Feature Selection Using Salp Swarm Algorithm with Chaos*

Sobhi Ahmed, and Majdi Mafarja
Department of Computer Science
Birzeit University
Birzeit, Palestine
sahmed@birzeit.edu,mmafarja@birzeit.edu

Hossam Faris, and Ibrahim Aljarah
King Abdullah II School for Information Technology
The University of Jordan
Amman, Jordan
hossam.faris@ju.edu.jo,i.aljarah@ju.edu.jo

ABSTRACT

The performance of classification algorithms is highly sensitive to the data dimensionality. High dimensionality may cause many problems to a classifier like overfitting and high computational time. Feature selection (FS) is a key solution to both problems. It aims to reduce the number of features by removing the irrelevant, redundant and noisy data, while trying to keep an acceptable classification accuracy. FS can be formulated as an optimization problem. Metaheuristic algorithms have shown superior performance in solving this type of problems. In this work, a chaotic version of Salp Swarm Algorithm (SSA) is proposed, which is considered one of the recent metaheuristic algorithms. The proposed approach is applied for the first time on feature selection problems. Four different chaotic maps are used to control the balance between the exploration and exploitation in the proposed approach. The proposed approaches are evaluated using twelve real datasets. The comparative results shows that the chaotic maps significantly enhances the performance of the SSA algorithm and outperforms other similar approaches in the literature.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; **Feature selection**;

KEYWORDS

Feature Selection, Optimization, Classification, Salp Swarm Algorithm

ACM Reference Format:
Sobhi Ahmed, and Majdi Mafarja and Hossam Faris, and Ibrahim Aljarah. 2018. Feature Selection Using Salp Swarm Algorithm with Chaos. In *Proceedings of 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (ISMSI '2018).* ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Dimensionality reduction is one of the elementary steps that assists data mining and machine learning algorithms to be faster and more efficient [15]. Dimensionality reduction can be divided into main types: Feature Extraction (FE), and Feature Selection (FS). In FE approaches, a new subset of features is created from the original features. On the other hand, FS involves the selection of a subset of the original features by removing the redundant or irrelevant features, while trying to increase the performance of the learning algorithm [16]. FS is a more popular dimensionality reduction technique than FE approaches and more widely used in machine learning. Dimensionality reduction becomes a very crucial step specially, when the learning process is applied on large amounts of data with high dimensionality [15].

Formally speaking, FS is the process of selecting n features from N original features where $n < N$ without any decrease in the performance of the learning algorithm (i.e., classification accuracy). FS approaches can be classified into two main types; filters and wrappers [16]. Filter approaches depend on the relations between the features without considering a learning algorithm to evaluate feature subsets. They rank all features in a dataset and remove the features that have weights less than a predefined threshold. In wrapper approaches, a learning algorithm is involved in the evaluation process, and the selected features must meet a certain criterion depending on the learning algorithm itself. Considering the computational time, filters are popular methods to high dimensional data due to their low costs. Despite the fact that wrappers are more expensive than filters in terms of computational time, they are preferred when the goal is to increase the prediction power of a learning algorithm.

In FS process, searching for the best subset is a challenging problem that attracted the attention of many researchers [4, 10]. One possible approach is to generate all possible solutions and select the best one. Although this approach guarantees optimality, it requires evaluating all feature subsets, which is impractical and time-consuming, especially when dealing with a large number of features. Another approach is to use heuristic search techniques. Recently, many feature selection approaches that use metaheuristic search algorithms have been proposed [18]. Nature-inspired metaheuristics algorithms have been widely used with feature selection problem and have shown a superior performance [8, 9, 17, 21]. There are two main types according to the source of inspiration: evolutionary-based algorithms (e.g., Genetic algorithm (GA) [11]), and swarm intelligence (SI) algorithms (e.g., Particle Swarm Optimization (PSO) [6]).

GA and PSO have been widely applied for FS [3, 17, 19]. More recent nature-inspired algorithms have been applied to FS problems and achieved good results. The Whale Optimization Algorithm (WOA) is a recent SI-based algorithm that has been used as a searching algorithm [21, 22]. Another approach that is based on Ant Lion Optimization (ALO) Algorithm has been proposed in [20]. A Binary

*Produces the permission block, and copyright information

ISMSI '2018, March 2018, Phuket, Thailand

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Gravitational Search Algorithm (BGSA) feature selection approach was proposed in [9].

