorithms is presented as a particle that has current position, personal best position and velocity. In each generation the PSO algorithm goes over the whole population and for each particle first updates its velocity based on the following expression:

$$\mathbf{v}_i^{(t+1)} = \omega^{(t+1)} \times \mathbf{v}_i^{(t)} + c_1 \times \mathbf{r}_1 \otimes (\mathbf{x}_{pb,i}^{(t)} - \mathbf{x}_i^{(t)}) + c_2 \times \mathbf{r}_2 \otimes (\mathbf{x}_{gb}^{(t)} - \mathbf{x}_i^{(t)}), \quad (2)$$

where $\otimes$ represents element wise multiplication, $\mathbf{r}_1$ and $\mathbf{r}_2$ have uniform random distributed values with D components $\in [0, 1]$, $\omega^{(t+1)}$ is an inertia weight, $\mathbf{v}_i^{(t)}$ is a vector representing i -th particle velocity at time t and $\mathbf{v}_i^{(t+1)}$ represents new velocity for particle i . Factors c_1 and c_2 represent social and cognitive learning rates respectively. After the velocity updating is done, the updating of particle position takes place as follows:

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}. \quad (3)$$

After particle updates its position, the personal best position, and the global best positions are update based on fitness value.

Opposition-base learning and velocity clamping are two fundamental phases of Opposition-Based Particle Swarm Optimization with Velocity Clamping algorithm (OVCP SO) defined in [16]. In the OVCP SO algorithm, opposition-based learning phase [19] is used for calculating the opposite par-

ticle $\mathbf{x}'_i$ as follows:

$$\mathbf{x}'_i = \mathbf{x}_{max} + \mathbf{x}_{min} - \mathbf{x}_i. \quad (4)$$

Opposition-based learning phase in OVCP SO consists of calculating opposite particles swarm $\mathbf{X}'$ then creating an union between original and opposite swarm of particle as $\mathbf{X}'' = \mathbf{X} \cup \mathbf{X}'$. And the last step of opposition-based learning phase is to select N best particles based on the fitness values of particles in $\mathbf{X}''$ that will present our new swarm of particles. The OVCP SO algorithm has velocity clamping operator that repairs the velocity of particles if the velocity exceeds some maximum velocity.

One of the the variations of PSO algorithm that we used in our work is Mutated Centered Unified Particle Swarm Optimization (MCUP SO) algorithm described in [20]. The MCUP SO algorithm uses two additional operators that are centering and mutation. Centering operator was described in the CPSO algorithm [12] and mutation operator was defined in Mutated Particle Swarm Optimization (MPSO) algorithm [17]. Both operators are executed at the end of the algorithm generation. The MCUP SO algorithm has a new equation for velocity updating defined as:

$$\begin{aligned} \mathbf{v}_i^{(t+1)} &= \omega^{(t+1)} \times \mathbf{v}_i^{(t)} \\ &+ c_1 \times \mathbf{r}_1 \otimes (\mathbf{x}_{pb,i}^{(t)} - \mathbf{x}_i^{(t)}) \otimes \mathbf{r}_3 \\ &+ c_2 \times \mathbf{r}_2 \otimes (\mathbf{x}_{gb}^{(t)} - \mathbf{x}_i^{(t)}) \otimes (1 - \mathbf{r}_3), \end{aligned} \quad (5)$$

where $\mathbf{r}_3$ is a D component vector with uniform random numbers $\in [0, 1]$.

Because of the bad performance of the PSO algorithm on multi-modal OPs, authors in [11] developed a PSO algorithm called Comprehensive Learning Particle Swarm Optimizer (CLPSO) algorithm which overcomes this problem. The CLPSO algorithm uses the basic PSO algorithm with additional comprehensive learning phase, where comprehensive learning phase uses all personal best positions for updating particles velocity. Particle velocity is update based on:

$$\mathbf{v}_i^{(t+1)} = \omega^{(t+1)} \times \mathbf{v}_i^{(t)} + c \times \mathbf{r} \otimes (\mathbf{z} - \mathbf{x}_i^{(t)}), \quad (6)$$

where c is user defined value, $\mathbf{r}$ is randomly uniform distributed vector with number of components equal to number of components in the solution and $\mathbf{z}$ is vector that is composed from different personal best positions.

