Feature Selection Using Modified Sine Cosine Algorithm with COVID-19 Dataset

Miodrag Zivkovic, Luka Jovanovic, Milica Ivanovic, Aleksa Krdzic, Nebojsa Bacanin, and Ivana Strumberger

Abstract The research proposed in this paper shows application of the sine cosine swarm intelligence algorithm for feature selection problem in the machine learning domain. Feature selection is a process that is responsible for selecting datasets' features that have the biggest effect on the performances and the accuracy of the system. The feature selection task performs the search for the optimal set of features through a enormous search space, and since the swarm intelligence metaheuristics have already proven their performances and established themselves as good optimizers, their application can drastically enhance the feature selection process. This paper introduces the improved version of the sine cosine algorithm that was utilized to address the feature selection problem. The proposed algorithm was tested on ten standard UCL repository datasets and compared to other modern algorithms that have been validated on the same test instances. Finally, the proposed algorithm was tested against the COVID-19 dataset. The obtained results indicate that the method proposed in this manuscript outperforms other state-of-the-art metaheuristics in terms of features number and classification accuracy.

Keywords Sine cosine algorithm · Swarm intelligence · Feature selection · Classification · COVID-19

M. Zivkovic (✉) · L. Jovanovic · M. Ivanovic · A. Krdzic · N. Bacanin · I. Strumberger
Singidunum University, Danijelova 32, 11000 Belgrade, Serbia
e-mail: mzivkovic@singidunum.ac.rs

L. Jovanovic
e-mail: luka.jovanovic.191@singimail.rs

M. Ivanovic
e-mail: milica.ivanovic.17@singimail.rs

A. Krdzic
e-mail: aleksa.krdzic.19@singimail.rs

N. Bacanin
e-mail: nbacanin@singidunum.ac.rs

I. Strumberger
e-mail: istrumberger@singidunum.ac.rs

V. Suma et al. (eds.), *Evolutionary Computing and Mobile Sustainable Networks*,
Lecture Notes on Data Engineering and Communications Technologies 116,
https://doi.org/10.1007/978-981-16-9605-3_2

1 Introduction

Rapid developments in information science have resulted in a dramatic increase in dataset dimensions over the past decade. Potential dimension reduction algorithms are needed to remove redundant or irrelevant information from these datasets, since these features can lead to reduced performance of learning algorithms [22].

Typically considered a mechanism for preprocessing, feature selections are used for decreasing the total number of input variables, as well as finding the most relevant subset from a complete features set. Feature selection reduces the dimensionality of data by removing the noise and irrelevant attributes. This challenge is very important, especially when the real-time classification is needed by finding optimal or near-optimal subset of features, the training process can be shortened and classification accuracy can be improved. It is applied so as to increase the precision of prediction results given by the machine learning model, by reducing complexity, and diminishing redundant and irrelevant features in the dataset. This can be crucial in case of some critical applications, such as medical diagnostic [10]. Feature subset evaluation and search strategy are the two primary stages of preprocessing. Search strategy uses techniques for subset feature selection, while feature subset evaluation utilizes a classifier for evaluating the quality for the selected feature subset. All methods for feature selection, according to reviewed literature, are defined as either filter based or wrapper based.

Metaheuristic algorithms are considered the most reliable and efficient techniques for optimization and show great results when applied to problems considered more challenging or with higher-dimensional datasets. As a result, these algorithms show great promise and have been applied to many real-world problem that require optimization and performance improvements [3, 4, 25, 32, 34, 36]. Although these algorithms are often nature inspired, this is not necessarily always the case as shown in the sine cosine algorithm (SCA) [20].

Because of the high accuracy results achieved, as well as the reduced computational times when compared to traditional discrete methods, the metaheuristic approach has been employed by researchers, in wrapper-based methods, when solving the problem of feature selection. A Gaussian mutational chaotic fruit fly optimization algorithm's [31] application has been suggested for tackling the problem of feature selection, specifically to classification tasks. An augmented model of the dragonfly algorithm (DA), the hyper-learning binary dragonfly algorithm (HLBDA), has been implemented for feature evaluation and utilized on coronavirus (COVID-19) datasets [28].

SCA is a population-based algorithm, named after its use of the sine and cosine functions in its formulation, originally intended for use in solving optimization problems [20]. The algorithm initially creates a collection of multiple randomized solutions requiring them to fluctuate toward the best solution during the exploitation phase or outwards to encourage exploration employing a mathematical model formed from the sine cosine functions.

Some deficiencies were observed in the original SCA while performing practical empirical simulations with standard unconstrained benchmarks. Because of this, we have attempted to improve the basic SCA by performing hybridization with the well-known ABC algorithm. The mSCA is benchmarked using ten datasets form the University of California, Irvine (UCI) repository, and Arizona State University, as well as a single dataset of the coronavirus disease (COVID-19).

The main contribution of this conducted research can be outlined in the following:

- Proposal of a mSCA applied to the problem of feature selection elements of the ABC algorithm is integrated into the SCA to improve exploratory behavior.
- Testing the mSCA on ten standard benchmark datasets with low medium and high dimensions sets represented.
- Comparing the mSCA to other advanced feature selection algorithms and demonstrating the improvements made.
- Applying the proposed mSCA to solving a case study of COVID-19.

