Expert Systems With Applications 164 (2021) 113873

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Dynamic Salp swarm algorithm for feature selection

Mohammad Tubishat [a,b], Salinah Ja'afar [a,*], Mohammed Alswaitti [c], Seyedali Mirjalili [d], Norisma Idris [b], Maizatul Akmar Ismail [e], Mardian Shah Omar [a]

[a] Department of Linguistics, Academy of Malay Studies University of Malaya, 50603 Kuala Lumpur, Malaysia
[b] Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
[c] School of Electrical and Computer Engineering (ICT), Xiamen University Malaysia, Bandar Sunsuria, 43900 Sepang, Selangor Darul Ehsan, Malaysia
[d] Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Fortitude Valley, Brisbane 4006 QLD, Australia
[e] Department of Information Systems, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

ARTICLE INFO

Keywords:
Salp swarm algorithm
Feature selection
Singer chaotic map
Local search algorithm (LSA)

ABSTRACT

Recently, many optimization algorithms have been applied for Feature selection (FS) problems and show a clear outperformance in comparison with traditional FS methods. Therefore, this has motivated our study to apply the new Salp swarm algorithm (SSA) on the FS problem. However, SSA, like other optimizations algorithms, suffer from the problem of population diversity and fall into local optima. To solve these problems, this study presents an enhanced version of SSA which is known as the Dynamic Salp swarm algorithm (DSSA). Two main improvements were included in SSA to solve its problems. The first improvement includes the development of a new equation for salps' position update. The use of this new equation is controlled by using Singer's chaotic map. The purpose of the first improvement is to enhance SSA solutions' diversity. The second improvement includes the development of a new local search algorithm (LSA) to improve SSA exploitation. The proposed DSSA was combined with the K-nearest neighbor (KNN) classifier in a wrapper mode. 20 benchmark datasets were selected from the UCI repository and 3 Hadith datasets to test and evaluate the effectiveness of the proposed DSSA algorithm. The DSSA results were compared with the original SSA and four well-known optimization algorithms including Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and Grasshopper Optimization Algorithm (GOA). From the obtained results, DSSA outperformed the original SSA and the other well-known optimization algorithms over the 23 datasets in terms of classification accuracy, fitness function values, the number of selected features, and convergence speed. Also, DSSA accuracy results were compared with the most recent variants of the SSA algorithm. DSSA showed a significant improvement over the competing algorithms in statistical analysis. These results confirm the capability of the proposed DSSA to simultaneously improve the classification accuracy while selecting the minimal number of the most informative features.

1. Introduction

Classification represents an important area in many applications which are mainly based on machine learning classifiers such as data mining (Emine and Ülker, 2020; Zhang et al., 2020). However, the performance of these classifiers is mainly influenced by the nature of the features used, in which some of them are frequently irrelevant and noisy. FS can be used to select the most informative features and remove irrelevant or noisy features. Furthermore, FS can reduce features dimensionality (space complexity), decrease the classifier learning time, and improve the classification accuracy (Alhamidi and Jatmiko, 2020; Mafarja et al., 2020; Neggaz et al., 2020; Zhang et al., 2020). There are two main methods of FS, namely, filter methods and wrapper methods. However, the main problem with the filter method that the features are selected independently without direct contact with the machine learning classifier (Faris, Heidari, et al., 2020). Whereas, the wrapper method selects features using the optimization algorithm and has direct contact with the classifier (Arora and Anand, 2019; Faris, Heidari, et al., 2020). Optimization algorithms have an advantage over the normal exhaustive search, including its ability to select an optimum or close to

* Corresponding author.
E-mail addresses: mtubishat@siswa.um.edu.my (M. Tubishat), b1salina@um.edu.my (S. Ja'afar), alswaitti.mohammed@xmu.edu.my (M. Alswaitti), ali.mirjalili@gmail.com (S. Mirjalili), norisma@um.edu.my (N. Idris), maizatul@um.edu.my (M.A. Ismail), mardianso@um.edu.my (M.S. Omar).

https://doi.org/10.1016/j.eswa.2020.113873
Received 8 November 2019; Received in revised form 30 June 2020; Accepted 8 August 2020
Available online 10 August 2020
0957-4174/© 2020 Published by Elsevier Ltd.

Table 1
SSA related works.

Reference	Problem solved	Remarks
(Ibrahim et al., 2017)	FS	They applied SSA without any improvements for FS on biomedical datasets
(Sayed et al., 2018)	FS	In this work, they substituted c_2 parameter in the SSA algorithm by using chaotic maps. They compared ten chaotic maps types for c_2 parameter and concluded that the logistic chaotic map was the best one. This improvement was used to solve convergence speed and the local optima problem.
(Faris et al., 2018)	FS	Improved SSA exploration by replacing Eq. (4) with the crossover operator of the GA algorithm. They applied a crossover between the current solution and its neighbor instead of using average as the original equation.
(Khamees et al., 2018)	FS	They improved SSA by hybridized it with simulating annealing (SA) to improve its exploitation. SA is employed as a local search algorithm at the end of each SSA iteration to improve the current best solution.
(Ahmed et al., 2018)	FS	In this work, they substituted $c_3$parameter in the SSA algorithm by using chaotic maps. They compared four chaotic maps types (circle map, logistic map, piecewise map, and tent map) for c_3 parameter and concluded that the circle chaotic map was the best one. This improvement was used to make a balance between SSA exploitation and exploration.
(Aljarah, Mafarja et al., 2018)	FS	They improved SSA by dividing the population of salps into a number of sub-chains with each sub-chain has one leader. In addition, for each salp in a given sub-chain, it will update its position using one of the developed four updating strategies, where these strategies were developed to find the value of $c_1$parameter. This improvement was used to make a balance between SSA exploitation and exploration
(Hegazy et al., 2018)	FS	They added a new weight parameter to the SSA positions update equations to improve its convergence speed and make a balance between SSA exploitation and exploration. This new inertia weight parameter takes values over [0,1].
(Meraihi et al., 2018)	Graph coloring problem	They substituted the random parameters c_2 and c_3 in the original SSA using a logistic chaotic map. This improvement was used to solve the local optima problem and improve SSA exploitation and exploration. They applied the improved SSA algorithm on the graph coloring problem.
(Ibrahim et al., 2019)	FS	They hybridized SSA with PSO into a new one called SSAPSO. In their proposed SSAPSO, they defined a new random variable and based on that random value: if it is greater than 0.5, they used SSA to update the salps positions; otherwise, they will use PSO to update the salps positions. This hybridization was used to enhance SSA exploitation and exploration.
(Hegazy et al., 2019)	FS	They substituted all random parameters c_1, c_2, and c_3 in the original SSA algorithm using chaotic maps. They compared five chaotic maps types (Logistic, Piecewise, Singer, Sinusoidal, and Tent) and concluded that the tent chaotic map was the best one. This improvement was used to solve convergence speed and the local optima problem.
(Ateya et al., 2019)	Software-defined networking (SDN)	Improved SSA performance by using chaotic maps. They substituted the value of c_2 parameter in the SSA algorithm by the value obtained from logistic chaotic map. This improvement was used to solve local optima problem.
(Majhi et al., 2019)	Function optimization problem	They changed Eq. (4) by adding the value which was resulted from the chaotic map to the value of the follower position. This improvement was used to improve SSA convergence speed and avoid local optima problem.
(Bentouati et al., 2019)	Power flow problem	They substituted c_2 parameter in SSA algorithm by using logistic chaotic map. This improvement was used to improve SSA convergence speed and exploration.
(Tubishat et al., 2020)	FS	In their work, they improved SSA by using Opposition Based Learning (OBL) at initialization phase of SSA to improve its population diversity. In addition, they proposed local search algorithm to avoid local optima problem. The local search algorithm works by selecting three features randomly from the current best solution. It will change these features to 1 and 0, then it will check if this solution is better than the current best solution it will replace it.
(Faris, Heidari, et al., 2020)	FS	They proposed time-varying technique for the number of leaders and followers during the main loop iteration of SSA. This improvement was included to solve the problems of SSA which include local optima and premature convergence.
(Neggaz et al., 2020)	FS	In their work, they replaced Eq. (4) in original SSA algorithm with update position equations of SCA. The equations of SCA update positions were used in place of Eq. (4) to update the positions of the followers. In addition, they used Disrupt Operator at the end of each iteration to update the whole population. These improvements were included into SSA to improve its exploration and population diversity.

an optimum subset of features within an acceptable amount of time. On the other hand, the exhaustive search requires generating all possible feature subsets to find the solution (if there are M features, then it requires to try 2^Mdifferent solutions to find the optimal subset of features), which is impractical and time-consuming for large-size datasets (Emine and Ülker, 2020; Faris, Heidari, et al., 2020; Mafarja et al., 2020).

Based on the No-Free-Lunch (NFL) theorem (Wolpert and Macready, 1997), there is no single optimizer that is capable of solving all optimization problems and at the same time, it is better than all optimization algorithms. Thus, one optimizer can be superior to other algorithms on a number of problems but not on all problems. Hence, many opportunities are still available such as the development of new optimization algorithms, or by improving one of the available optimization algorithms. In addition, the newly developed or improved algorithms can outperform

```
Initialize the salps (Search-agents) positions x_i (i = 1, 2, ..., n)
while (t < max_iterations)
        find the fitness value of each salp in the chain
        F = best salp position based on fitness value
        Update c_1 by Equation (2)
        for (every salp (x_i))
            if (i ==1)
                Update leader-salp position by Eq (2)
            else
                Update follower-salp position by Eq (4)
            end if
        end for
        update the salps which go further than the search space limits based on the upper
        and lower limits of the problem variables
        t=t+1
end while
return F
```

Fig. 1. The SSA Pseudocode (Mirjalili et al., 2017).

```
T_1 = F
K = 1
while (K < max_LSA_number_of_iterations)
        Number_of_Selected_Features=Rand of 2 or 5
        LSA select Randomly two or five features from T_1 based on Number_of_Selected_Features value
        For each selected features in T_1
            if Feature_value == 1   (1 mean the feature is selected and 0 means not selected)
                Feature_value = 0
            else
                Feature_value =1
            end if
        Calculate the fitness value of T_1
        if f(T_1) < f(F)
            F= T_1
        end if
        K = K + 1
end while
return F
```

Fig. 2. The LSA algorithm Pseudocode.

the previously existing algorithms. the NFL theorem has motivated this study to improve the Salp swarm algorithm (SSA) and to use it for the FS problem.