Salp Swarm Algorithm (SSA) is a recent SI-based algorithm that mimics the behavior of salps in the sea [23]. SSA showed a good performance when tested with many optimization problems. Competency, flexibility, and simplicity are the main characteristics of SSA. This motivated our attempts to use it as a search strategy in a wrapper FS approach. As some metaheuristic algorithm, SSA is a stochastic algorithm that has a considerable number of random components in its mathematical models. Controlling those parameters highly influences the performance of the algorithm. Chaos theory is one of the popular mathematical approaches that proved its efficiency in improving the performance of metaheuristics algorithms [27]. It has been proven in the literature that using chaotic systems to replace the random numbers in the mathematical models improves the global convergence capability of the algorithm and prevents stagnation in the local optima [1, 12]. In this paper, we investigate the impact of four chaotic maps on the performance of the recently proposed SSA algorithm for the first time in the literature.

The rest of this paper is organized as follows: Section 2 presents basics of the SSA algorithm. A brief description about the used chaotic maps and the details of the proposed approach is provided in Section 3. In Section 4, the experimental results are presented and analyzed. Finally, in Section 5, conclusions are given.

2 OVERVIEW OF SSA

The Salp Swarm Algorithm is a recent nature-inspired metaheuristic algorithm that mimics the special swarming behavior of salps in the sea. Salps usually live is groups and they often form a swarm called salp chain. The first salp is denoted as the leader, while the others are followers. In the mathematical model of the SSA, location of the leader salp should be updated by Eq. 1

$$x_j^1 = \begin{cases} F_j + c_1 \left(\left(ub_j - lb_j \right) c_2 + lb_j \right) & c_3 \geq 0.5 \\ F_j - c_1 \left(\left(ub_j - lb_j \right) c_2 + lb_j \right) & c_3 < 0.5 \end{cases} \quad (1)$$

where x_j^1 is the position of the best solution (i.e., the leader), F_j is the j^{th} dimension in the food source position. c_1 is an important variable in SSA that gradually decreases over the course of iterations (see Eq. 2), to allow high exploration at the early stages of the optimization process, then high exploitation in last steps. c_2 and c_3 are two random numbers in the range [0, 1]. They direct the next position in the j^{th} dimension towards $+\infty$ or $-\infty$ as well as dictating the step size. ub_j is the upper bound of the j^{th} dimension and lb_j is the lower bound of the j^{th} dimension.

$$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \quad (2)$$

where l represents the current iteration, L indicates the maximum number of iterations.

The followers' positions are updated using Eq. 3.

$$x_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1} \right) \quad (3)$$

where $i \geq 2$ and x_j^i represents the position of the i^{th} follower at the j^{th} dimension.

The pseudocode of the SSA algorithm is presented in Algorithm 1.

Algorithm 1 Pseudocode of the SSA algorithm

```
Generate randomly initial salp population x_i(i = 1, 2, ..., n)
for iteration from 1 to maxIteration do
    Evaluate each salp in the population
    Denote the best salp as F
    Update the value of c_1 by Eq. 2
    for (each salp (x_i)) do
        if x_i is a leader then
            Update position using Eq. 1
        else
            Update position using Eq. 3
Return F
```

2.1 Binary SSA (BSSA)

The original SSA was modeled to solve continuous optimization problems. To use this algorithm with features selection problems, which are binary by nature, the algorithm must be reformed accordingly. In Binary SSA (BSSA), the salps are allowed to move in restricted directions that are bounded by 0 and 1. The most popular and easy way to control the conversion process is to use Transfer Functions (TF) [24]. TF is employed to define a probability of updating an element in the feature subset (solution) to be 1 or 0. A sigmoid TF was produced by Kennedy and Eberhart [13] to convert the continuous version of PSO to a discrete version as in Eq. 4.

$$T(x_j^i(t)) = \frac{1}{1 + \exp^{-x_j^i(t)}} \quad (4)$$

where x_j^i is the j^{th} element in solution x in the j^{th} dimension, and t is the current iteration.

A feature in the feature subset will be selected or deselected as in Eq. 5 which uses the the produced probability from Eq. 4.

$$x_i^k(t+1) = \begin{cases} 0 & \text{If } rand < T(x_i^k(t+1)) \\ 1 & \text{If } rand \geq T(x_i^k(t+1)) \end{cases} \quad (5)$$

where $x_i^d(t+1)$ is the $i-th$ element at the d^{th} dimension in x solution.