The remainder of this article is organized according to the following order: Sect. 2 shows a summary of the reviewed literature. Section 3 consists of a description of the original SCA. Section 4 shows experimental results and discusses the findings based on said results. Section 5 summarizes the findings and presents proposals for the direction of further work in this field.

2 Literature Review

When we have large datasets that are too difficult to classify, the use of swarm intelligence-based algorithm is suggested. Each large dataset contains features that are insignificant and irrelevant which can prove to be difficult when trying to analyze and interpret data. Swarm intelligence algorithm's purpose is to reduce dimensionality (feature selection) by keeping only useful features and those containing rich information. As a result of using dimensionality reduction technique, we have better understanding and interpretation of data, as well as higher accuracy of the results. There are two main steps in dimensionality reduction process, extracting features and selecting features. Before any further explanation of those features, we should give a short overview of swarm intelligence algorithms.

Swarm intelligence algorithms are part of the artificial intelligence (AI) field, and they are so-called nature-inspired metaheuristics [29]. Many groups of animals form collective intelligence which means that every member acts independently, but they mutually exchange information. That information eventually takes the group toward the optimal solution of their problem. Such animal colonies are ants, birds, hawks, fish, and more [16, 21, 29]. Nature-inspired metaheuristics are not that efficient at finding the most optimal solutions inside the search area, but they are efficient at finding the candidate solutions. Furthermore, they are especially good at finding possible solutions inside very large search areas. Because they take unreasonable amount of time to find the most optimal solution, swarm intelligence algorithms

are also classified as NP-hard problems [15]. Many diverse problems can be solved with swarm intelligence algorithms such as wireless sensor network optimization [4, 32], cloud computing [6, 8, 35] and optimization of neural networks [2, 5, 12, 24], machine learning, and COVID-19 prediction [33], all the way to solving complicated problems in the field of medicine [7].

In order to prepare raw and unprocessed data, feature extraction is used [17]. A new dataset is formed by keeping some of the core features after which new features can be derived. Eventually, we have a new dataset that is cleaner, containing only features relevant to the specific problem and with fewer dimensions compared to the original dataset.

Since we have our most relevant and important data after feature extraction, the next step is feature selection. With feature selection, we select attributes previously defined in original dataset. This step is extremely important since the combination of the right attributes can improve the model's performance and accuracy. A common example of feature selection, alongside feature extraction, is image processing and analysis. Large amount of statistical features can be retrieved from the image, but a combination of only a few gives satisfactory results.

A side effect of feature selection is a possible loss of a certain amount of information, but, due to achieving simplicity of the model and significant performance improvement, it is well worth it. There are three distinct categories of techniques for selecting features, the wrapper, filter, and embedded technique [9].

Filter techniques choose the features that should contain the most information, without taking into consideration whether there are any relationships between the features or not. Wrapper techniques choose features that are most accurate to our machine learning model by going through all feature combinations. As for the embedded technique, the features are chosen while the model is still being constructed [9]. With these techniques, a decent performance can be achieved on relatively small datasets, but, for larger datasets, because of the decline in performance, a different method should be used such as swarm intelligence algorithm. In a reasonable amount of computational time, satisfactory results on large datasets are provided by the algorithm.

3 Original and Proposed Modified Sine Cosine Algorithm

SCA originally designed with the purpose of solving optimization problems, and first introduced by Seyadali Mirjalili [20], is a generally new population-based algorithm. The algorithm stochastically looks for the most optimum solution to our problems. At the very beginning, it starts with a randomized set of solutions, then repeatedly evaluates this set against an objective function, and follows a given ruleset that forms the core of the given optimization technique. As such, finding the most optimal solution in the first iteration is not guaranteed; however, given enough iterations and a large enough collection of randomized solutions, the probability of the global optimal solution being found increases.

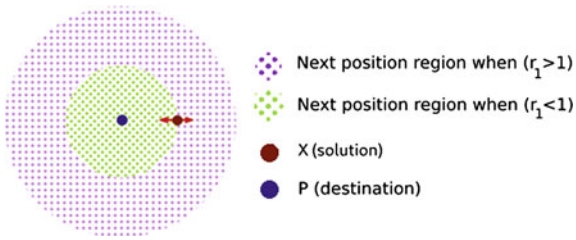

Fig. 1 Sine and cosine effects on the upcoming position from Eq. 1

The process of optimization in the stochastic population-based approach, regardless of the algorithm being applied, can be split across two distinct phases: exploration phase and exploitation phase. In the exploration phase, the algorithm quickly, in a very random manner, combines solutions from a given random set, looking through the search space for the most favorable regions. With the exploitation phase, the changes are gradually made, however, noticeably less severe than those from the exploitation phase.

The original SCA proposes the use of the following equations for position updating in both phases Eq (1):

$$\begin{aligned} X_i^{t+1} &= X_i^t + r_1 \times \sin(r_2) \times |r_3 P_i^t - X_i^t| \\ X_i^{t+1} &= X_i^t + r_1 \times \cos(r_2) \times |r_3 P_i^t - X_{i_i}^t| \end{aligned} \tag{1}$$

where X represents the current solution's position in the i-th dimension after the t-th iteration, P_i represents the point of destination in the i-th dimension, r_1, r_2 and r_3 are random numbers, and || indicates an absolute value.