The SSA algorithm was chosen due to several merits such as simplicity (Masdari et al., 2019), efficiency and flexibility (Abbassi et al., 2019), easy to implement, and it has fewer number of parameters to initialize in comparison to other algorithms (Zhang et al., 2018; Hegazy et al., 2019). The SSA is featured with its highly stochastic nature (Ali et al., 2019). In addition, based on (Mirjalili et al., 2017). The SSA demonstrated its capability to solve problems with various sizes. However, The SSA still has some limitations in some problem types such as population diversity and local optima incident. In order to solve these problems, an improved version (DSSA) of the SSA is proposed in this paper. The proposed DSSA algorithm takes the advantages of the original SSA together with two major improvements. The new proposed DSSA algorithm can avoid stagnating into local optima and improve solutions diversity.

In this study, the main contributions are summarized as follows:

- An improved version DSSA of the original SSA is proposed with two folds:
  - The first improvement includes the use of a Singer chaotic map to select between positions update equations.
  - The development of such a new equation tends to improve the diversity of the solution of SSA.
  - The second improvement includes the development of a new LSA algorithm to improve SSA exploitation, where this LSA algorithm is executed conditionally based on the improvement of the best solution value.
- The development of feature selection in wrapper mode based on the new proposed DSSA.
- The proposed DSSA was tested and evaluated on 20 benchmark datasets from the UCI repository and three Hadiths datasets. From the conducted experiments, the improvements have enhanced the performance of the DSSA algorithm against its peers (SSA, ALO, PSO, GOA, and GA) and confirm its outperformance. In addition, DSSA was compared with the most recent six variants of the SSA algorithm where DSSA has clearly outperformed these SSA variants.

The rest of this paper is organized as follows: Section 2 presents some related works; Section 3 presents some preliminaries and theoretical background details. Section 4 presents the details of the proposed DSSA algorithm. Section 5 presents the details of the conducted experiments

Initialize the salps (Search-agents) positions x_i $(i = 1, 2, \ldots, n)$
find the fitness value of each salp in the chain
F = *best salp position based on fitness value*
while ($t <$ *max_iterations*)
    Update c_1 *by Equation (3)*
    for *(every salp* (x_i)*)*
        if $(i == 1)$
            Update leader-salp position by Eq (2)
        else
            U_value=*Singer_choatic_map using equation (1)*
            if (U_value $<$ 0.5)
                Update follower-salp position by Eq (4)
            else
                Update follower-salp position by new Equation (5)
        end if
    end for
    Repostion the salps which go further than the search space limits.
    find the fitness value of each salp in the chain
    F = *best salp position based on fitness value*
    Find the value of **Improvement_counter** variable
    If $(Improvement\_counter \geq 2)$
        call LSA on F to improve its value
    end if
    $t = t + 1$
end while
return F

Fig. 3 Pseudocode of Proposed DSSA algorithm.

Fig. 3. Pseudocode of Proposed DSSA algorithm.

and the reported results from these experiments. Finally, Section 6 concludes this study.

2. Related works

Recently, many optimization algorithms have been applied to the FS problem in wrapper mode based on their capabilities and achieved promising results. For example (Mafarja and Mirjalili, 2018), used Whale optimization algorithm (WOA) (Hancer et al., 2018), used Artificial bee colony (ABC) (Martarelli and Nagano, 2018), used Constructive Genetic Algorithm (CGA) (Chen et al., 2017), used Bacterial foraging optimization (BFO) (Sayed, Hassanien, et al., 2019), used Crow search algorithm (CSA) (Aljarah, Ala'M, et al., 2018), used Grasshopper optimization algorithm (GOA) (Abualigah et al., 2018), used particle swarm optimization (PSO) (Gu et al., 2018; Aličković and Subasi, 2017), used genetic algorithm (GA) (Sindhu et al., 2017), used sine–cosine algorithm (SCA) (Martarelli and Nagano, 2019), used Biased Random-Key GA (Zhang et al., 2020), used Differential Evolution (DE) (Tawhid and Ibrahim, 2020), used hybrid PSO with Flower Pollination Algorithm (FPA) (Taradeh and Mafarja, 2020), used Thermal Exchange Optimization (TEO) (Faris et al., 2018; Sayed et al., 2018; Hegazy et al., 2019; Faris, Heidari, et al., 2020; Neggaz et al., 2020; Tubishat et al., 2020), used SSA (Thaher et al., 2020), used Harris hawks optimizer (HHO) (Martarelli and Nagano, 2020), proposed three algorithms for FS one based on PSO and two based on biased random-key GA and many more.

Based on the NFL theorem, many alternatives still available to be applied to the FS problem. For this reason, it motivated our work to improve the SSA algorithm to be used in FS. SSA was developed by Mirjalili et al. (2017). The main idea of SSA was inspired by salps behaviors in the ocean. These salps behaviors such as salps navigation and foraging. SSA outperformed other well-known optimization algorithms based on the conducted experiments and obtained results by Mirjalili et al. (2017). In addition, SSA has been applied to different types of problems including FS and achieved competitive performance. For example El-Fergany and Hasanien (2019), used SSA to solve the problem of optimal power flow El-Fergany (2018), used SSA to find the optimal values of polymer exchange membrane fuel cells Ibrahim et al. (2018), used SSA for segmentation of the fish image, Abbassi et al. (2019) used SSA for the identification of parameter values of photovoltaic cell models Abusnaina et al. (2018), used SSA for neural network training, and Zhang et al. (2018) used SSA for the estimation of parameter values

Table 5
Parameters setting of each optimization algorithms.

Algorithm	Parameter settings
GA	Crossover_ratio = 0.9, Mutation_ratio = 0.1 are set identical to setting in (Hegazy et al., 2018, 2019; Arora & Anand, 2019; Arora et al., 2019)
GOA	c_Max = 1, c_Min = 0.00004 are set identical as the authors of the GOA setting (Saremi et al., 2017)
PSO	Acceleration_constants (C1 = 2,C2 = 2), Inertia_Weights (W1 = 0.9, W2 = 0.4) are set identical as setting in (Ibrahim et al., 2019; Emine & Ülker, 2020; Thaher et al., 2020)
SSA	$c_2$and c_3 random numbers over [0,1] are set identical as the authors of SSA setting (Mirjalili et al., 2017)
DSSA	$c_2$and c_3 random numbers over [0,1] are set identical as the authors of SSA setting (Mirjalili et al., 2017) *max_LSA_number_of_iterations* = 10

Table 6
Parameter settings of all competing algorithms for all experiments.

Parameter	Value
Population	10
Number of iterations	100
Number of runs for each Algorithm	30

Table 7
The average classification accuracy of the competing algorithms over 30 runs.

Dataset	SSA	DSSA	PSO	GA	ALO	GOA
Exactly	0.95000	**0.99917**	0.88583	0.79733	0.72133	0.71700
Credit	0.76667	**0.77933**	0.76133	0.74833	0.71967	0.72450
Heart	0.86728	**0.88642**	0.85309	0.83951	0.77346	0.78148
m-of-n	0.99333	**0.99967**	0.97300	0.92017	0.84017	0.85933
Ionosphere	0.92018	**0.94395**	0.91923	0.90639	0.87454	0.87740
Sonar	0.92658	**0.95219**	0.93707	0.90110	0.84038	0.85161
Spect	0.80443	**0.82950**	0.80127	0.77441	0.72420	0.73048
Vehicle	0.75430	**0.76493**	0.73931	0.72729	0.68809	0.70228
Vowel	0.94848	**0.95791**	0.94545	0.93451	0.91229	0.91077
Australian	0.87391	**0.88575**	0.85894	0.82754	0.79783	0.77681
Vote	0.97389	**0.98389**	0.97444	0.96778	0.95056	0.95111
Hill Valley	0.62477	**0.65693**	0.63276	0.62011	0.56321	0.57996
OBS-Network	0.96186	**0.97690**	0.95953	0.94806	0.93891	0.94434
QSAR	0.88452	**0.89858**	0.88199	0.87046	0.84739	0.85371
Spambase	0.92850	**0.93937**	0.92756	0.91861	0.89412	0.90231
Waveform	0.83287	**0.83870**	0.83207	0.82447	0.79903	0.80737
Libras	0.83704	**0.87037**	0.85139	0.83102	0.79213	0.80000
HDS1	0.75949	**0.85356**	0.82178	0.71897	0.55781	0.59823
HDS2	0.89721	**0.96261**	0.94134	0.83116	0.69069	0.76530
HDS3	0.86124	**0.92481**	0.90543	0.80310	0.68682	0.75194
CNAE	0.87670	**0.91836**	0.90201	0.86204	0.77407	0.79907
Penglungew	0.93138	**0.94714**	0.92222	0.90825	0.87521	0.88482
Colon	0.91667	**0.93590**	0.88611	0.88397	0.85150	0.85150

in the curve of water retention in the soil (Faris, Mirjalili, et al., 2020), used SSA to train extreme learning machine (ELM).

However, SSA suffers from some problems like other optimization algorithms such as local optima and population diversity problems. Therefore, two main improvements were included in SSA to solve these problems. In addition, to make it suitable for datasets with different dimensionality. These improvements include 1) New Eq. (5) for followers' update position. In addition, the selection between this new Eq. (5) and original Eq. (4) which already exists in the original SSA is controlled using the singer chaotic map. 2) The development of a local search algorithm (LSA) to be suitable for low, mid, and high dimensional data sets, where this algorithm execution is controlled based on a new variable that tracks the improvement happened on the best solution.