3 THE PROPOSED APPROACH

In this section we present the proposed chaotic BSSA (CBSSA) approaches.

3.1 Chaotic Maps

Chaotic maps are nonlinear dynamic systems, representing a random-like, deterministic process, that is non-converging bounded; and sensitive to the initial parameter [5]. Chaotic maps are apparently similar to the randomness of a simple deterministic dynamical system and unpredictable. They also possess an element of regularity [5]. For a dynamic system to be considered chaotic, it must be sensitive to initial conditions, topologically mixing, dense periodic

orbits, ergodic, and stochastically intrinsic [26]. The use of chaotic sequences in SSA can be helpful to better escape from local minima compared to the classical SSA. In this approach, the value of the c_3 variable in Eq. 1 is set using chaotic maps instead of using random numbers. The selected chaotic maps are listed as follows:

$$\text{Circle: } x_{i+1} = mod(x_i + b - (\frac{a}{2\pi})\sin(2\pi x_i), 1), a = 0.5, b = 0.2, \quad (6)$$

$$\text{Logistic: } x_{i+1} = a x_i (1 - x_i), \ a = 4, \quad (7)$$

$$\text{Piecewise: } x_{i+1} = \begin{cases} \frac{x_i}{P} & 0 \le x_i < P \\ \frac{x_i - P}{0.5 - P} & P \le x_i < 0.5 \\ \frac{1 - P - x_i}{0.5 - P} & 0.5 \le x_i < 1 - P \\ \frac{1 - x_i}{P} & 1 - P \le x_i < 1 \end{cases}, \ P = 0.4, \quad (8)$$

$$\text{Tent: } x_{i+1} = \begin{cases} \frac{x_i}{0.7} & x_i < 0.7 \\ \frac{10}{3}(1 - x_i) & x_i \ge 0.7 \end{cases} \quad (9)$$

In this approach, the uniformly distributed random parameter c_3 is modified and replaced with chaotic sequences obtained from the aforementioned equations; circle map (Eq. 6), logistic map (Eq. 7), piecewise map (Eq. 8), and tent map (Eq. 9). Because the proposed approaches use the binary SSA, the range of all employed maps lies in the interval of (0,1). The main objective of using the chaotic maps is to enhance the performance of binary SSA algorithm by improving the global convergence by escaping local optima.

3.2 Feature selection using chaotic SSA

Wrapper FS considers both maximizing performance of the learning algorithm (e.g., classification accuracy) and minimizing the number of selected features. Each solution is evaluated according to the proposed fitness function, which employs the k-NN classifier [2] as an evaluator and considers the number of selected features in the solution. In order to balance between the number of selected features in each solution (minimum) and the classification accuracy (maximum), the fitness function in Eq. 10 is used in SSA algorithm to evaluate search agents.

$$Fitness = \alpha \cdot Err_{rate} + \beta \frac{|n|}{|N|} \quad (10)$$

where Err_{rate} represents the classification error rate, $|n|$ and $|N|$ are the number of selected features and the number of original features in the dataset respectively, α and β are the weights of the classification error rate and selection ratio, $\alpha \in [0, 1]$ and $\beta = (1 - \alpha)$ adopted from [22].

In CBSSA, each solution is represented as a binary vector with N elements, where N is the number of features in a dataset. The 0 value indicates that the corresponding element is not selected and the 1 value indicates that the feature is selected.

4 EXPERIMENTAL RESULTS

In this paper, twelve well-known datasets from the UCI data repository [14] were utilized to assess the efficacy of the proposed approaches as in Table 1. The results of all approaches were obtained by performing twenty consecutive runs. Five well-known FS approaches were implemented and used for comparison purposes. The algorithms were implemented in Matlab and executed on an Intel Core i5 machine, 2.2 GHz CPU and 4 GB of RAM. The maximum number of iterations is 100 and the population size is 10. The means of fitness values, classification accuracy, number of selected features and computational time for each approach were reported. In this work, the K-NN classifier (where K = 5 [22]) is used as an evaluator in the wrapper feature selection approach. Each dataset is divided in two parts; 80% for training and 20% for testing. In all experiments in this section, the parameters are adopted as follows: BGSA [25], bGWO, bALO, BPSO, GA [7]

Table 1: Benchmark datasets

Dataset name	No. of classes	No. of features	No. of samples
Abalone	11	8	3842
Glass	6	9	214
Iris	3	4	150
Letter	26	16	20000
Shuttle	7	9	58000
Spambase	2	57	4601
Tae	3	5	151
Vehicle	4	18	846
Waveform	3	21	5000
Wine	3	13	178
Wisconsin	2	9	683
Yeast	9	8	1484