In Eq. (2), a combination of these two Eq. (1) can be seen:

$$X_i^{t+1} = \begin{cases} X_i^{t+1} = X_i^t + r_1 \times \sin(r_2) \times |r_3 P_i^t - X_i^t|, & r_4 < 0.5 \\ X_i^{t+1} = X_i^t + r_1 \times \cos(r_2) \times |r_3 P_i^t - X_{i_i}^t|, & r_4 \geq 0.5 \end{cases} \tag{2}$$

where r_4 represents a random value in [0,1].

The four major parameters of the SCA are r_1, r_2, r_3, and r_4, as shown in the equations above. Parameter r_1 defines region of the following position. Said position signifies one of the two possible spaces: the space between the solution and destination or the space outside of the two. Parameter r_2 dictates the movement away from or toward the destination, or more precisely, how distant the movement is. The role of parameter r_3 is to stochastically diminish ($r_3 < 1$) or emphasize ($r_3 > 1$) the distribution effects on distance definition. Lastly, parameter r_4 plays the part of switching between the sine and cosine components in Eq. (2).

The effects of the sine and cosine functions on Eqs. (1) and (2) are depicted in Fig. 1. The search space in between the two solutions is dictated by these two equations as depicted in said figure. These two equations can also be expanded to include higher dimensions; however, Fig. 1 depicts a two-dimensional model.

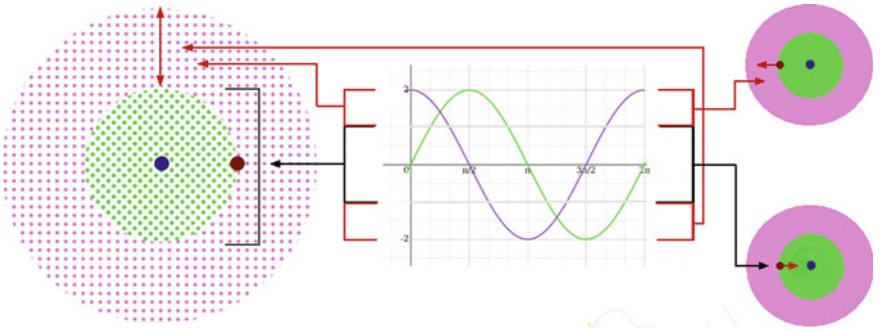

Fig. 2 Sine and cosine with the range in $[-2, 2]$ allow a solution to go around (inside the space between them) or beyond (outside the space between them) the destination

The sine and cosine functions cyclic pattern allows for solution repositioning around a different solution. This can provide a guarantee of exploitation in the defined space enclosed by the two calculated solutions. Altering the range of sine and cosine function enables the solutions to search outside the space that is defined by the corresponding destinations, and this is done so as to ensure exploration.

While changing the function range, as shown in Fig. 2, it is necessary to update the new position of the solution taking into account positions of the existing solutions. The updated position is attained by choosing a random value in range $[0, 2\pi]$ for r_2 from Eq. 2, and it can be either on the outside or on the inside. This mechanism ensures both exploitation and exploration of the search space.

The algorithm needs to have the ability to balance both exploration and exploitation when searching for promising regions inside a given search space. This is done to eventually converge on a global optimum. The SCA does this by changing range of the sine and cosine adaptively in Eq. 2 according to Eq. 3:

$$r_1 = a - t\frac{a}{T} \tag{3}$$

where a represents a constant, T represents the maximum amount of allowed repetitions, and finally t represents the active iteration.

Through many repetitions of Eq. 2, we get a decreasing range of sine and cosine as shown in Fig. 3.

By taking into consideration both Figs. 2 and 3, it can be deduced that the SCA focuses on exploitation when the given ranges are in $[-1, 1]$, and on exploration when the ranges are in between $(1, 2]$ and $[-2, -1)$.

Finally, the pseudocode for the SCA are shown in Fig. 4. As depicted, the algorithm begins the process of optimization with randomized set of solutions. Every time the algorithm encounters a solution, it considers the most optimal so far, and it assigns it as a target point. The algorithm then, in regard to the most optimal solution, updates other solutions. During this process, the iteration counter is increased, and

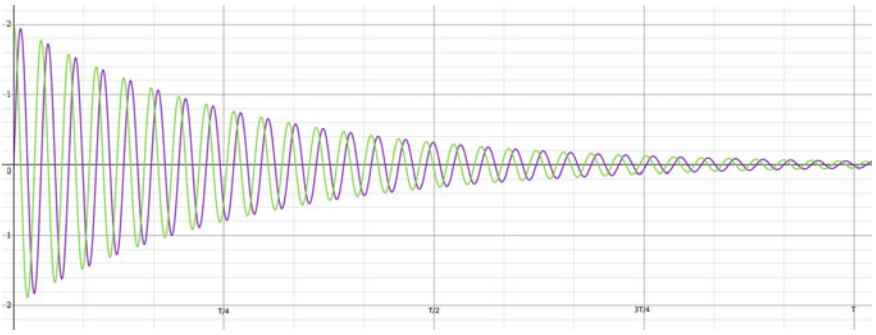

Fig. 3 Decreasing range of sine and cosine ($a = 3$)

initialize a randomized of search agents (X)
do
  evaluate each of the search agents by the objective function
  update the best solution obtained so far $(P = X^*)$
  update r_1, r_2, r_3, r_4
  update the positions of search agents
while $t < maximumnumberofiterations$
return the best solution obtained so far as the global optimum