The chaos theory has many chaotic maps types and the main feature of the chaotic map represented by its ability to generate random numbers. These random numbers have some merits such as unpredictable, ergodicity, regularity, and non-repetitive (Gleick and Berry, 1987; dos Santos Coelho and Mariani, 2008). In previous studies, many optimization algorithms were improved by using chaos theory. From these studies, it is confirmed that chaos theory can improve the performance of these optimization algorithms. For example (Liu et al., 2018), improved exploitation of PSO by using chaos theory (Arora and Anand, 2018), improved convergence speed of GOA by using chaos (Kohli and Arora, 2018), improved the convergence speed of Grey Wolf Optimizer (GWO) by using chaos theory (Kaur and Arora, 2018), improved the convergence speed of whale optimization algorithm (WOA) by using chaos theory (Wang et al., 2019), improved population diversity and exploitation of chicken swarm optimization (CSO) algorithm by using chaos theory (Sharma et al., 2019), improved the stochastic nature of Spider monkey Optimization (SMO) algorithm by using chaos theory (Sayed, Tharwat, et al., 2019), improved the convergence speed of Dragonfly Algorithm (DA) by using chaos theory (Kaveh and

Table 8
the average number of selected features of all competing algorithms over 30 runs.

	SSA		DSSA		PSO		GA		ALO		GOA	
Dataset	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Exactly	6.68	1.0561	**6.12**	**0.1131**	7.02	0.2213	7.40	1.897	6.72	0.5567	7.10	3.1891
Credit	9.93	1.7707	**8.15**	**0.1614**	9.72	1.3578	8.36	1.6623	10.21	1.3831	10.03	2.2177
Heart	5.34	1.4475	**4.95**	0.5227	5.95	1.0540	5.13	**0.4922**	6.07	1.1243	6.36	1.9812
m-of-n	7.03	1.0936	**6.16**	**0.5611**	6.90	0.8788	7.70	1.4462	6.98	0.9185	7.17	1.6790
Ionosphere	14.06	3.1970	**9.81**	2.0016	15.54	**1.8892**	14.73	2.4066	15.94	2.1192	16.27	2.2233
Sonar	27.11	3.5513	**18.96**	**2.1412**	27.05	3.4199	25.66	3.0911	29.28	3.7383	30.08	6.1966
Spect	10.38	1.6625	**8.58**	**1.4554**	10.89	2.6117	10.33	1.6551	10.31	1.6001	10.78	3.0018
Vehicle	8.55	1.7711	**8.29**	**1.3223**	8.86	1.4910	9.20	1.5333	8.90	1.4733	9.25	2.2055
Vowel	7.84	1.5708	7.29	1.2169	**7.12**	**0.7102**	8.73	1.3830	7.42	1.6110	7.42	2.1654
Australian	5.35	1.0998	**4.99**	**1.0431**	6.16	1.7764	5.03	1.9734	6.34	1.3143	6.62	2.1097
Vote	6.07	2.2240	**5.17**	**1.3621**	7.32	1.6870	6.93	2.2153	7.81	1.4581	8.07	3.1282
Hill Valley	47.30	7.4272	**41.35**	**4.6602**	47.85	6.2244	46.33	5.3314	49.48	6.5552	49.66	8.2221
OBS-Network	6.91	1.8819	**4.76**	**1.4765**	8.98	2.5270	5.40	1.5401	9.96	1.5003	9.41	2.0067
QSAR	19.41	2.9910	**15.94**	**2.6958**	20.40	3.3971	19.23	3.0515	20.85	3.9482	20.25	6.2986
Spambase	29.05	2.7153	**26.89**	**2.3139**	29.02	3.1845	30.96	3.1114	28.69	3.9472	29.30	5.3653
Waveform	13.23	2.1768	12.80	**2.1313**	11.48	2.7987	14.60	3.4291	**11.28**	2.6785	11.88	5.0028
Libras	42.16	4.7157	**31.25**	**3.6999**	42.54	5.3944	40.83	4.8262	44.44	5.7832	44.79	8.2975
HDS1	412.81	23.1187	**316.93**	**13.3858**	408.17	20.8865	399.80	22.0021	413.21	18.1620	410.59	34.8070
HDS2	745.56	34.6166	**635.66**	**23.2957**	745.94	27.8800	747.23	35.1023	749.27	37.9104	748.75	40.2367
HDS3	590.30	21.7881	**477.83**	**15.1887**	582.53	19.1807	580.10	26.2333	587.99	31.2322	588.11	33.8799
CNAE	424.10	18.7771	**335.02**	**15.2735**	425.81	25.4107	421.67	19.0003	428.24	34.8866	428.02	16.7537
Penglungew	148.43	7.3930	**92.40**	**4.5207**	151.99	10.5547	135.87	10.1119	158.33	9.9243	159.56	18.8045
Colon	960.72	42.6156	**819.31**	**20.5814**	970.47	51.4807	892.97	61.5517	989.51	31.4443	988.57	38.4203

Table 9
The average fitness values of all competing algorithms over 30 runs.

	SSA		DSSA		PSO		GA		ALO		GOA	
Dataset	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Exactly	0.05460	0.0237	**0.00561**	**0.0049**	0.11861	0.0066	0.20633	0.0414	0.28132	0.0771	0.28630	0.05662
Credit	0.23573	0.0079	**0.22213**	**0.0058**	0.24080	0.0081	0.25333	0.0095	0.28258	0.0129	0.27751	0.0301
Heart	0.13508	0.0089	**0.11585**	0.0061	0.14898	**0.0049**	0.16284	0.0113	0.22856	0.0218	0.22097	0.0178
m-of-n	0.01185	0.0045	**0.00522**	0.0023	0.03232	0.0078	0.08495	**0.0011**	0.16403	0.0019	0.14547	0.0330
Ionosphere	0.08242	0.0061	**0.05775**	0.0041	0.08318	0.0186	0.09700	0.0227	0.12869	**0.0021**	0.12574	0.0247
Sonar	0.07669	0.0091	**0.05008**	**0.0039**	0.06552	0.0217	0.10219	0.0173	0.16283	0.0288	0.15181	0.0325
Spect	0.19830	0.0181	**0.17275**	**0.0042**	0.20150	0.0081	0.22803	0.0517	0.27751	0.0671	0.27163	0.0097
Vehicle	0.24788	0.0105	**0.23727**	**0.0057**	0.26286	0.0114	0.27510	0.0093	0.31367	0.0087	0.30008	0.0467
Vowel	0.05728	0.0038	**0.04792**	**0.0000**	0.06012	0.0092	0.07155	0.0466	0.09326	0.0077	0.09448	0.0718
Australian	0.12802	0.0147	**0.11637**	**0.0031**	0.14284	0.0121	0.17433	0.0081	0.20456	0.0461	0.22543	0.1299
Vote	0.02130	0.0041	**0.01867**	**0.0029**	0.02857	0.0088	0.03623	0.0374	0.05376	0.0068	0.05325	0.0548
Hill Valley	0.37593	0.0029	**0.34347**	**0.0012**	0.36784	0.0233	0.38073	0.0066	0.43729	0.0123	0.42065	0.1180
OBS-Network	0.03985	0.0129	**0.02429**	**0.0092**	0.04212	0.0239	0.05399	0.0719	0.06493	0.0699	0.05902	0.0233
QSAR	0.11870	0.0471	**0.10410**	**0.0135**	0.12116	0.0144	0.13294	0.0220	0.15624	0.0216	0.14957	0.0169
Spambase	0.07593	0.0319	**0.06491**	**0.0137**	0.07687	0.0177	0.08601	0.0616	0.10989	0.0155	0.10200	0.0955
Waveform	0.17249	0.0155	**0.16637**	**0.0071**	0.17294	0.0248	0.18073	0.0097	0.20472	0.0082	0.19672	0.0088
Libras	0.16553	0.0097	**0.13137**	**0.0080**	0.15091	0.0411	0.17183	0.0494	0.21061	0.0096	0.20284	0.0334
HDS1	0.24300	0.0467	**0.14849**	**0.0022**	0.18105	0.0318	0.28304	0.0819	0.44273	0.0915	0.40263	0.0733
HDS2	0.10665	0.0798	**0.04106**	**0.0015**	0.06292	0.0455	0.17214	0.0665	0.31122	0.0543	0.23730	0.0925
HDS3	0.14227	0.0363	**0.07825**	**0.0038**	0.09833	0.0597	0.19985	0.0917	0.31500	0.0883	0.25052	0.0891
CNAE	0.12691	0.0117	**0.08443**	**0.0076**	0.10177	0.0081	0.14151	0.0159	0.22868	0.0090	0.20385	0.0089
Penglungew	0.07202	0.0211	**0.05442**	**0.0162**	0.08049	0.0179	0.09501	0.0442	0.12828	0.0337	0.11868	0.0209
Colon	0.08704	0.0282	**0.06717**	**0.0034**	0.11698	0.0366	0.11933	0.0175	0.15192	0.0165	0.15182	0.0272

Javadi, 2019), improved convergence speed and make a balance between its exploitation and exploration of the firefly algorithm (FA) by using chaos theory (Sayed, Hassanien, et al., 2019), improved convergence speed of CSA and prevent it from falling into local optima by using chaos theory. Hence, based on the literature investigation, the use of chaos theory has improved the optimization algorithms performance. As a consequence of previous achievements and the demonstrated potential of chaos theory, it gave us the inspiration to combine the SSA algorithm with chaos theory. Therefore, the singer's chaotic map is combined with SSA to improve the balance between SSA exploration and exploitation. In addition, the new equation is added to update the salps position based on the best solution. This equation with the singer chaotic map is used to improve solutions diversity and the SSA convergence speed.