Inspecting the results in Table 2, it is evident that the use of different chaotic maps instead of using the random numbers has improved the performance of the proposed algorithm. Using the Circle chaotic map enables SSA algorithm to outperform other algorithms in 50% of the datasets, followed by the Logistic chaotic map which enabled the SSA to perform better than other approaches in 42% of the datasets. Comparing with other approaches, we can see that the performance of the proposed approaches is better than other approaches (including GA and PSO) in many cases, and have the same performance in the worst cases. In terms of the average selection ratio, we can see from Table 3 that the proposed approaches have a competitive performance in many datasets when compared with other approaches. It is worth mentioning that CBSSA approaches outperformed the basic SSA algorithm in 10 datasets, which proves the efficiency of using the chaotic maps with this algorithm.

The overall performance of the proposed approaches can be better judged when studying the fitness values that combine both classification accuracy and the selection ratio. From the results in Table 4, it can be seen that CBSSA approaches obtained better results than other approaches in ten datasets out of twelve.

From the previous results, it is clear that the performance of the algorithm was enhanced by using the chaotic maps to replace the random numbers in SSA. This is because the algorithm becomes able to explore a considerable portion of the feature space, which gives more chances to select the most informative features. In CBSSA, the chaotic maps were used to generate the values of the main parameter that controls the balance between exploration and exploitation in the algorithm. The main reason of the better performance of

Table 2: Average Classification Accuracy from 20 Runs for All Approaches

Dataset	BSSA	CBSSA1	CBSSA2	CBSSA3	CBSSA4	bGWO	BGSA	BPSO	bALO	GA
Abalone	0.247	**0.265**	0.245	0.263	0.248	0.244	0.235	0.237	0.260	0.256
Glass	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Iris	0.967	**1.000**	**1.000**	**1.000**	0.967	0.998	0.933	0.967	0.967	**1.000**
Letter	0.961	0.965	0.967	**0.970**	0.964	0.958	0.926	0.962	0.962	0.965
Shuttle	**0.999**	0.998	**0.999**	**0.999**	0.998	0.998	**0.999**	**0.999**	0.998	**0.999**
Spambase	0.929	0.936	0.932	0.932	0.924	0.912	0.908	0.932	0.940	**0.947**
Tae	0.613	0.645	0.645	0.613	0.581	0.677	0.684	**0.774**	0.677	0.576
Vehicle	0.731	**0.757**	0.739	0.741	0.732	0.712	0.675	0.716	0.738	0.713
Waveform	0.799	**0.810**	0.802	0.804	0.809	0.777	0.768	0.798	0.806	0.806
Wine	0.461	**0.540**	0.498	0.267	0.408	0.253	0.167	0.233	0.315	0.318
Wisconsin	**0.985**	0.973	**0.985**	0.983	**0.985**	0.971	0.977	**0.985**	0.964	0.983
Yeast	0.532	0.535	**0.562**	0.519	0.495	0.516	0.488	0.556	0.519	0.524

Table 3: Average Number of Selected Features from 20 Runs for All Approaches

Dataset	BSSA	CBSSA1	CBSSA2	CBSSA3	CBSSA4	bGWO	BGSA	BPSO	bALO	GA
Abalone	5	4	**2.2**	6	5.19	5.35	4.45	4.4	5	4.7
Glass	2.1	2.45	2.25	2	3.32	4.25	3.5	**1**	2.65	3.05
Iris	3	2	**1**	**1**	3.00	3.9	2.15	2	2	**1**
Letter	13.55	**11**	11.75	12.45	12.10	14.4	11.05	11.15	11.95	**11**
Shuttle	3	2.7	3.05	3	2.62	4	4.1	**2**	2.65	3.05
Spambase	41.4	44.15	44.65	40.7	43.73	40.45	29.5	30.95	46.7	**28.9**
Tae	**2**	3	3	**2**	**2.00**	3.7	3.35	**2**	**2**	2.15
Vehicle	10.55	12.7	10.45	9.95	11.54	11.25	8.4	**7.3**	12.55	7.95
Waveform	15.95	17.2	17.5	16.65	17.29	17.7	**13.3**	13.65	16.95	14.05
Wine	6.35	6.45	6.2	4.65	5.36	7.15	6.75	**1.65**	6.3	3.55
Wisconsin	5	4	4.2	4.65	**3.49**	4.55	4.75	4.6	4.9	4.6
Yeast	7	7	7	7	5.98	7.25	**5.65**	7	6	6.55