Fig. 4 General steps of the original SCA algorithm

the ranges of sine and cosine function are, after every iteration, updated emphasizing exploitation of the defined search space. When the counter reaches the maximum allowed amount of iterations, the optimization process of the original SCA stops. Other conditions for termination can be implemented as well, including the total number of functional evaluations or reaching a desired global optimum accuracy.

3.1 Proposed Modified SCA Approach

Notwithstanding the fact that the basic SCA metaheuristics establish excellent results for standard benchmark instances [20], based on additional conducted experiments with basic congress on evolutionary computation (CEC) benchmark suites, it was concluded that the basic SCA can be further improved.

As many other swarm intelligence approaches, original SCA may be stuck in non-optimal regions of the search domain in early iterations of execution. In this early phase, due to the lack of exploration power, if the search process is not "lucky" and if does not register optimal domain of the search space, algorithm may stuck in sub-optimal domain for many iterations. As a consequence, worse mean values are generated, and performance of the metaheuristic is seriously degraded.

Without adding complexity to the algorithm, abovementioned drawback of original SCA can overcome by introducing simple mechanism in the search process as follows: after every iteration, 5% of worst solutions from the population are replaced with the randomly generated individuals within the boundaries of the search space in the first 50% of iterations:

$$X_{\text{rnd}}^{j} = L^{j} + \phi \cdot (U^{j} - L^{j}), \tag{4}$$

where X_{rnd}^{j} is j-th component of the newly generated random solution, phi is the value derived from the uniform distribution, and U^{J} and L^{j} are upper and lower boundaries of j-th parameter, respectively.

Based on conducted simulations, it was concluded that in approximately first 50% of iterations described exploration mechanism should be triggered. However, in later iterations, this mechanism is not needed, and it would only represent an obstacle in performing a fine-tuned search around the promising domain of the search region. Proposed method is named modified SCA (mSCA), and its pseudocode is shown in Algorithm 1.

Algorithm 1 Pseudocode of proposed mSCA

Initialization. Generation of the starting random population of N individuals X within the boundaries of the search space and calculation of its fitness.
Initialize the maximal number of iterations T.
do
  for all X in the generated population **do**
    Evaluate utilizing the fitness function.
    if $f(X)$ better than $f(P)$ **then**
      Update the position of the best solution so far ($P = X^{*}$).
    end if
  end for
  Update r_1 parameter
  Update r_2, r_3, and r_4 parameters.
  Update the positions of search agents
  if $t < T \cdot 0.5$ **then**
    Replace 5% worst solution with random one using Eq. (4)
  end if
while ($t < T$)
return P the best solution found.

4 Experiments and Discussion

In the research presented in this manuscript, the proposed mSCA algorithm was tested on ten basic datasets and one additional COVID-19 dataset. The experimental simulations in this research were executed through 20 independent runs, while each run consisted of 70 iterations. The size of the population was set to 8, and a mixed initializer was utilized to randomly select 2/3 from the available amount of features. The suggested improved optimization method's performance has been tested on ten UCI datasets that are very popular among researchers and used as a benchmark in Table 1.

Table 1 List of experimental simulation datasets

No.	Name	Features	Samples
1	Glass	10	214
2	Hepatitis	19	155
3	Lymphography	18	148
4	Primary tumor	17	339
5	Soybean	35	307
6	Horse colic	27	368
7	Ionosphere	34	351
8	Zoo	16	101
9	Musk 1	166	476
10	Arrhythmia	279	452
11	COVID-19	15	TBD

The performance of mSCA was evaluated on a computer with a central processing unit (CPU) with a clock frequency of 2.90 GHz, additionally with 16.0G of available random access memory (RAM) and programmed in the language of Python with Anaconda framework using machine learning libraries including NumPy, SciPy and scikit-learn. The performance is judged based on five calculated evaluation metrics. The evaluation metrics include optimal fitness value, average fitness value, fitness value normal divination, precision of classification, and the ratio of feature selection with each method executed and evaluated 20 times. The repetition is performed to better represent results and avoid bias caused by optimization algorithms stochastic nature. The result averages are logged and presented after the last iteration of the 20 individual runs.

The mSCA in tested against ten standard datasets and COVID-19 dataset. And its performance is then evaluated. The datasets are acquired from the UCI repository [11] and Arizona State University [18]. Table 2 represents best overall fitness while Table 3 represents the mean fitness metric. Tables 4 and 5 each represent standard deviation, average classification accuracy and feature selection of already referenced ten datasets. The best results are marked in bold in each table, except in the case of tie, where none of the results are marked. Tests of the proposed mSCA have been conducted on different structures, so as to provide evidence of the algorithms efficiency and performance in differing dimension.