In literature, different types of local search techniques had been hybridized with different optimization algorithms. For example (Toksari, 2016), improved the Ant Colony Optimization (ACO) by using Iterated Local Search (ILS) algorithm (Mavrovouniotis et al., 2017), improved ACO using a local search operator (Riahi and Kazemi, 2018), improved the exploitation ability of ACO by using SA as a local search operator (Mafarja and Mirjalili, 2017), improved WOA by using simulated annealing (SA). They used SA to check if there is a better solution than the current, where SA was executed at the end of each WOA iteration (Oh et al., 2004), improved the searchability of GA and prevent it from falling into local optima by using Local search operations (Asadzadeh, 2015), improved the best solution of GA and its population diversity by using a local search procedure (Abdel-Basset et al., 2018), improved WOA using the local search strategy (Moradi and Gholampour, 2016), improved the exploitation of PSO and solved its premature convergence problem using a local search strategy (Marinakis et al., 2017), improved the particles' positions of PSO using variable neighborhood search (VNS) as the local algorithm (Sakamoto et al., 2018), improved the convergence speed of PSO using hill climbing (Nekkaa and Boughaci, 2016), improved Harmony search (HS) exploitation using stochastic local search (SLS) (Kato et al., 2018), improved the particles' positions of PSO using Random-Restart Hill Climbing (Lin and Guan, 2018), improved the exploitation of PSO by using ILS (Yan et al., 2019), they improved the exploitation of Coral reefs optimization (CRO) using SA as a local operator at the end of each CRO iteration (Shehab et al., 2017), improved the exploration and convergence speed of the Cuckoo Search Algorithm (CSA) by using hill climbing as a local search strategy (Abed-alguni and Alkhateeb, 2018), solved the premature convergence problem of CSA by using the βhill-climbing algorithm (Al-Betar, 2017; Ou et al., 2016), improved the exploitation of Bird-mating optimization (BMO) using hill climbing on the current best solution at the end of each BMO iteration (Zhao et al., 2019), improved the exploitation of Biogeography-based optimization (BBO) using variable neighborhood search (VNS) as the local algorithm at the end of each BBO iteration on the current best solution (Sulaiman et al., 2018), improved the exploitation of artificial bee colony (ABC) using a local search technique called evolutionary gradient search (EGS) (Pei et al., 2019), improved the convergence speed of Bat algorithm (BA) using VNS as a local search operator. Based on the obtained results from these improved algorithms, it is confirmed that local search techniques can improve the performance of these optimization algorithms. Thus, in this

Table 10
The p-values based on the Wilcoxon test of all competing algorithms in terms of the average classification accuracy over 30 runs ($P \geq 0.05$ are bolded and underlined).

Dataset	SSA	PSO	GA	ALO	GOA
Exactly	2.85E−05	6.43E−09	4.08E−11	1.67E−12	1.69E−12
Credit	4.53E−02	4.44E−03	6.32E−06	2.80E−09	3.63E−09
Heart	7.41E−03	2.59E−03	2.10E−05	6.17E−09	1.93E−10
m-of-n	1.24E−03	2.42E−04	4.27E−10	7.17E−12	6.84E−12
Ionosphere	1.99E−03	3.79E−03	5.96E−05	7.86E−09	9.74E−09
Sonar	1.92E−03	4.79E−02	3.51E−06	9.02E−10	6.25E−09
Spect	1.32E−02	4.11E−03	1.65E−05	1.90E−10	2.70E−09
Vehicle	3.51E−02	3.18E−04	1.15E−06	1.14E−10	1.01E−09
Vowel	2.00E−02	5.68E−03	1.04E−05	2.62E−09	3.13E−08
Australian	4.93E−02	3.70E−06	1.11E−08	4.49E−11	5.01E−11
Vote	2.13E−02	3.93E−02	1.43E−03	3.50E−07	1.67E−07
Hill Valley	2.98E−03	1.03E−02	1.21E−03	1.63E−09	1.10E−08
OBS-Network	9.64E−03	2.00E−03	1.27E−05	1.92E−07	2.50E−07
QSAR	4.50E−03	2.30E−03	3.43E−07	4.02E−10	1.77E−09
Spambase	1.52E−06	5.95E−07	6.60E−10	2.99E−11	3.64E−11
Waveform	2.54E−02	2.74E−02	1.69E−06	6.27E−11	1.30E−10
Libras	3.20E−03	**6.67E−02**	4.97E−04	9.46E−08	7.18E−07
HDS1	2.39E−07	6.70E−03	4.46E−09	2.80E−11	3.21E−11
HDS2	3.51E−07	4.92E−03	2.29E−09	2.71E−11	5.72E−11
HDS3	1.43E−05	**1.22E−01**	3.18E−09	2.52E−11	3.31E−11
CNAE	2.44E−09	1.23E−03	4.57E−09	3.02E−11	3.02E−11
Penglungew	5.56E−03	1.86E−03	7.69E−04	2.87E−06	1.09E−05
Colon	1.70E−02	4.71E−04	3.37E−04	2.28E−05	2.28E−05

Table 11
The average execution time (in seconds) of all competing algorithms over 30 runs.

Dataset	SSA	DSSA	PSO	GA	ALO	GOA
exactly	**7.3133**	10.6142	7.8908	7.9293	7.9645	8.5151
credit	11.5629	16.1637	11.9659	**11.3876**	12.6310	12.8693
heart	**5.8879**	8.5615	6.0040	6.0167	6.3986	6.5211
m-of-n	**10.1709**	13.9863	10.6710	11.0321	10.3438	11.5054
ionosphere	**7.3738**	10.7088	7.5033	7.5055	8.9498	9.2464
sonar	**5.8763**	8.3357	5.9819	6.0522	8.2967	8.8577
spect	**6.4297**	9.4147	6.5263	6.5096	7.2771	7.4696
vehicle	**10.1825**	14.7774	10.2914	10.6597	11.0833	11.4326
vowel	**11.8322**	18.0451	12.2818	12.3767	11.8630	12.5042
Australian	**7.5136**	10.9680	7.7853	7.9590	8.5368	8.5167
vote	**5.8151**	8.6307	6.0438	6.3408	7.0544	6.7502
Hill Valley	**11.372**	15.9411	11.8444	11.8518	15.9094	16.8029
OBS-Network	9.9193	13.8742	10.4199	**9.8698**	11.9232	11.8737
QSAR	14.1878	18.9808	14.2465	**14.1713**	16.2971	16.5882
spambase	117.8354	160.8999	119.4960	123.6934	**114.9288**	123.6163
waveform	83.0819	113.0517	79.8227	86.7007	**66.2986**	78.2362
libras	**9.6967**	13.4219	9.7767	9.8740	13.8349	15.5360
HDS1	**8.8475**	11.9894	8.9185	9.5002	38.0964	40.6548
HDS2	**13.8070**	16.1694	13.9154	14.3816	64.0405	71.0344
HDS3	**11.0413**	14.0452	11.2123	11.0510	51.0133	58.2193
CNAE	**65.4802**	72.8175	65.9936	65.7346	82.8435	85.278
penglungew	6.2824	7.9081	6.1569	**6.1492**	17.8298	21.2288
Colon	4.7024	6.4262	4.6431	**4.5998**	47.3695	62.4746

study, a new local search algorithm (LSA) is developed and used to improve SSA exploitation. This LSA works based on making perturbations on the selected features from the current best solution. This LSA will be executed only if there is no improvement on the current best solution after a number of iterations.

Moreover, a different version of the SSA algorithm has been improved using chaos theory or local search algorithms (Faris, Mirjalili, et al., 2020). In addition, SSA or its improved variants have been also applied to the FS problem. To show the difference between our proposed DSSA and other related works that utilized SSA or its improved variants Table 1, presents a comprehensive summary of these SSA related works.

In our previous work (Tubishat et al., 2020), we employed OBL and local search to improve some of the solutions obtained during the optimization process by SSA. In this work, however, the main equations of the SSA algorithm have been modified to change its search patterns and better solve feature selection problems. Based on Table 1, the proposed DSSA is different from these works in the following terms:

Table 12
Comparison of the DSSA classification accuracy against classification without features selection.

Dataset	DSSA	No Features Selection
Exactly	**0.99917**	0.69400
Credit	**0.77933**	0.67800
Heart	**0.88642**	0.66667
m-of-n	**0.99967**	0.86300
Ionosphere	**0.94395**	0.84334
Sonar	**0.95219**	0.80325
Spect	**0.82950**	0.66303
Vehicle	**0.76493**	0.66904
Vowel	**0.95791**	0.87071
Australian	**0.88575**	0.67101
Vote	**0.98389**	0.89667
Hill Valley	**0.65693**	0.48847
OBS-Network	**0.97690**	0.90605
QSAR	**0.89858**	0.80379
Spambase	**0.93937**	0.79765
Waveform	**0.83870**	0.81520
Libras	**0.87037**	0.73889
HDS1	**0.85356**	0.66312
HDS2	**0.96261**	0.86773
HDS3	**0.92481**	0.82326
CNAE	**0.91836**	0.86574
Penglungew	**0.94714**	0.75846
Colon	**0.93590**	0.77051

1) The development of a new update position Eq. (5) to update the positions of the followers in the salps chain.
2) The proposal of a new variable $U_v alue$ where its value obtained based on singer chaotic map. However, in other algorithms which employed chaotic maps, they replaced the original parameters of SSA including c_1, c_2, and c_3 parameters as shown in Table 1. The new variable $U_v alue$ will be used to select between the old Eq. (4) and the new Eq. (5) to update followers' positions.
3) The development of the LSA algorithm to improve the best solution. This algorithm works by selecting two or five features from the best solution randomly. Then, it will change the features with 1 to 0 and vice versa. The major difference of this work is the random selection of two or five features to cover datasets with different dimensionality ranging from low to high, and this is clearly observed from DSSA performance from the obtained results on datasets with different dimensionality.
4) The introduction of a new variable called Improvement$_c$ounter, where this variable is added to control the execution of LSA based on best solution improvement. This variable is used to reduce the complexity of DSSA.
5) In this study, the developed DSSA algorithm has been applied to three newly collected Hadiths datasets on Arabic, English, and Malay languages.