In our implementation, we added solution repairing operator that is executed after the position of particle is updated. Solution repairing is done based on mirroring back the components of the solution that are out of the allowed search space back into the search space. Mirroring of bad components is based on modulo operation.

3. CLUSTERING OPTIMIZATION

Clustering is one of the most important data analysis techniques that involves analysis of multivariate data. It is a method of unsupervised machine learning for pattern recognition, data mining, supervised machine learning, image analysis, bioinformatics, prediction, etc. Result of clustering optimization is centers of clusters, where instances of one cluster have maximum similarity between each other and

minimum similarity from instances of the other groups [13]. In Figure 1, a simple clustering problem is presented, where we can simply find the centers of the clusters. In Figure 2,

Figure 1: Example of simple clustering dataset

the harder clustering problem is shown, where the centers for clusters representing Class 0 and Class 2 are not so trivial to find. Clustering problem defined in Figure 2 is quite

Figure 2: Example of hard clustering dataset

hard to solve with K-means algorithm because mean values of attributes a_1 and a_2 are very close for both classes. Even instances of class 0 and class 2 overlap for both attributes. Therefore we use optimization to solve the problem of clustering for all three classes represented in Figure 2. For clustering optimization we will need a fitness function that will guide our PSO algorithms toward optimal solutions in the search space. In this study, we propose the following fitness functions for clustering:

- basic clustering function,
- K-means clusters evaluation function,
- K-means clusters evaluation function with penalty function 1, and
- K-means clusters evaluation function with penalty function 2.

In the remainder of this section, the proposed fitness functions for clustering optimization are described in more detail.

If we have a dataset as presented in Figure 1, we can use the basic clustering function, expressed as:

$$f(\mathbf{O}, \mathbf{Z}) = \sum_{j=0}^{K-1} \sum_{i=0}^{N-1} w_{i,j} \times \|\mathbf{o}_i - \mathbf{z}_j\|_2, \quad (7)$$

where $\mathbf{O}$ is a dataset without data labels, $\mathbf{Z}$ represents centers of clusters, $\mathbf{W}$ is a matrix with weights for instances for each cluster, N is a number of instances in data set $\mathbf{O}$, K represents a number of clusters in $\mathbf{Z}$ and $\|\mathbf{o}_i - \mathbf{z}_j\|_2$ is the Euclidean distance between instance $\mathbf{o}_i$ and cluster center $\mathbf{z}_j$. In paper [7] authors added weights $\mathbf{W}$ which are in basic clustering function set to one. Through weights, we can provide additional information to the PSO algorithms. One example of weights usage can be with setting the weight $w_{i,j}$ to one if the instance i belong to cluster j else the weight is set to zero. Second example of weights usage can be for fuzzy clustering where weight $w_{i,j}$ has the value $\in [0, 1]$.

The second fitness function for clustering optimization is defined, as follows:

$$f(\mathbf{z}, \mathbf{O}) = p(\mathbf{z}) + \sum_{i=0}^{N-1} \min_{j=0}^{K-1} (w_{i,j} \times \|\mathbf{o}_i - \mathbf{z}_j\|_2), \quad (8)$$

where p denotes a penalty function [1]. Optimization function from Eq. (8) is used for evaluating clusters in the K-means algorithm [9], where penalty function is not used.

The third fitness function for clustering optimization takes into account intra-cluster distances, in other words:

$$p(\mathbf{z}) = \sum_{\forall e \in \mathcal{J}} \sum_{j=0}^{A-1} \min \left(\frac{r_j}{K}, \max \left(0, \frac{r_j}{K} - |z_{e0,j} - z_{e1,j}| \right) \right), \quad (9)$$

where r_j is calculated with $|u_j - l_j|$ and presents the intra-cluster distance between j attributes upper u_j and lower l_j limit, A is the number of attributes in one instance and $\mathcal{J}$ is a set containing vectors with two components, that represents indexes of clusters. Set $\mathcal{J}$ is generated with the help of Alg. 1 which gives pairs of cluster indexes that need to be checked for intra-cluster distances. The penalty function 1