The obtained results from Tables 2, 3, 4 and 5 from conducted experiments proved the efficiency and efficacy of mSCA proposed algorithm. Based on the empirical analysis, a deduction can be made that the proposed mSCA can yield higher-quality results than the algorithms it has been tested against. The eight algorithms tested in this paper are (BDA) [19], binary artificial bee colony (BABC) [14], binary multiverse optimizer (BMVO) [1], binary particle swarm optimization (BPSO) [30], chaotic crow search algorithm (CCSA) [23], binary coyote optimization algorithm (BCOA) [27], evolution strategy with covariance matrix adaptation (CMAES) [13]

Table 2 Best fitness metric over ten UCI datasets for the compared approaches

No.	Dataset	HLBDA	BDA	BABC	BMVO	BPSO	CCSA	BCOA	CMAES	LSHADE	SCA	mSCA
1	Glass	0.0067	0.0067	0.0067	0.0067	0.0067	0.0067	0.0067	0.0067	0.0067	0.0067	0.0067
2	Hepatitis	0.1154	0.1245	0.1305	0.1226	0.1235	0.1309	0.1220	0.1229	0.1235	0.1223	**0.1151**
3	Lymphography	0.1116	0.1181	0.1122	0.1297	0.1177	0.1309	0.1257	0.1171	0.1226	0.1288	**0.1109**
4	Primary tumor	0.5646	0.5731	0.5673	0.5887	0.5623	0.5756	0.5642	0.5623	0.5883	0.5685	0.5623
5	Soybean	**0.2009**	0.2073	0.2035	0.2421	0.2189	0.2294	0.2035	0.2010	0.2037	0.2112	0.2010
6	Horse colic	0.1298	0.1330	0.1350	0.1440	0.1309	0.1418	0.1304	0.1298	0.1327	0.1429	**0.1297**
7	Ionosphere	**0.0694**	0.0731	0.0830	0.0980	0.0814	0.0906	0.0715	0.0746	0.0719	0.0719	0.0697
8	Zoo	0.0332	0.0325	0.0332	0.0332	0.0325	0.0338	0.0325	0.0334	0.0325	0.0337	0.0325
9	Musk 1	0.0608	0.0626	0.0879	0.0942	0.0783	0.0836	0.0662	0.0739	0.0633	0.0792	**0.0606**
10	Arrhythmia	0.2926	0.3179	0.3330	0.3352	0.3282	0.3399	0.3106	0.3269	0.2999	0.3215	**0.2923**

Table 3 Statistical mean fitness metric over ten datasets for the compared approaches

No.	Dataset	HLBDA	BDA	BABC	BMVO	BPSO	CCSA	BCOA	CMAES	LSHADE	SCA	mSCA
1	Glass	0.0112	0.0112	0.0111	0.0116	0.0111	0.0116	0.0111	0.0119	0.0116	0.0116	0.0112
2	Hepatitis	0.1311	0.1368	0.1386	0.1454	0.1334	0.1457	0.1425	0.1430	0.1399	0.1382	**0.1310**
3	Lymphography	0.1311	0.1359	0.1350	0.1527	0.1342	0.1515	0.1416	0.1392	0.1474	0.1305	**0.1309**
4	Primary tumor	0.5850	0.5932	0.5845	0.6088	0.5850	0.5996	0.5944	0.5935	0.5989	0.5949	0.5852
5	Soybean	0.2124	0.2212	0.2254	0.2594	0.2245	0.2479	0.2168	0.2177	0.2211	0.2238	**0.2121**
6	Horse colic	0.1358	0.1427	0.1480	0.1674	0.1409	0.1699	0.1434	0.1419	0.1479	0.1483	**0.1355**
7	Ionosphere	0.0842	0.0928	0.1016	0.1106	0.0958	0.1114	0.0868	0.0882	0.0909	0.0913	0.0842
8	Zoo	0.0401	0.0409	0.0397	0.0482	0.0369	0.0493	0.0437	0.0470	0.0503	0.0425	**0.0398**
9	Musk 1	0.0673	0.0832	0.0961	0.1082	0.0929	0.1017	0.0792	0.0842	0.0793	0.0903	**0.0669**
10	Arrhythmia	0.3160	0.3341	0.3450	0.3538	0.3413	0.3529	0.3290	0.3352	0.3270	0.3275	0.3160

Table 4 Standard deviation results for ten datasets included in the comparative analysis