3. Preliminaries

This section provides a brief background of the Chaotic map and original SSA with their corresponding analogies and mathematical representations.

3.1. Chaotic map

In normal random number generators, the random number is generated based on probability distributions. Thus, to generate an unbiased random number, many types of research in literature used chaotic maps (Jana et al., 2018; Kaur and Arora, 2018). The chaos map is a non-linear deterministic system that can generate random numbers with a number of characteristics. The generated numbers by this system involve the unpredictability, ergodicity, regularity, and non-repetitiveness (Zhang and Feng, 2018). Chaos has the ability to improve the solutions' diversity of the search algorithm based on its ergodic property (Arora and Anand, 2018).

Table 13
The average classification accuracy of DSSA in comparison to related variants of SSA 30 runs.

Dataset	DSSA	(Ahmed et al., 2018)	(Aljarah, Mafarja, et al., 2018)	(Faris et al., 2018)	(Hegazy et al., 2019)	(Ibrahim et al., 2019)	(Neggaz et al., 2020)
exactly	**0.99917**	0.96711	0.99693	0.98030	0.96126	0.98755	0.98030
credit	0.77933	*0.77123*	*0.76716*	*0.76891*	**0.78212**	*0.76831*	*0.76900*
heart	0.88642	*0.86956*	*0.83309*	*0.86050*	*0.82810*	*0.84700*	**0.90560**
m-of-n	**0.99967**	0.99512	0.99920	0.99180	0.99656	0.99423	0.98750
ionosphere	0.94395	*0.93134*	*0.93769*	*0.91820*	*0.89770*	0.95100	**0.98500**
sonar	0.95219	*0.92767*	*0.94808*	*0.93720*	*0.74800*	0.96200	**0.99680**
spect	0.82950	0.81333	0.83333	0.83610	0.81614	0.80736	**0.93890**
vehicle	0.76493	*0.75700*	*0.75913*	*0.75516*	*0.64340*	**0.76544**	*0.76111*
vowel	**0.95791**	0.94911	0.95014	0.94893	0.94887	0.95179	0.95202
Australian	0.88575	*0.87434*	**0.88912**	*0.87886*	*0.87655*	*0.87746*	*0.88107*
vote	**0.98389**	0.95424	0.95489	0.95110	0.96747	0.97132	0.98060
Hill Valley	0.65693	*0.63313*	*0.63744*	**0.65713**	*0.53390*	*0.62768*	*0.64300*
OBS-Network	**0.97690**	0.96761	0.96457	0.96221	0.96538	0.97154	0.96481
QSAR	**0.89858**	0.88911	0.88787	0.88987	0.89141	0.88722	0.89182
spambase	**0.93937**	0.93600	0.92962	0.93324	0.89560	0.89200	0.90700
waveform	0.83870	*0.81000*	*0.73643*	*0.73350*	**0.84190**	*0.79130*	*0.76360*
libras	**0.87037**	0.83919	0.84917	0.86254	0.85141	0.84510	0.86455
HDS1	**0.85356**	0.76128	0.77247	0.76556	0.78422	0.78230	0.79577
HDS2	**0.96261**	0.89978	0.91221	0.92373	0.91778	0.90900	0.92810
HDS3	**0.92481**	0.87765	0.88182	0.89328	0.87134	0.90406	0.90705
CNAE	**0.91836**	0.87916	0.88194	0.87763	0.89480	0.89120	0.89701
penglungew	0.94714	*0.93933*	*0.90721*	*0.87750*	*0.93321*	*0.93391*	**1.00000**
Colon	**0.93590**	0.82321	0.85826	0.68600	0.87203	0.88606	0.85444

As mentioned earlier, chaos was combined with many optimization algorithms and improved the performance of these algorithms. There are many types of chaotic maps, and one type of these chaotic maps is the singer chaotic map. The equation of the singer map is presented in Eq. (1)

$$x_{k+1} = \mu\left(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.302875x_k^4\right) \quad (1)$$

where μ is the control parameter with any value in $[0.9, 1.08]$ and $x_0 \in (0,1)$ and $x_k \in (0,1)$

3.2. The original Salp swarm algorithm (SSA)

Salp swarm algorithm (SSA) is one of the most recently developed swarm optimization algorithms which was developed by Mirjalili et al. (2017). The main idea behind the SSA algorithm is to mimic the swarming behavior of salps in the ocean based on the idea of the salps chain. The salps belong to the Salpidae's family which are barrel-shaped creatures. In addition, Salps tissues and movements are similar to that of jellyfish (Henschke et al., 2016). Salps have specific swarming behavior throughout their life in the ocean. This behavior is called a salp chain, and this salp chain behavior can be used in their movements for food search.

Based on the mentioned salps behavior (Mirjalili et al., 2017), formulated this salp chain into mathematical equations and optimization algorithm. The SSA starts the search process by generating a population of salps. Then, the population is divided into two different groups. One is called the leader and the other is called the followers. The leader represents the chain head which guides the followers in the chain, while the other salps represent the followers. Each salps position is represented as n dimensions in the search space, where n represents the number of variables in the problem. In addition, the food source F represents the target of salps to search for. The mathematical model of SSA is represented by the following equations, where the meanings of all notations used in these equations are shown in Appendix A in Table 2.

Therefore, the following equations are used to update the leader of the salp chain in SSA (Mirjalili et al., 2017)

$$x_j^1 = \begin{cases} F_j + c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right)c_3 \geq 0.5 \\ F_j - c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right)c_3 < 0.5 \end{cases} \quad (2)$$

The value of the parameter c_1 will be determined using Eq. (2) (Mirjalili et al., 2017) and c_1 will be calculated as follows

$$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \quad (3)$$

Eq. (4) will be used to update the followers' positions (Mirjalili et al., 2017)

$$x_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right) \quad (4)$$

In Eq. (4), the value of i must be set to $i \geq 2$ (e.g. that it is a follower; otherwise when the value is 1 this means it is a leader). The original SSA algorithm pseudocode is shown in Fig. 1. The SSA starts by initializing salps positions (population). After that, the objective function will be used to find the fitness value of each salp. The fittest salp will be considered as the current best solution (Food Source F).

Furthermore, in every iteration of the SSA, the value of the parameter c_1 will be updated by Eq. (3). The best solution represents the leader position. To update the leader position, SSA uses Eq. (2). To update the followers' positions, SSA uses Eq. (4). The SSA steps will be repeated based on the specified max number of iterations. As shown in Fig. 1, the required parameters by SSA are the number of salps n and the max number of iterations.

4. The proposed Dynamic Salp swarm algorithm (DSSA)

This section outlines the main improvements which are proposed in the original SSA algorithm. The first improvement includes the use of Singer chaotic map to choose between Eq. (4) or the newly developed Eq. (5) for updating the followers' positions. Based on Singer's chaotic value, if it is less than 0.5, then Eq. (4) will be used to update the follower position; otherwise, the newly developed Eq. (5) will be used to update the follower position.

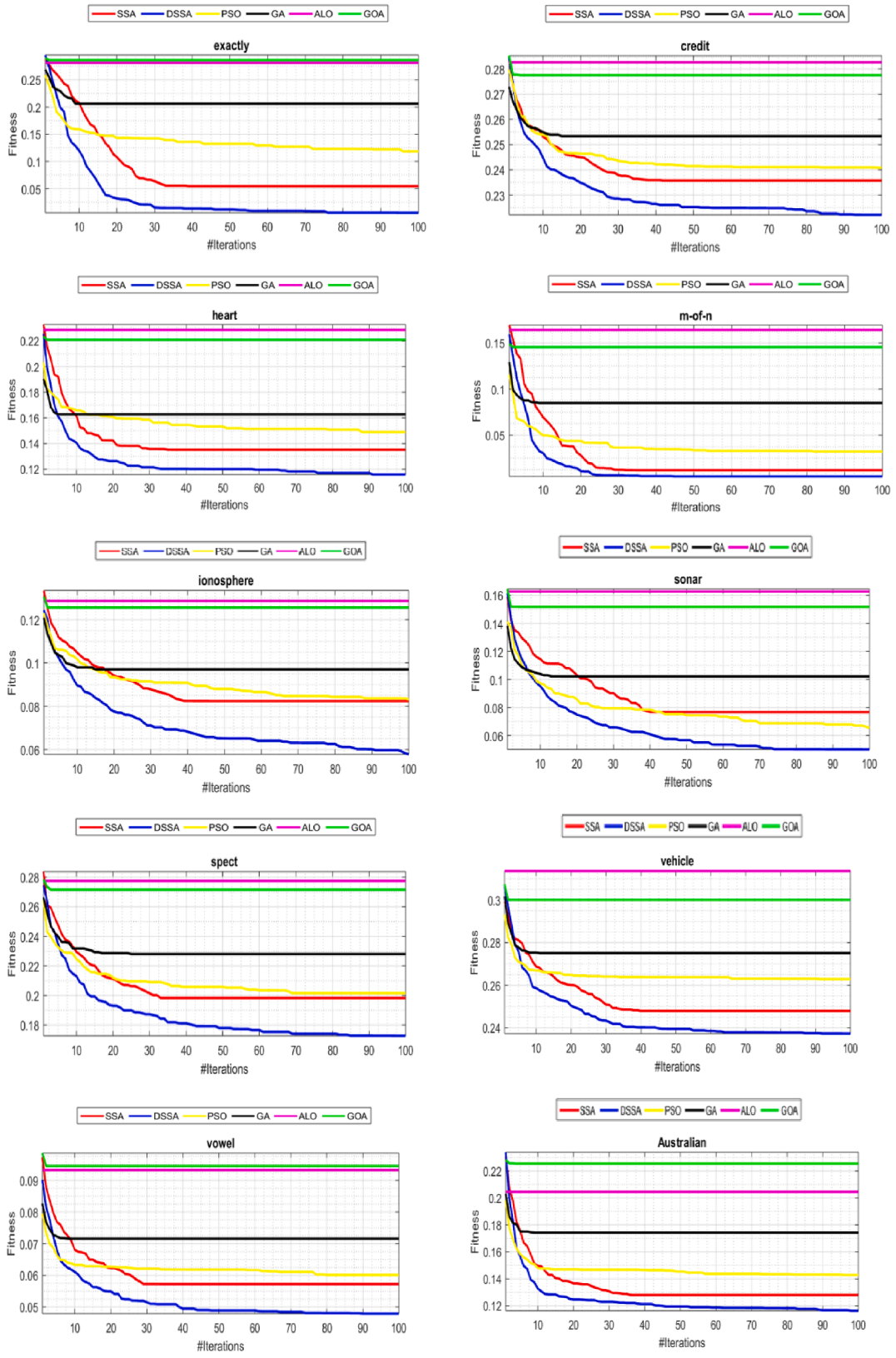

Fig. 4. Comparison of Convergence curves of all competing algorithms over the 23 datasets.