Algorithm 1: Creating set $\mathcal{J}$ for calculating penalty of individual $\mathbf{Z}$ based on cluster distances

Input: K
Output: $\mathcal{J}$

```

1  $\mathcal{J} \leftarrow \{\};$ 
2 for  $i \leftarrow 0$  to  $K - 1$  do
3   for  $j \leftarrow 1$  to  $K - i - 1$  do
4     if  $i \neq K - j - 1$  then
5        $\mathcal{J} \leftarrow \{i, K - j - 1\};$ 
6     end
7   end
8 end
9 return  $\mathcal{J};$ 

```

adds penalty based on intra-clusters distances, which means if the intra-clusters distance is too small then based on that distance the solution gets its penalty. The maximum penalty

that a solution can get is equal to:

$$\sum_{i=0}^{A-1} \frac{u_j - l_j}{K}, \quad (10)$$

where the search space is divided into K partitions and every cluster should be in partition occupied only by one cluster.

In our paper, we introduce the penalty function 2 for clustering, defined as:

$$p(\mathbf{Z}, \mathbf{O}, \mathbf{c}) = (1 - a(\mathbf{Z}, \mathbf{O}, \mathbf{c})) \times \|\mathbf{u} - \mathbf{l}\|_2 \times 2, \quad (11)$$

where $\mathbf{u}$ represents upper limits of the search space, $\mathbf{l}$ represents lower limits of the search space, $\mathbf{c}$ represents a vector of class or class labels of an instances in dataset $\mathbf{O}$, and a is a function for calculating the accuracy of a given centers $\mathbf{Z}$. In Eq. (11) we added multiplication factor of two, because we want to give this penalty a higher weight. Eq. (11) redefines the problem from unsupervised to supervised learning method, because we added the information of how the instance is classified to the learning method.

4. EXPERIMENTS AND RESULTS

In our work, we performed two separate experiments. The first experiment, compares the fitness functions for classification based on clustering. The second experiment, presents statistical comparison of used PSO algorithms for classification based on clustering with the best fitness function from our first experiment. Python code of implemented algorithms can be accessed at GitHub¹ and Python code of experiments can be accessed at GitHub².

In our experiments we had to adjust the representation of the individual so that all operators of PSO algorithms could work properly. Figure 3 shows an example of an individual in a population, where $\mathbf{z}_1, \dots, \mathbf{z}_k$ represents clusters. Centers have K components which represent center points for each attribute of the dataset. Figure 3 shows attributes centers in form of symbols $z_{1,1}, \dots, z_{1,n}$ for cluster one and $z_{k,1}, \dots, z_{k,n}$ for cluster k . For all PSO algorithms used in

Figure 3: Example of solution coding

our work, we used 1,000 solution evaluations as an stopping criteria.

In our experiments, we used the Friedman non-parametric statistical test [5], where the hypothesis H_0 asserts that there is no significant statistical differences between algorithms. If the Friedman test rejected H_0 , then we used the Nemenyi post-hoc statistical test [14] for calculating critical

¹<https://github.com/kb2623/NiaPy>

²https://github.com/kb2623/NiaPy-examples/tree/master/clustering_datasets

distances determining, that the results of two algorithms are significantly different, when their critical distance intervals do not overlap. To be really confident that the difference between the results is significant, we ran a Wilcoxon 2-paired signed-rank test [22]. We present the result of Wilcoxon test in tables, where character "+" denotes that the results of two algorithms are statistically significantly different, while the character "~" shows that the difference between the results is not statistically significant. For better accuracy of statistical testing, we used 51 runs. To get a better accuracy of Friedman tests, we used more than five classifiers in our second experiment as a rule described in [2].

For comparing the results of our two experiments we used error rate calculated as:

$$e(\mathbf{Z}, \mathbf{O}, \mathbf{c}) = 1 - \frac{a(\mathbf{Z}, \mathbf{O}, \mathbf{c})}{N}, \quad (12)$$

where $\mathbf{O}$ is a dataset, $\mathbf{Z}$ represents centers of clusters, N represents number of instances in a dataset $\mathbf{O}$, $\mathbf{c}$ hold the labels that represent classes for each instance in a dataset $\mathbf{O}$ and a is the function that returns the number of correctly classified instances of the dataset $\mathbf{O}$ based on centers $\mathbf{Z}$. Before the experiment runs we divided the dataset into training and validation sets, where the validation set contained 30% of samples of the original dataset. The error rates that we compared were taken from the validation set after the optimization process finished. Because of using the error rate for statistical analysis with a combination of the Friedman test ranks, we can say that the best algorithm has the smallest mean rank.