Table 4: Average Fitness Values from 20 Runs for All Approaches

Dataset	BSSA	CBSSA1	CBSSA2	CBSSA3	CBSSA4	bGWO	BGSA	BPSO	bALO	GA
Abalone	0.752	**0.732**	0.751	0.737	0.751	0.755	0.763	0.761	0.739	0.743
Glass	0.002	0.002	0.002	0.002	0.003	0.004	0.004	**0.001**	0.003	0.003
Iris	0.041	0.005	**0.003**	**0.003**	0.041	0.011	0.071	0.038	0.038	**0.003**
Letter	0.047	0.042	0.040	**0.038**	0.044	0.050	0.080	0.045	0.045	0.042
Shuttle	0.005	0.005	0.004	0.004	0.004	0.006	0.006	**0.003**	0.005	0.004
Spambase	0.077	0.071	0.075	0.074	0.083	0.095	0.096	0.072	0.068	**0.058**
Tae	0.387	0.357	0.357	0.387	0.419	0.327	0.320	**0.228**	0.323	0.424
Vehicle	0.273	**0.248**	0.264	0.262	0.272	0.291	0.326	0.285	0.267	0.289
Waveform	0.207	**0.196**	0.204	0.202	0.198	0.229	0.236	0.206	0.201	0.198
Wine	0.539	**0.460**	0.502	0.729	0.589	0.745	0.830	0.760	0.683	0.678
Wisconsin	0.020	0.031	0.019	0.022	**0.018**	0.034	0.028	0.020	0.041	0.022
Yeast	0.472	0.469	**0.442**	0.485	0.508	0.488	0.514	0.449	0.484	0.479

the BCSSA approaches was that the chaotic maps radically change over the course of iterations. This assists the BCSSA to switch between the updating mechanisms more frequently and consequently showing higher exploration. This did not degrade the exploitation since there was no change in the adaptive mechanisms of BCSSA. In fact, the proposed algorithm stochastically switches between exploration and exploitation, while the maximum changes in the solutions is decreased using the adaptive mechanism.

5 CONCLUSIONS

In this paper, a chaotic version of the SSA algorithm was proposed and applied to feature selection problems. Four chaotic maps were used to replace the random operator that controls the balance between exploration and exploitation in the algorithm. Twelve UCI benchmark datasets were used to assess the performances of the proposed approaches. The results are compared with three of the most recent wrapper feature-selection methods including chaotic

Table 5: Average Computational Time from 20 Runs for All Approaches

Dataset	BSSA	CBSSA1	CBSSA2	CBSSA3	CBSSA4	bGWO	BGSA	BPSO	bALO	GA
Abalone	25.0	22.1	22.3	22.9	25.0	24.9	20.9	**20.6**	24.2	23.9
Glass	7.3	7.7	7.6	7.7	7.8	7.7	7.2	**6.9**	7.2	8.3
Iris	6.9	6.9	7.1	6.8	7.1	7.0	**6.2**	6.6	6.9	7.6
Letter	1682.9	1428.8	1180.8	1114.0	1719.7	1934.5	**430.6**	825.5	1744.3	1568.9
Shuttle	612.3	685.1	661.0	643.4	801.9	798.9	529.0	**474.4**	662.3	540.5
Spambase	195.1	202.5	198.5	197.7	237.8	256.0	**166.3**	167.0	257.6	188.5
Tae	6.7	7.0	7.0	7.1	7.0	7.3	6.7	**6.5**	7.0	8.3
Vehicle	10.4	11.3	10.8	11.0	11.3	11.4	9.8	**9.4**	10.4	11.3
Waveform	118.1	120.7	120.1	111.9	134.9	146.8	**91.2**	107.4	139.4	131.5
Wine	**9.0**	9.9	9.9	10.1	9.7	9.8	9.3	9.2	9.4	10.5
Wisconsin	8.9	9.6	9.4	9.4	9.4	9.6	8.9	**8.7**	9.3	10.1
Yeast	14.4	15.0	14.2	14.5	15.4	16.2	**12.6**	13.8	15.4	15.3

BGWO, BGSA, and BALO, and two classical FS methods including PSO, and GA. SSA has shown a competitive performance, and the results proved the ability of chaotic maps in improving the quality of the SSA algorithm.