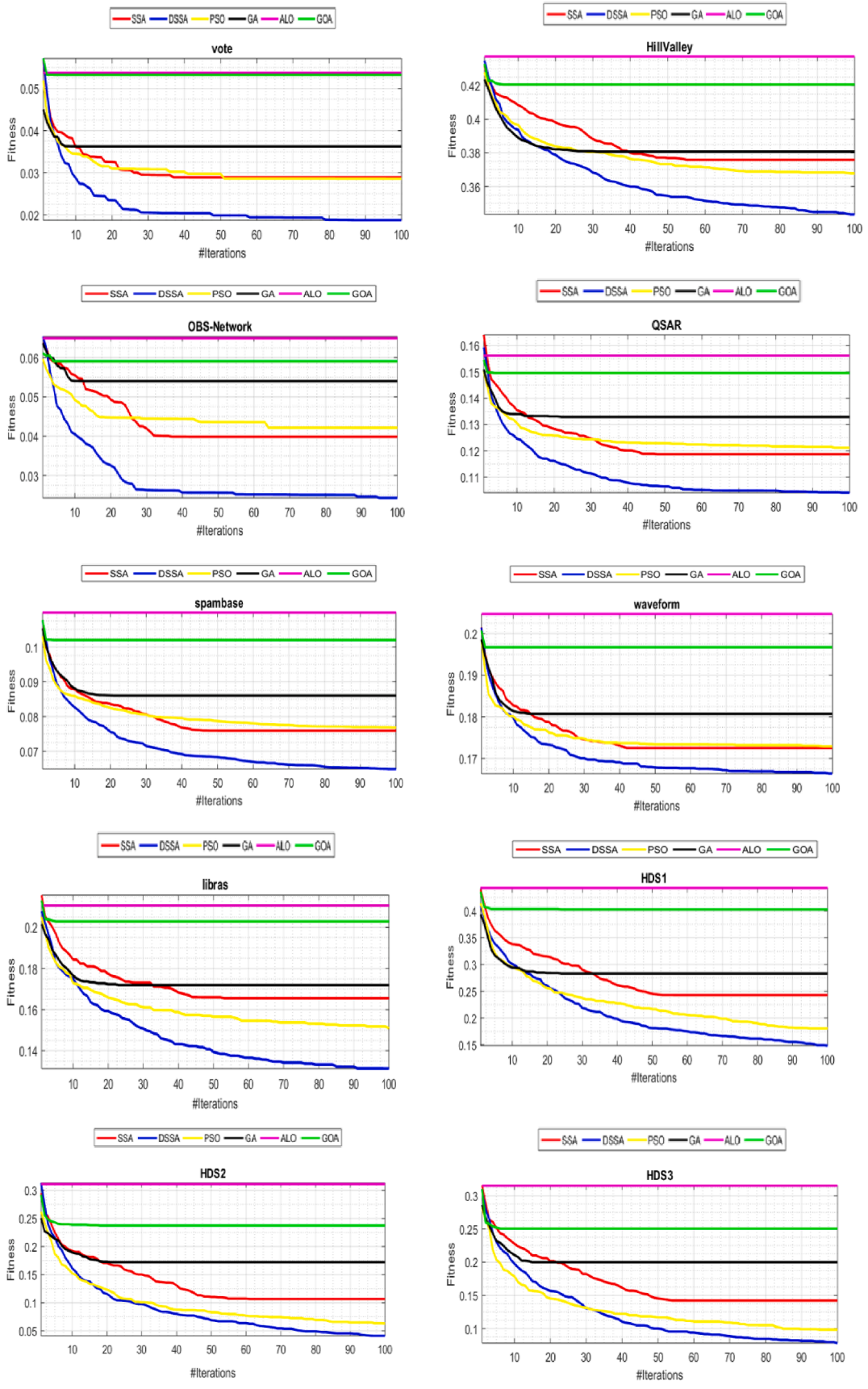

Fig. 4. (*continued*).

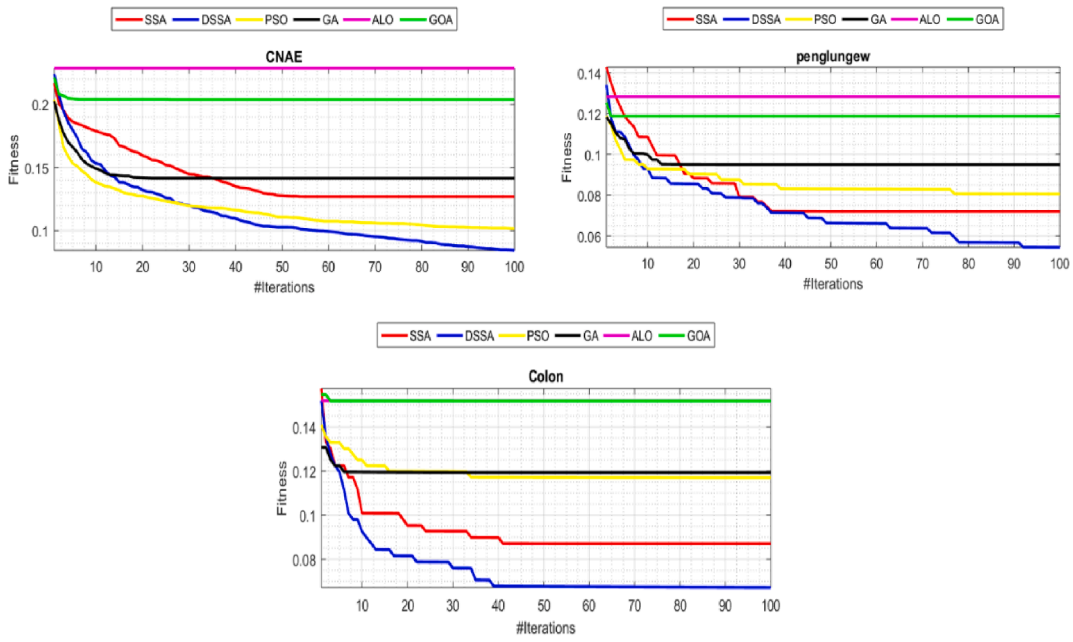

Fig. 4. (*continued*).

$$x_j^i = \frac{1}{3}(x_1 + x_2 + x^*) \tag{5}$$

The description of the used notations in DSSA and LSA equations is shown in Appendix A in Tables 2 and 3 respectively.

The main aim of the first improvement is to enhance solutions diversity and make a balance between exploitation and exploration in the SSA. The second improvement to the original SSA includes the conditional use of the LSA algorithm to improve the current best solution (SSA exploitation). LSA will be called at the end of each SSA iteration on the current best solution *F* to check if there is a better solution. However, LSA will not be used at each iteration to avoid more computational requirements. Therefore, a new counter variable (*"Improvement_counter"*) was added at the end of the SSA to track the improvement on the best solution *F* value. If the value of *"Improvement_counter* $\geq$ *2"*, then this means that two iterations of the SSA occurred without improvements on *F* value. Therefore, the LSA algorithm will be called to improve the value of the current best solution *F*. Fig. 2 shows the newly developed LSA algorithm. The LSA starts by making the necessary initializations. At each LSA iteration, two or five features will be selected randomly based on the value of the *"Number_of_Selected_Features"* variable which will be generated randomly. The value *"Number_of_Selected_Features"* is either two or five (two or five to cover both large and small datasets). Now, the values of the selected features will be flipped to either 0 (not selected feature) or 1 (selected feature). In addition, the LSA will find the fitness value of the new solution T_1 and if it is better than the current *F* best solution, then LSA will update *F* and set its value to T_1 value. The overall improvements to SSA are shown in Fig. 3, which displays the pseudo-code of the proposed DSSA algorithm based on the mentioned improvements.

Fig. 3 presents the structure of the proposed DSSA algorithm. This DSSA algorithm includes two improvements, one of which is the use of Singer chaotic map to select between Eq. (4) or the newly developed Eq. (5) for updating followers' positions. The other improvement includes the use of the LSA, where LSA is conditionally executed based on the value of the *"Improvement_counter"* variable. The DSSA starts by generating a random number of population (Salps). Each solution (Salp) is represented in terms of binary values (*"1" means selected feature and "0" means not_selected feature*). Also, the FS problem can be considered as a multi-objective optimization problem that concerns with achieving higher classification accuracy with a minimal number of included features (selected features) in the solution. Then, DSSA will calculate the food fitness of each solution based on the fitness function. Now, the solution with the minimum fitness value will be assigned the current best solution (Food source). Therefore, the fitness objective function which is based on KNN will guide DSSA to select the best solution (food source). In the second stage, DSSA will update the positions of salps based on Eqs. (2), (4), and (5), respectively. Each iteration, the fitness value will be recalculated to update the best solution (food source). In the end, when DSSA reaches the maximum number of iterations, it will return the best solution as a vector of binary values, Thus, the selected features in this solution will be used by KNN on the testing part of the dataset. For testing and evaluation of the DSSA on the FS problems, DSSA was used with K-Nearest Neighbor (KNN) classifier in a wrapper-mode. The proposed DSSA algorithm is tested and evaluated on 20 datasets as outlined in the next section.