4.1 Fitness function comparison

The first experiment that we conducted was for testing performance of the fitness functions used for classification. In this test, we used four different fitness functions described in Section 3 with the standard PSO algorithm [10] and the K-means algorithm. The K-means algorithm was initialized with random centers of clusters. For the experiment we generated a random dataset with four different classes and 500 samples that had nine components or so called attributes. For dataset generation we used scikit-learn's [15] function `make_blobs`. In the generated dataset each class had 125 instances. Figure 4 is showing significant differences between

Figure 4: Samples grouped by class with mean values for each attribute in generated dataset

intervals of data values for each class. In some cases mean values are quite far away from each other. So based on this information, the generated dataset should not be a hard problem for clustering.

Table 1: Time of execution for one function evaluation based on selected optimization function

Label	Clustering function	fitness	Mean execution time	Time (worst)	complexity
C	Eq. (7)		11.8 μ s $\pm$ 73.3 ns		$O(NKA)$
CM	Eq. (8) with penalty set to 0		12 μ s $\pm$ 56.3 ns		$O(KA(N+1))$
CMP	Eq. (8) with penalty based on Eq. (9)		46.3 μ s $\pm$ 270 ns		$O\left(KA\left(N + \frac{K-1}{2}\right)\right)$
CC	Eq. (8) with penalty based on Eq. (11)		76.9 ms $\pm$ 510 μ s		$O(NA(2K+1))$

Figure 5: The Friedman test mean ranks with the Nemenyi post-hoc test with $q_\alpha = 0.05$ for optimization functions and K-means algorithm

Table 1 shows a time complexity for each optimization function used in our first experiment. The experiment was ran on computer with Intel's i5-4570 processor on one thread with 16 GB of main memory. It can be seen from the results, that the last fitness functions labeled CC is the best function for classification based on clustering. This fitness function has the highest time complexity and consequently the longest execution time, but gives the best results compared to other fitness functions used as seen on Figure 5. Friedman test gave us the statistic value of 151.82682 and p-value of $8.26469e-32$, so for q_α of 0.05 we can say that K-means and PSO algorithm with different fitness functions work significantly different on generated dataset. In Figure 5, we can observe that the Nemenyi post-hoc test detected some statistical insignificant differences between two groups of used methods. First group with C, CM and CMP and second group with K-means and CC. From Table 2, we can observe that second group has statistically significant difference, but for the first group only methods C and CM do not have statistically significant differences.

4.2 Comparison of PSO algorithms

In our second experiment we measured the performance of six different PSO algorithms and the basic K-means algorithm labeled as KM. For the fitness function of PSO algorithms we used Eq. (8) with penalty function described in

Table 2: The Wilcoxon test showing statistical differences for $q_\alpha = 0.05$ for all clustering methods used

	KMeans	C	CM	CMP	CC
KMeans	∞	+	+	+	+
C	/	∞	$\sim$	+	+
CM	/	/	∞	+	+
CMP	/	/	/	∞	+
CC	/	/	/	/	∞

Eq. (11), since it showed the best performance in our first experiment.

We measured the performance of algorithms used on three different datasets, which are:

- **Iris:** Dataset has 150 instances with four attributes where instances are labeled with three different classes. All attributes in the dataset are elements of $\mathbb{R}^+$. Each class in the dataset has 50 instances.
- **Breast Cancer Wisconsin (BCW):** Dataset has 569 instances distributed into two classes and each instance has 30 attributes. All attributes in the dataset are elements of $\mathbb{R}^+$ numbers set. The first class contains 212 instances, while the second class contains 357 instances.
- **Wine:** One of the hardest dataset used in our second experiment is the Wine dataset. The dataset has 178 instances with 13 attributes. All attributes except one are elements of $\mathbb{R}^+$ set. Only attribute Proline is an element of $\mathbb{N}^+$ set. The dataset has three classes. The dataset contains 59 instances which belong to the first class, 71 instances of the second and 48 instances of the third class.