REFERENCES

[1] Bilal Alatas, Erhan Akin, and A Bedri Ozer. 2009. Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons & Fractals* 40, 4 (2009), 1715–1734.

[2] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.

[3] Li-Yeh Chuang, Cheng-Hong Yang, and Jung-Chike Li. 2011. Chaotic maps based on binary particle swarm optimization for feature selection. *Applied Soft Computing* 11, 1 (2011), 239–248.

[4] Manoranjan Dash and Huan Liu. 1997. Feature selection for classification. *Intelligent data analysis* 1, 1-4 (1997), 131–156.

[5] Leandro dos Santos Coelho and Viviana Cocco Mariani. 2008. Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. *Expert Systems with Applications* 34, 3 (2008), 1905–1913.

[6] Russell Eberhart and James Kennedy. 1995. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE, 39–43.

[7] E Emary and Hossam M Zawbaa. 2016. Impact of chaos functions on modern swarm optimizers. *PloS one* 11, 7 (2016), e0158738.

[8] Hossam Faris, Ibrahim Aljarah, and Bashar Al-Shboul. 2016. *A Hybrid Approach Based on Particle Swarm Optimization and Random Forests for E-Mail Spam Filtering*. Springer International Publishing, Cham, 498–508.

[9] Hossam Faris, Mohammad A. Hassonah, Ala' M. Al-Zoubi, Seyedali Mirjalili, and Ibrahim Aljarah. 2017. A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture. *Neural Computing and Applications* (02 Jan 2017).

[10] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182.

[11] John Henry Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[12] A Kaveh. 2017. Chaos embedded metaheuristic algorithms. In *Advances in metaheuristic algorithms for optimal design of structures*. Springer, 375–398.

[13] James Kennedy and Russell C Eberhart. 1997. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, Vol. 5. IEEE, 4104–4108.

[14] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml

[15] Huan Liu and Hiroshi Motoda. 2012. *Feature selection for knowledge discovery and data mining*. Vol. 454. Springer Science & Business Media.

[16] Huan Liu and Lei Yu. 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering* 17, 4 (2005), 491–502.

[17] Majdi Mafarja and Salwani Abdullah. 2013. Investigating memetic algorithm in solving rough set attribute reduction. *International Journal of Computer Applications in Technology* 48, 3 (2013), 195–202.

[18] Majdi Mafarja and Salwan Abdullah. 2014. Fuzzy modified great deluge algorithm for attribute reduction. In *Recent Advances on Soft Computing and Data Mining*. Springer, Cham, 195–203.

[19] Majdi Mafarja and Salwani Abdullah. 2015. A fuzzy record-to-record travel algorithm for solving rough set attribute reduction. *International Journal of Systems Science* 46, 3 (2015), 503–512.

[20] Majdi Mafarja, Derar Eleyan, Salwani Abdullah, and Seyedali Mirjalili. 2017. S-Shaped vs. V-Shaped Transfer Functions for Ant Lion Optimization Algorithm in Feature Selection Problem. In *Proceedings of the International Conference on Future Networks and Distributed Systems*. ACM, 14.

[21] Majdi Mafarja and Seyedali Mirjalili. 2017. Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing* (2017).

[22] Majdi Mafarja and Seyedali Mirjalili. 2017. Whale Optimization Approaches for Wrapper Feature Selection. *Applied Soft Computing* 62 (2017), 441–453.

[23] Seyedali Mirjalili, Amir H Gandomi, Seyedeh Zahra Mirjalili, Shahrzad Saremi, Hossam Faris, and Seyed Mohammad Mirjalili. 2017. Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. *Advances in Engineering Software* 114 (2017), 163–191.

[24] Seyedali Mirjalili and Andrew Lewis. 2013. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation* 9 (2013), 1–14.

[25] Esmat Rashedi and Hossein Nezamabadi-pour. 2014. Feature subset selection using improved binary gravitational search algorithm. *Journal of Intelligent & Fuzzy Systems* 26, 3 (2014), 1211–1221.

[26] Rashi Vohra and Brajesh Patel. 2012. An Efficient Chaos-Based Optimization Algorithm Approach for Cryptography. *Communication Network Security* 1, 4 (2012), 75–79.

[27] Ivan Zelinka and Guanrong Chen. 2010. Motivation for application of evolutionary computation to chaotic systems. *Evolutionary algorithms and chaotic systems* (2010), 3–36.