The computational complexity of the DSSA in the best case is equal to $O(t(d \times n + Cof \times n))$ where *t* represents the number of DSSA iterations, *d* represents the dimension of problem to be solved (number of variables), *n* is the population size (number of generated solutions by DSSA), and *Cof* represents the complexity of the fitness function. Therefore, DSSA's best complexity achieved when the LSA is not called, and this complexity is equal to the complexity of the original SSA algorithm. The worst case of complexity means that LSA always executed after every two iterations of the main loop, which is equal to $O(t(d \times n + Cof \times n) + t/2 \times u)$, where *u* represents the *max_LSA_number_of_iterations.*

5. Experimental results and discussion

All algorithms were implemented using MATLAB. To evaluate the performance of the proposed DSSA algorithm, 20 benchmark datasets were used from the University of California at Irvine (UCI) as shown in Appendix A in Table 4. In addition, three Hadiths datasets with a total of 23 datasets were used in all experiments. These datasets are ranging from low, mid, to high dimensionality to test the efficiency and applicability of the DSSA on datasets with different dimensions. The Hadiths datasets were collected from the Sahih Al-Bukhari book and the English and Malay translated versions of the Sahih Al-Bukhari book. HDS1 dataset was collected from the Arabic version of Sahih Al-Bukhari (https://sunnah.com/bukhari#). HDS2 was collected from the Malay translated version of Sahih Al-Bukhari from (www.ibnumajjah.com).

Table 2
The description of the notations used in SSA mathematical equations.

Notation	Description
F	The target (food source) of salps to search for
x_j^1	Leader position in the chain in the j^{th} dimension
F_j	The food source position in the j^{th} dimension
lb_j	The lower bound of the j^{th}dimension
ub_j	The upper bound of the j^{th} dimension
c_1	The controller parameter for the balance between exploitation and exploration in SSA
c_2**and**c_3	Random values over the interval [0,1]
l	The current iteration
L	The maximum number of iterations of SSA to find a solution
x_j^i	The follower position demonstrated by the i^{th} follower position in the j^{th} search-space dimension

Table 3
The description of the notations used in DSSA and LSA mathematical equations.

Notation	Algorithm	Description
x_1 and x_2	DSSA	Represent any two randomly selected solutions from the populations of salps
x^*	DSSA	Represents the current best solution
Improvement_counter	DSSA	Track the improvement on the best solution F value to control the call of LSA algorithm
U_value	DSSA	Produce chaotic value based on singer chaotic map which will be used to select between Eq. (4) or new Eq. (5) to update the positions of the followers
Number_of_Selected_Features	LSA	A random value which can be 2 or 5
T1	LSA	Used to store F (current best solution at end of each SSA iteration)
K	LSA	Used to store the value of LSA current iteration

HDS3 dataset was collected from the English translated version of Sahih Al-Bukhari from (https://sunnah.com/bukhari#). These three Hadiths datasets were preprocessed and converted to a Term frequency-inverse document frequency (TF-IDF) matrix. The details of all datasets are presented in Appendix A in Table 4.

In the conducted experiments, the wrapper-FS-mode was used with the KNN classifier. Furthermore, in the conducted experiments, each dataset was evaluated using K-fold where k = 5 (the number of folds), in which one-fold was used for testing, while the other folds were used for training. For the datasets shuffling, the cross-fold (K-fold) validation was applied on these datasets which used by the majority of research works on FS problem (Ahmed et al., 2018; Aljarah, Mafarja et al., 2018; Faris et al., 2018; Hegazy et al., 2019; Ibrahim et al., 2019; Neggaz et al., 2020). K-fold cross-validation is based on using different random parts each time of the dataset, such as one part is chosen randomly for testing and the other parts also will be chosen randomly for training. Also, for features, it will be shuffled by the optimization algorithms. The optimization algorithm will generate the initial solutions by selecting features randomly from the full set of features. To get statistically reliable and repeatable results, each experiment was conducted 30 times for each dataset and the average of these 30 runs was reported in terms of the accuracy, fitness value, the number of selected features, convergence curves, and run time.

The proposed DSSA algorithm was compared to the original SSA and four other well-known optimization algorithms including the PSO, GA, ALO, and GOA. The parameter settings were used for each algorithm is shown in Table 5. Theses parameters were set identical to the given reference settings which are shown in Table 5.

The details of the other experimental settings are shown in Table 6. The main performance metric for evaluating the proposed algorithm in all experiments is classification accuracy.

The performance of the DSSA in comparison to the standard SSA and other optimization algorithms (PSO, GA, ALO, and GOA) is reported in Table 7, where the best-obtained results are bolded. Based on the reported results, the DSSA has outperformed all other optimizers in terms of the average classification accuracy on all 23 datasets. The surpassing of the DSSA resulted from its ability to escape the local optima using the LSA and the improvements to the solutions' diversity by employing the singer chaotic map and the new update equation. By examining the accuracy results in Table 7, the superiority of the DSSA is confirmed with a clear difference in the majority of datasets. These obtained results by DSSA emphasizes its stability to balance the exploitation and exploration by selecting the most informative features. In addition, it emphasizes its stability to perform very well over datasets with different dimensions.

Based on the number of selected features as shown in Table 8, the DSSA outperformed the standard SSA in all 23 datasets as it selects fewer features. Furthermore, the DSSA outperformed other algorithms including PSO, GA, ALO, and GOA in 21 out of 23 datasets in feature reduction as it selects the lowest number compared to these algorithms. The reason behind selecting a fewer number of features by the proposed DSSA is imputed to the employed LSA algorithm. Each iteration, the LSA flips the bits of the solution and improves the best solution by selecting the features that are relevant while discarding the irrelevant ones. This consequently led to an improvement in the average classification accuracy. It is noteworthy that the included modifications to the DSSA has improved its exploitation and exploration capabilities and gave it more searchability over datasets ranging from low to high dimensions. To illustrate the stability of the obtained results by the proposed DSSA in terms of features reduction, the standard deviation (STD) values were compared to other competing algorithms. As shown from Table 8, DSSA is clearly outperformed other algorithms (best STD and AVG results are indicated using bold font). Similarly, the effect of the modifications and the achieved search balance enhanced the obtained fitness values by the DSSA. Table 9 lists the average fitness and STD values of all competing algorithms and confirms the outperformance of the DSSA over all other algorithms. As shown in Table 9, DSSA is clearly outperformed other algorithms (best STD and AVG results are indicated using bold font). These results confirm the stability and the repeatability of the DSSA algorithm over datasets of different dimensionality.

To evaluate the significance of the reported results, Wilcoxon's statistical test was used. This test can assess the significance and robustness of the DSSA compared to the original SSA and the other optimization algorithms. Wilcoxon's statistical test was used by the majority of studies which were conducted on FS problems such as the recently published studies in (Sayed et al., 2018; Ibrahim et al., 2019; Abdel-Basset et al., 2020; Arora et al., 2020; Emine and Ülker, 2020; Faris, Heidari, et al., 2020; Li et al., 2020). Also, in these studies, they used UCI datasets in their experiments. In this test, the used significance level is 5% which was determined based on the results of the average classification accuracy over 30 runs. Table 10 presents the ***P* values** which were obtained from the Wilcoxon test of all competing algorithms. In this test, if the ***P*-value** is less than 5%, this means that the DSSA has significant improvement over the competing algorithms. otherwise, if ***P*-value** is greater than 5% (underlined), there is no significant difference. The results in Table 10 shows that the DSSA has significant improvement over the other algorithms in most of the datasets in terms of the average classification accuracy.

To further evaluate and scalability of the DSSA, Fig. 4 shows the average convergence curves of the competed algorithms over the 23 datasets. It is clearly observed that the DSSA is the fastest in terms of convergence speed amongst all datasets while achieving the best fitness values. These results prove the suitability of the included modifications to the DSSA. Thus, it is obviously noticed from the conducted

Table 4
Description of the experimental datasets.

#	Dataset	Number of features	Number of instances
1	Exactly	13	1000
2	Credit	20	1000
3	Heart	13	270
4	m-of-n	13	1000
5	Ionosphere	34	351
6	Sonar	60	208
7	Spect	22	267
8	Vehicle	18	846
9	Vowel	13	990
10	Australian	14	690
11	Vote	16	300
12	Hill Valley	100	606
13	OBS-Network	21	1075
14	QSAR	41	1055
15	Spambase	57	4601
16	Waveform	21	5000
17	Libras	90	360
18	HDS1	828	214
19	HDS2	1498	227
20	HDS3	1178	215
21	CNAE	856	1080
22	Penglungew	325	73
23	Colon	2000	62

experiments that DSSA always performs better than the original SSA algorithm. Such outperformance of the DSSA over SSA comes from the use of the LSA to improve the exploitation ability of the DSSA and prevent it from stagnating into local optima. Another reason is due to the employment of a Singer chaotic map to select between Eq. (4) or the new update Eq. (5). This improvement proved DSSA's ability to balance exploration and exploitation whilst improving the solutions' diversity.

In order to validate the modifications' effect on the execution time of the DSSA, Table 11 lists the average execution time of all competing algorithms. It is noted that the SSA, GA, ALO performed better in 16, 5, 2 out of 23 datasets in terms of average execution time over all other algorithms, respectively. As shown in Table 11, there is a slight increase in the DSSA execution time compared to other algorithms due to calling the LSA. Although the DSSA requires more execution time in some datasets than other algorithms, it significantly outperforms other algorithms in terms of accuracy, feature reduction, fitness values, and statistical results as reported earlier, which is a matter of compromise.