All datasets were obtained from the Machine learning repository [3]. The performance was measured based on the error rates from 51 runs for each dataset.

Table 3: PSO algorithms parameters

	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
NP	25	25	25	25	25	25
c_1	2.0	2.0	2.0	2.0	2.0	2.0
c_2	2.0	2.0	2.0	2.0	2.0	2.0
ω	/	0.7	/	0.7	0.7	0.7
$\mathbf{v}_{min}$	/	-1.5	/	/	/	/
$\mathbf{v}_{max}$	/	1.5	/	/	/	/
m	/	10	/	/	/	/
ω_0	/	0.9	/	/	/	/
ω_1	/	0.4	/	/	/	/
c	/	1.49445	/	/	/	/
p_0	/	/	0.3	/	/	/
w_{min}	/	/	0.4	/	/	/
w_{max}	/	/	0.9	/	/	/
δ	/	/	0.1	/	/	/
μ	/	/	/	10	/	10

Table 3 shows all the parameter values used by PSO algorithms. Some parameters are used for all PSO algorithms, while some are algorithm specific. If a parameter is not used for a specific algorithm the symbol “/” is used.

From Table 4, we can see that the best classifiers are K-means and original PSO algorithm, because they got the best minimum error rates. The CLPSO algorithm has the best mean value and the MPSO algorithm has the smallest standard deviation. One major setback of the OVCP SO algorithm was its worst found centers for clusters, that’s classification accuracy was only 29%. Friedman test gave the

Table 4: Basic statistics of used algorithms for 51 runs on Iris dataset

	KM	PSO	CLPSO	OVCP SO	MPSO	CPSO	MCUP SO
mean	0.0588	0.0553	0.0544	0.0644	0.0505	0.0579	0.0562
std	0.0376	0.0305	0.0356	0.0954	0.0274	0.0424	0.0380
min	0.0000	0.0000	0.0222	0.0222	0.0222	0.0222	0.0222
median	0.0444	0.0444	0.0444	0.0444	0.0444	0.0444	0.0444
max	0.1777	0.1333	0.2000	0.7111	0.1555	0.2444	0.1777

statistic value of 104.81 with p-value of 2.48215e−20. For q_α value of 0.05 we can say based on the Friedman test that used algorithms work significantly different. Mean ranks of algorithms from Figure 6 fit the results in the Table 4. Figure 6 shows that the MCUP SO algorithm holds the smallest rank, despite not having the best accuracy. We can see that

Figure 6: Friedman mean ranks and Nemenyi post-hoc test with $q_\alpha = 0.05$ for PSO algorithms and the K-means algorithm on Iris dataset

Nemenyi test implies that MCUP SO is not significantly different from CPSO, MPSO, CLPSO and PSO algorithms. Based on results in Table 5 we can reject the hypothesis of Nemenyi test. The Wilcoxon test detected insignificant difference only between MCUP SO, CPSO, MPSO and PSO algorithms. From the Wilcoxon test we can observe that the MCUP SO algorithm is significantly different compared to OVCP SO, CLPSO and K-means algorithms. MCUP SO, CPSO and MPSO algorithms performed the best based on Friedman mean rank and the Wilcoxon post-hoc test.

From the Table 6, we can observe that the best accuracy obtained the CLPSO algorithm, the best mean value obtained the OVCP SO algorithm, while the PSO algorithm recorded the smallest standard deviation. The MCUP SO algorithm

Table 5: Detected significant statistical differences with the Wilcoxon test with $q_\alpha = 0.05$ on Iris dataset

	KM	PSO	CLPSO	OVCP SO	MPSO	CPSO	MCUP SO
KM	∞	+	+	+	+	+	+
PSO	/	∞	~	+	~	~	~
CLPSO	/	/	∞	~	+	~	+
OVCP SO	/	/	/	∞	+	+	+
MPSO	/	/	/	/	∞	~	~
CPSO	/	/	/	/	/	∞	~
MCUP SO	/	/	/	/	/	/	∞

obtained good results, it had the best median value and the best accuracy in the worst run in all of the 51 runs. Friedman test gave the statistics value of 174.63646 with p-value of 4.66982e−35. For q_α value of 0.05 we can say that the Friedman test rejected the H_0 hypothesis. We can see from the Figure 7, that the MCUP SO algorithm obtained the smallest rank, because of the average standard deviation and smallest error rate on the worst run of the algorithm. On the Figure 7, we can see that MCUP SO,