No.	Dataset	HLBDA	BDA	BABC	BMVO	BPSO	CCSA	BCOA	CMAES	LSHADE	SCA	mSCA
1	Glass	0.0033	0.0033	0.0033	0.0033	0.0033	0.0032	0.0033	0.0036	0.0033	0.0034	0.0033
2	Hepatitis	0.0093	0.0062	**0.0057**	0.0099	0.0080	0.0065	0.0139	0.0132	0.0091	0.0069	0.0062
3	Lymphography	0.0130	0.0121	0.0125	0.0104	0.0138	0.0117	0.0134	0.0151	0.0147	0.0121	**0.0111**
4	Primary tumor	0.0104	0.0099	0.0085	0.0080	0.0116	0.0120	0.0135	0.0136	0.0134	0.0134	**0.0078**
5	Soybean	0.0087	0.0094	0.0089	0.0118	**0.0041**	0.0108	0.0103	0.0116	0.0109	0.0104	0.0088
6	Horse colic	**0.0039**	0.0088	0.0073	0.0164	0.0069	0.0140	0.0110	0.0097	0.0158	0.0087	0.0041
7	Ionosphere	0.0086	0.0105	0.0101	0.0053	0.0056	0.0090	0.0104	0.0070	0.0099	0.0098	**0.0052**
8	Zoo	0.0079	0.0080	0.0073	0.0101	0.0065	0.0097	0.0092	0.0101	0.0116	0.0085	0.0073
9	Musk 1	**0.0062**	0.0099	0.0063	0.0076	0.0079	0.0079	0.0077	0.0075	0.0097	0.0081	0.0063
10	Arrhythmia	0.0094	0.0088	0.0042	0.0077	0.0073	0.0065	0.0100	0.0057	0.0130	0.0118	**0.0039**

Table 5 Percentage of selected feature for ten datasets included in comparative analysis

No.	Dataset	HLBDA	BDA	BABC	BMVO	BPSO	CCSA	BCOA	CMAES	LSHADE	SCA	mSCA
1	Glass	0.2900	0.2900	0.3050	0.3250	0.3050	0.3300	0.3050	0.2900	0.3050	0.3050	0.2900
2	Hepatitis	0.3184	0.3658	0.3158	0.3605	0.3263	0.3526	0.3053	0.3500	0.3368	0.3368	**0.3043**
3	Lymphography	0.4500	0.4806	0.5083	0.4889	0.5000	0.4972	0.4472	0.5083	**0.4333**	0.4806	0.4500
4	Primary tumor	0.6676	0.6118	0.6706	0.5941	0.6676	0.6441	0.6059	0.6265	0.6412	0.6402	**0.5923**
5	Soybean	0.6529	0.6229	0.6429	0.5743	0.6414	0.5971	0.6371	0.6286	0.6486	0.6362	**0.5723**
6	Horse colic	**0.0870**	0.1370	0.2426	0.2574	0.1963	0.2926	0.1056	0.1130	0.1500	0.0945	0.0892
7	Ionosphere	0.2191	0.2676	0.2897	0.2882	0.2441	0.3426	0.2265	0.2456	0.2824	0.2741	**0.2183**
8	Zoo	0.4563	0.4469	0.4969	0.5188	0.4250	0.4938	0.4531	**0.4500**	0.4938	0.5091	0.4523
9	Musk 1	0.4687	0.4783	0.4946	0.4602	0.4964	0.5033	0.4458	0.4946	0.4849	0.5012	**0.4451**
10	Arrhythmia	0.4050	0.4699	0.4803	0.4303	0.4706	0.4787	0.4048	0.4627	0.4495	0.4152	0.4048

and success history-based adaptive differential evolution with linear population size reduction (LSHADE) [26] algorithms.

Based on the presented results, it can be concluded that the proposed mSCA metaheuristics clearly outperformed the original SCA approach for all observed metrics. In general, when compared to other approaches included in the simulations, mSCA obtained the best performances. Based on the results from Table 2, the proposed mSCA approach obtained the best results for best fitness metrics on five out of the ten UCI datasets. When the statistical mean fitness metric is observed, from Table 3, it can be concluded that the mSCA obtained the best results on six out of ten UCI datasets. In case of the standard deviation, Table 4 shows that the mSCA obtained the best results on four datasets and tied the best results on the Glass dataset. In Table 5, comparative analysis between proposed mSCA and other approaches in terms of selected features (expressed as ratios of total number of features in the datasets) is presented. From results, it can be seen that proposed mSCA in average utilizes a smaller number of features than other methods which means that it managed to substantially reduce the problem dimensions, which makes the training process of a classifier much faster (Figs. 5 and 6).

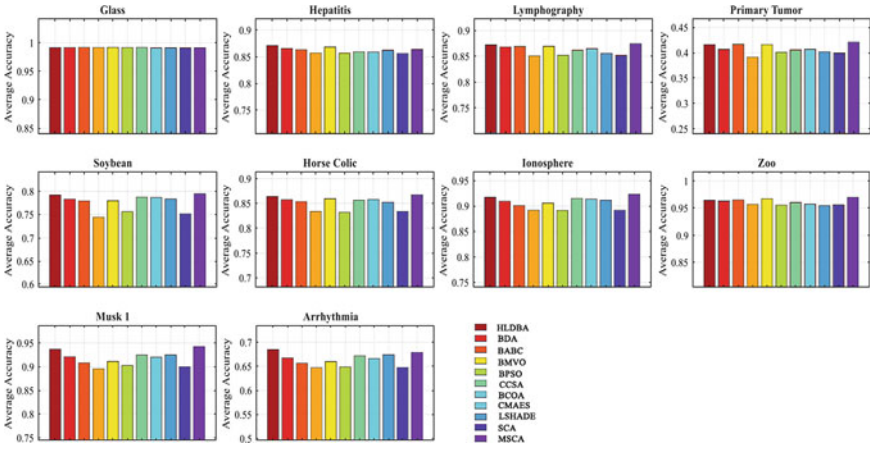

Fig. 5 Average classification accuracy over ten datasets included in the comparative analysis