Table 6: Basic statistics of used algorithms for 51 runs on BCW dataset

	KM	PSO	CLPSO	OVCP SO	MPSO	CPSO	MCUP SO
mean	0.1606	0.1552	0.1643	0.1479	0.1598	0.1672	0.1536
std	0.0768	0.0686	0.0779	0.0697	0.0728	0.0892	0.0723
min	0.0818	0.0818	0.0760	0.0818	0.0818	0.0877	0.0877
median	0.1345	0.1345	0.1403	0.1345	0.1345	0.1345	0.1169
max	0.3567	0.3684	0.3567	0.3567	0.3567	0.3684	0.3333

Figure 7: Friedman mean ranks and the Nemenyi post-hoc test with $q_\alpha = 0.05$ for PSO algorithms and the K-means algorithm on BCW dataset

MPSO, CPSO and PSO algorithms do not show significant differences based on the Nemenyi test, but the Wilcoxon test found insignificant differences only between MCUP SO, CPSO and MPSO. If we check the PSO algorithm compared to MCUP SO, MPSO and CPSO algorithms, then we can observe that the Wilcoxon test detected significant differences only between PSO and MCUP SO algorithms. MPSO, CPSO, MCUP SO algorithms based on Friedman mean rank and the Wilcoxon post-hoc test work the best for this dataset.

Table 7: Detected significant statistical differences with the Wilcoxon test with $q_\alpha = 0.05$ on BCW dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
KM	∞	+	+	+	+	+	+
PSO	/	∞	+	+	$\sim$	$\sim$	+
CLPSO	/	/	∞	+	+	+	+
OVCPSO	/	/	/	∞	+	+	+
MPSO	/	/	/	/	∞	$\sim$	$\sim$
CPSO	/	/	/	/	/	∞	$\sim$
MCUPSO	/	/	/	/	/	/	∞

In Table 8, we can see that two algorithms, namely MPSO and CPSO algorithms, obtained the best cluster centers for classification on the Wine dataset. We can see that the MPSO algorithm got the best and the worst clusters centers, and the best median and standard deviation values on this dataset. Table 8 is showing that not only the MPSO algorithm but CPSO and CLPSO algorithms got the best clusters centers for classification. Best median values were obtained with MPSO, OVCPSO and CLPSO algorithms. The Friedman test gave us the statistic value of 58.06224

Table 8: Basic statistics of used algorithms for 51 runs on Wine dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
mean	0.3151	0.3111	0.2977	0.2970	0.2930	0.3122	0.3162
std	0.0490	0.0515	0.0519	0.0495	0.0457	0.0594	0.0510
min	0.2222	0.2037	0.1851	0.2037	0.1851	0.1851	0.2407
median	0.3148	0.3148	0.2962	0.2962	0.2962	0.3148	0.3148
max	0.4814	0.4259	0.4259	0.4074	0.3703	0.4444	0.4629

with p-value of $1.1131e-10$. For q_α of 0.05 we can say that the Friedman test detected significant differences between used algorithms. From the results seen in Table 8 we would suspect that the MPSO algorithm would have the smallest mean rank, but the results from Figure 8 show that the CPSO algorithm has the smallest mean rank. As we can see from Figure 8 there is no significant differences between PSO, MPSO, CPSO and MCUPSO algorithms, which can be confirmed with results collated in Table 9. For the basic PSO algorithm the Wilcoxon test detected only two significant differences. As seen from Figure 8 the CPSO algo-

Table 9: Detected significant statistical differences with the Wilcoxon test with $q_\alpha = 0.05$ on Wine dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
KM	∞	+	$\sim$	+	+	+	+
PSO	/	∞	+	$\sim$	$\sim$	$\sim$	$\sim$
CLPSO	/	/	∞	$\sim$	+	+	+
OVCPSO	/	/	/	∞	+	+	+
MPSO	/	/	/	/	∞	$\sim$	$\sim$
CPSO	/	/	/	/	/	∞	$\sim$
MCUPSO	/	/	/	/	/	/	∞

gorithm has the smallest mean rank, but we can not say that

Figure 8: Friedman mean ranks and the Nemenyi post-hoc test with $q_\alpha = 0.05$ for PSO algorithms and the K-means algorithm on Wine dataset

this algorithm is the winner for this dataset. Because of insignificant differences between CPSO, MPSO and MCUPSO algorithms we can say that this three algorithms work best for for this dataset.

5. CONCLUSIONS

In our work, we used clustering optimization for classification. We proposed a new fitness function that has two components. The first component is a clustering function that is used in the K-means algorithm for clusters evaluation and the second component is a penalty function, which is the basis for supervised learning. Our proposed fitness function is a weighted sum of this two components. First component has weight equal to 0.25 and second component has weight equal to 0.75. As it turns out, on used datasets, this fitness function works well. In our work we tried to eliminate initial clusters defect of the K-means algorithm, which makes the K-means algorithm converge fast to some local optimum.

One of the options for feature work is to use more function evaluations with an additional archive, because as we have seen in our experiments, good solutions can be found only after 1,000 function evaluations. Because evolutionary computation is a large research area, we would look for other optimization algorithm. In our work, we did not found the solution for detecting the number of clusters in dataset. This is a challenging task for currently known algorithms and would be a good option for feature work. A multi-objective optimization algorithm would be a good starting point for detecting a number of optimal clusters hidden in data.

6. ACKNOWLEDGMENTS

Authors would like to thank the contributors of nature inspired framework for Python, called NiaPy [21], for they efforts in the development of the framework. The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0041).

7. REFERENCES

- [1] A. Chehouri, R. Younes, J. Perron, and A. Ilinca. A constraint-handling technique for genetic algorithms

- using a violation factor. *arXiv preprint arXiv:1610.00976*, 2016.
- [2] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.
 - [3] D. Dua and C. Graff. UCI machine learning repository, 2017.
 - [4] R. C. Eberhart, Y. Shi, and J. Kennedy. *Swarm intelligence*. Elsevier, 2001.
 - [5] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.
 - [6] X. Han, L. Quan, X. Xiong, M. Almeter, J. Xiang, and Y. Lan. A novel data clustering algorithm based on modified gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 61:1 – 7, 2017.
 - [7] A. Hatamlou. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*, 222:175 – 184, 2013. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.
 - [8] A. Hatamlou, S. Abdullah, and M. Hatamlou. Data clustering using big bang–big crunch algorithm. In *International Conference on Innovative Computing Technology*, pages 383–388. Springer, 2011.
 - [9] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).
 - [10] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942 – 1948 vol.4, 12 1995.
 - [11] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295, June 2006.
 - [12] Y. Liu, Z. Qin, Z. Shi, and J. Lu. Center particle swarm optimization. *Neurocomputing*, 70(4-6):672–679, 2007.
 - [13] S. J. Nanda and G. Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation*, 16:1 – 18, 2014.
 - [14] P. Nemenyi. Distribution-free multiple comparisons (doctoral dissertation, princeton university, 1963). *Dissertation Abstracts International*, 25(2):1233, 1963.
 - [15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
 - [16] F. Shahzad, A. R. Baig, S. Masood, M. Kamran, and N. Naveed. Opposition-based particle swarm optimization with velocity clamping (ovcpso). In *Advances in Computational Intelligence*, pages 339–348. Springer, 2009.
 - [17] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73, May 1998.
 - [18] K. Socha. *Ant Colony Optimization for Continuous and Mixed-Variable Domains*. PhD dissertation, Université Libre de Bruxelles, 2014.
 - [19] H. R. Tizhoosh. Opposition-based learning: a new scheme for machine intelligence. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06)*, volume 1, pages 695–701. IEEE, 2005.
 - [20] H.-C. Tsai. Unified particle swarm delivers high efficiency to particle swarm optimization. *Applied Soft Computing*, 55:371 – 383, 2017.
 - [21] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr. NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, 3, 2018.
 - [22] F. Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.