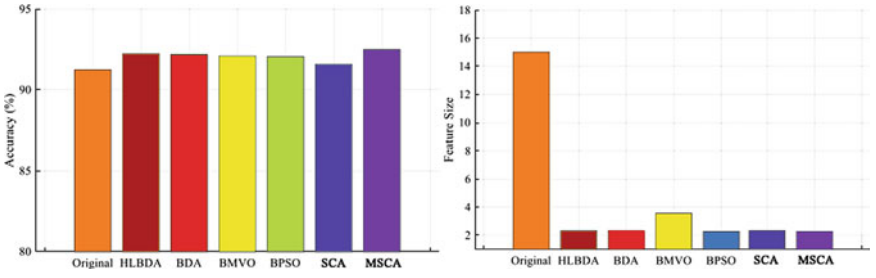

Fig. 6 Accuracy and feature size of the proposed SCA and mSCA on the COVID-19 dataset

5 Conclusion

The conducted research that is presented in this manuscript proposes a novel feature selection method. The implemented mSCA metaheuristics address the drawbacks of the original SCA method that are observed from the results of the conducted experiments. The proposed mSCA approach was later used to help find the crucial features for the classification process. The presented algorithmic method of optimization was validated on ten benchmark datasets, and the results are represented in comparison with other swarm intelligence metaheuristics. Finally, the mSCA method was used on COVID-19 dataset. The conducted experiments results indicate that the mSCA approach outperformed other methods included in the comparative analysis. Based on defined research contributions, the novelty of proposed research can be summed as follows: more efficient SCA metaheuristics are devised, solving feature selection challenge was improved in terms of classification accuracy, and the number of employed features and classification for the most recent and important COVID-19 dataset was performed.

The future research in this area will be focused on including additional datasets to the experimental simulations. Also, the future work will deal with adaptation of other swarm intelligence metaheuristics, with a goal to further enhance the classification accuracy.

References

1. Al-Madi, N., Faris, H., Mirjalili, S.: Binary multi-verse optimization algorithm for global optimization and discrete problems. Int. J. Mach. Learn. Cybern. **10**(12), 3445–3465 (2019). https://doi.org/10.1007/s13042-019-00931-8
2. Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., Tuba, M.: Monarch butterfly optimization based convolutional neural network design. Mathematics **8**(6), 936 (2020)
3. Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., Tuba, M., Zivkovic, M.: Task scheduling in cloud computing environment by grey wolf optimizer. In: 2019 27th Telecommunications Forum (TELFOR). pp. 1–4. IEEE (2019)
4. Bacanin, N., Tuba, E., Zivkovic, M., Strumberger, I., Tuba, M.: Whale optimization algorithm with exploratory move for wireless sensor networks localization. In: International Conference on Hybrid Intelligent Systems. pp. 328–338. Springer (2019)
5. Bezdan, T., Tuba, E., Strumberger, I., Bacanin, N., Tuba, M.: Automatically designing convolutional neural network architecture with artificial flora algorithm. In: Tuba, M., Akashe, S., Joshi, A. (eds.) ICT Systems and Sustainability, pp. 371–378. Springer Singapore, Singapore (2020)
6. Bezdan, T., Zivkovic, M., Antonijevic, M., Zivkovic, T., Bacanin, N.: Enhanced flower pollination algorithm for task scheduling in cloud computing environment. In: Machine Learning for Predictive Analysis, pp. 163–171. Springer (2020)
7. Bezdan, T., Zivkovic, M., Tuba, E., Strumberger, I., Bacanin, N., Tuba, M.: Glioma brain tumor grade classification from mri using convolutional neural networks designed by modified fa. In: International Conference on Intelligent and Fuzzy Systems. pp. 955–963. Springer (2020)
8. Bezdan, T., Zivkovic, M., Tuba, E., Strumberger, I., Bacanin, N., Tuba, M.: Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. In: International Conference on Intelligent and Fuzzy Systems. pp. 718–725. Springer (2020)

9. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers and Electrical Engineering **40**(1), 16–28 (2014). https://doi.org/10.1016/j.compeleceng.2013.11.024. https://www.sciencedirect.com/science/article/pii/S0045790613003066, 40th- year commemorative issue
10. Chen, J.I.Z., Hengjinda, P.: Early prediction of coronary artery disease (cad) by machine learning method-a comparative study. J. Artif. Intell. **3**(01), 17–33 (2021)
11. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
12. Gajic, L., Cvetnic, D., Zivkovic, M., Bezdan, T., Bacanin, N., Milosevic, S.: Multi-layer perceptron training using hybridized bat algorithm. In: Computational Vision and Bio-Inspired Computing, pp. 689–705. Springer (2021)
13. Hansen, N., Kern, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions, pp. 282–291. Lecture Notes in Computer Science. Springer (2004)
14. He, Y., Xie, H., Wong, T.L., Wang, X.: A novel binary artificial bee colony algorithm for the set-union knapsack problem. Future Gener. Comput. Syst. **78**, 77–86 (2018). https://doi.org/10.1016/j.future.2017.05.044. https://www.sciencedirect.com/science/article/pii/S0167739X17310415
15. Johnson, D.S.: The np-completeness column: an ongoing guide. J. Algorithms **6**(3), 434–451 (1985). https://doi.org/10.1016/0196-6774(85)90012-4. https://www.sciencedirect.com/science/article/pii/0196677485900124
16. Karaboga, D.: Artificial bee colony algorithm. Scholarpedia **5**(3), 6915 (2010). https://doi.org/10.4249/scholarpedia.6915, revision #91003
17. Levine, M.: Feature extraction: a survey. Proc. IEEE **57**(8), 1391–1407 (1969). https://doi.org/10.1109/PROC.1969.7277
18. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J., Liu, H.: Feature selection: a data perspective. ACM Comput. Surv. (CSUR) **50**(6), 94 (2018)
19. Mirjalili, S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. **27**(4), 1053–1073 (2016). https://doi.org/10.1007/s00521-015-1920-1
20. Mirjalili, S.: Sca: a sine cosine algorithm for solving optimization problems. Knowl Based Syst **96** (2016). DOI 10.1016/j.knosys.2015.12.022
21. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007. https://www.sciencedirect.com/science/article/pii/S0965997813001853
22. Ranganathan, G.: A study to find facts behind preprocessing on deep learning algorithms. J. Innov. Image Proces. (JIIP) **3**(01), 66–74 (2021)
23. Sayed, G.I., Hassanien, A.E., Azar, A.T.: Feature selection via a novel chaotic crow search algorithm. Neural Computing and Applications **31**(1), 171–188 (2019). https://doi.org/10.1007/s00521-017-2988-6
24. Strumberger, I., Tuba, E., Bacanin, N., Zivkovic, M., Beko, M., Tuba, M.: Designing convolutional neural network architecture by the firefly algorithm. In: 2019 International Young Engineers Forum (YEF-ECE). pp. 59–65. IEEE (2019)
25. Strumberger, I., Tuba, E., Zivkovic, M., Bacanin, N., Beko, M., Tuba, M.: Dynamic search tree growth algorithm for global optimization. In: Doctoral Conference on Computing, Electrical and Industrial Systems. pp. 143–153. Springer (2019)
26. Tanabe, R., Fukunaga, A.: Improving the search performance of shade using linear population size reduction. 2014 IEEE Congress on Evolutionary Computation (CEC) pp. 1658–1665 (2014)
27. Thom de Souza, R.C., de Macedo, C.A., dos Santos Coelho, L., Pierezan, J., Mariani, V.C.: Binary coyote optimization algorithm for feature selection. Pattern Recognition **107**, 107470 (2020). https://doi.org/10.1016/j.patcog.2020.107470. https://www.sciencedirect.com/science/article/pii/S0031320320302739
28. Too, J., Mirjalili, S.: A hyper learning binary dragonfly algorithm for feature selection: A covid-19 case study. Knowledge-Based Systems **212**,(2021). https://doi.org/10.1016/j.knosys.2020.106553. https://www.sciencedirect.com/science/article/pii/S0950705120306821

29. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010), pp. 65–74. Springer (2010)
30. Yin, P.Y.: A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. Journal of Visual Communication and Image Representation **15**(2), 241–260 (2004). https://doi.org/10.1016/j.jvcir.2003.12.001. https://www.sciencedirect.com/science/article/pii/S1047320303000981
31. Zhang, X., Xu, Y., Yu, C., Heidari, A.A., Li, S., Chen, H., Li, C.: Gaussian mutational chaotic fruit fly-built optimization and feature selection. Expert Systems with Applications **141**,(2020). https://doi.org/10.1016/j.eswa.2019.112976. https://www.sciencedirect.com/science/article/pii/S0957417419306943
32. Zivkovic, M., Bacanin, N., Tuba, E., Strumberger, I., Bezdan, T., Tuba, M.: Wireless sensor networks life time optimization based on the improved firefly algorithm. In: 2020 International Wireless Communications and Mobile Computing (IWCMC). pp. 1176–1181. IEEE (2020)
33. Zivkovic, M., Bacanin, N., Venkatachalam, K., Nayyar, A., Djordjevic, A., Strumberger, I., Al-Turjman, F.: Covid-19 cases prediction by using hybrid machine learning and beetle antennae search approach. Sustain. Cities Soc. **66**, 102669 (2021)
34. Zivkovic, M., Bacanin, N., Zivkovic, T., Strumberger, I., Tuba, E., Tuba, M.: Enhanced grey wolf algorithm for energy efficient wireless sensor networks. In: 2020 Zooming Innovation in Consumer Technologies Conference (ZINC). pp. 87–92. IEEE (2020)
35. Zivkovic, M., Bezdan, T., Strumberger, I., Bacanin, N., Venkatachalam, K.: Improved harris hawks optimization algorithm for workflow scheduling challenge in cloud–edge environment. In: Computer Networks, Big Data and IoT, pp. 87–102. Springer (2021)
36. Zivkovic, M., Zivkovic, T., Venkatachalam, K., Bacanin, N.: Enhanced dragonfly algorithm adapted for wireless sensor network lifetime optimization. In: Data Intelligence and Cognitive Informatics, pp. 803–817. Springer (2021)