Lastly, To highlit the importance of the FS phase by the DSSA algorithm, Table 12 shows a comparison between the DSSA accuracy and the accuracy without using the FS phase. The reported results confirm the significance of the FS phase as it achieved better average classification accuracy due to decreasing the dimensionality of these datasets and removing the redundant, irrelevant, and noisy features.

Taking all the aforementioned results into consideration, the DSSA showed that it is a rival competitor when applied to solve the FS problems on a wide range of datasets. Firstly, the DSSA avoids the local optima problem by employing the LSA. Furthermore, the DSSA achieves the balance between exploration and exploitation using the singer chaotic map with the newly developed updating equation. The effect of all modifications enhanced the DSSA ability to provide higher quality solutions over other optimization algorithms in terms of the average classification accuracy, fitness function values, and convergence speed. These results conclude that the DSSA is a promising choice when applied to FS problems.

To confirm the superiority of the proposed DSSA, we compared DSSA with the most recent and related improved variants of SSA including works conducted by Ahmed et al. (2018), Aljarah, Mafarja et al. (2018), Faris et al. (2018), Hegazy et al. (2019), Ibrahim et al. (2019), and Neggaz et al. (2020). All parameter settings follow the original authors' configurations as reported in their respective manuscripts. The results of the conducted experiments are shown in Table 13. As shown from the accuracy results, DSSA outperformed other variants on 13 out of 23 datasets (***best results are indicated using bold font***). For the related works, it can be observed that the algorithms proposed by Aljarah, Mafarja et al. (2018), Faris et al. (2018), and Ibrahim et al. (2019) achieved only the best result on one dataset for each. In addition, the algorithm proposed by Hegazy et al. (2019) obtained the best results on two datasets, where the rival competitor algorithm proposed by Neggaz et al. (2020) attained best results on 5 out of the 23 used datasets. The aforementioned results are reasonable due to the continuous improvements over the original SSA creates more variants with stronger abilities to solve diverse kinds of problems.

It is worth mentioning that although the DSSA did not achieve the best results in the other 10 datasets, it obtained comparative results compared to the six related works. (*it is indicated by highlighting all other works that achieved accuracy results less than the DSSA with shaded color and italic font*). For example, on **credit**, **heart**, **vehicle**, **Australian**, **Hill Valley**, **waveform**, and **penglungew** datasets, DSSA got the 2nd rank compared to other SSA variants. Furthermore, on the **ionosphere**, and **sonar** datasets DSSA got the 3rd rank compared to other SSA variants works. Also, DSSA got the 4th rank on the **spect** dataset. These results confirmed the stability of the proposed DSSA over datasets of different dimensionalities.

Therefore, based on these achieved results of the DSSA algorithm as shown in Table 13, it is clearly confirmed that the included improvements into DSSA have made it more suitable for feature selection problems. This is imputed to the improvements that made the DSSA less prone to the local optima incident and increased the diversification of the solutions throughout the search. Consequently, the DSSA had the ability to outperform other related variants of SSA in different types of datasets with different dimensionality ranging from low to high scales.

6. Conclusion

In this work, an improved variant of the original SSA algorithm is introduced in order to solve SSA local optima problem and achieve the balance between exploration and exploitation. The new dynamic SSA (DSSA) was combined with the KNN classifier in a wrapper mode for FS problems. Two improvements were included in DSSA. The first improvement includes the use of a Singer chaotic map with a newly developed equation to update the Salp position. Furthermore, Singer chaotic map was used to select between the update equations. This improvement was used to improve solutions' diversity and balance exploration and exploitation. The other improvement includes the development of a new LSA algorithm to improve DSSA exploitation and avoid it from falling into local optima. In addition, a counter variable was used to track the continuous improvement of the best solution in each DSSA iteration. Based on this counter variable, LSA will be called. This counter variable was used to reduce the computational complexity of the DSSA. The proposed DSSA was evaluated on 23 datasets and compared with the original SSA and other four well-known optimization algorithms including PSO, GA, ALO, and GOA. DSSA results are superior to all algorithms in classification accuracy, fitness values, and convergence speed. In addition, the DSSA has outperformed SSA in all datasets in terms of features reduction, while it outperformed other algorithms in 18 out of 20 datasets. Moreover, based on significance statistical results, the DSSA performed better and showed significant enhancement over the other competing algorithms. Also, the DSSA sowed a faster convergence rate over the competing algorithms in all tested datasets.

Further future works can be conducted such as using the new LSA algorithm with the proposed conditional variable to be hybridized with other metaheuristic algorithms. Moreover, the effect of the new update equation on other optimization algorithms could be investigated. In addition, the new DSSA algorithm can be applied to different types of problems such as engineering applications, industry applications, sentiment analysis, functions optimization, cybersecurity applications, and parameter optimization.

CRediT authorship contribution statement

Mohammad Tubishat: Conceptualization, Data curation, Resources, Investigation, Methodology, Writing - original draft, Software, Validation, Writing - review & editing. **Salinah Ja'afar:** Funding acquisition, Project administration, Resources, Supervision, Writing - review & editing. **Mohammed Alswaitti:** Funding acquisition, Resources, Validation, Writing - review & editing. **Seyedali Mirjalili:** Formal analysis, Writing - review & editing. **Norisma Idris:** Supervision, Writing - review & editing. **Maizatul Akmar Ismail:** Project administration, Writing - review & editing. **Mardian Shah Omar:** Resources, Data curation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research work was financially supported by the University of Malaya under the University Malaya Research Grant UMRG RP043C-17HNE; and Xiamen University Malaysia Research Fund (Grant no. XMUMRF/2019-C4/IECE/0012).

Appendix A. Description of used notations and datasets

References

Abbassi, R., Abbassi, A., Heidari, A. A., & Mirjalili, S. (2019). An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. *Energy Convers. Manage., 179*, 362–372.

Abdel-Basset, M., El-Shahat, D., El-henawy, I., de Albuquerque, V. H. C., & Mirjalili, S. (2020). A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. *Expert Syst. Appl., 139*, Article 112824.

Abdel-Basset, M., Manogaran, G., El-Shahat, D., & Mirjalili, S. (2018). A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Future Generation Computer Systems, 85*, 129–145.

Abed-alguni, B. H., & Alkhateeb, F. (2018). Intelligent hybrid cuckoo search and β-hill climbing algorithm. Journal of King Saud University-Computer and Information Sciences.

Abualigah, L. M., Khader, A. T., & Hanandeh, E. S. (2018). A new feature selection method to improve the document clustering using particle swarm optimization algorithm. *J. Comput. Sci., 25*, 456–466.

Abusnaina, A. A., Ahmad, S., Jarrar, R., & Mafarja, M. (2018). Training neural networks using salp swarm algorithm for pattern classification. Paper presented at the Proceedings of the 2nd International Conference on Future Networks and Distributed Systems.

Ahmed, S., Mafarja, M., Faris, H., & Aljarah, I. (2018). Feature selection using salp swarm algorithm with chaos. Paper presented at the Proceedings of the 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence.

Al-Betar, M. A. (2017). $$\beta $$-Hill climbing: an exploratory local search. *Neural Comput. Appl., 28*(1), 153–168.

Alhamidi, M. R., & Jatmiko, W. (2020). Optimal feature aggregation and combination for two-dimensional ensemble feature selection. *Information, 11*(1), 38.

Ali, T. A. A., Xiao, Z., Sun, J., Mirjalili, S., Havyarimana, V., & Jiang, H. (2019). Optimal design of IIR wideband digital differentiators and integrators using salp swarm algorithm. *Knowl.-Based Syst., 104834*.

Aličković, E., & Subasi, A. (2017). Breast cancer diagnosis using GA feature selection and Rotation Forest. *Neural Comput. Appl., 28*(4), 753–763.

Aljarah, I., Ala'M, A.-Z., Faris, H., Hassonah, M. A., Mirjalili, S., & Saadeh, H. (2018). Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. *Cognit. Comput.*, 1–18.

Aljarah, I., Mafarja, M., Heidari, A. A., Faris, H., Zhang, Y., & Mirjalili, S. (2018). Asynchronous accelerating multi-leader salp chains for feature selection. *Appl. Soft Comput., 71*, 964–979.

Arora, S., & Anand, P. (2018). Chaotic grasshopper optimization algorithm for global optimization. *Neural Comput. Appl.*, 1–21.

Arora, S., & Anand, P. (2019). Binary butterfly optimization approaches for feature selection. *Expert Syst. Appl., 116*, 147–160.

Arora, S., Sharma, M., & Anand, P. (2020). A novel chaotic interior search algorithm for global optimization and feature selection. *Appl. Artif. Intell., 34*(4), 292–328.

Arora, S., Singh, H., Sharma, M., Sharma, S., & Anand, P. (2019). A new hybrid algorithm based on Grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection. *IEEE Access, 7*, 26343–26361.

Asadzadeh, L. (2015). A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Comput. Ind. Eng., 85*, 376–383.

Ateya, A. A., Muthanna, A., Vybornova, A., Algarni, A. D., Abuarqoub, A., Koucheryavy, Y., & Koucheryavy, A. (2019). Chaotic salp swarm algorithm for SDN multi-controller networks. *Eng. Sci. Technol. Int. J.*

Bentouati, B., Javaid, M., Bouchekara, H., & El-Fergany, A. A. (2019). Optimizing performance attributes of electric power systems using chaotic salp swarm optimizer. *Int. J. Manage. Sci. Eng. Manage.*, 1–11.

Chen, Y.-P., Li, Y., Wang, G., Zheng, Y.-F., Xu, Q., Fan, J.-H., & Cui, X.-T. (2017). A novel bacterial foraging optimization algorithm for feature selection. *Expert Syst. Appl., 83*, 1–17.

dos Santos Coelho, L., & Mariani, V. C. (2008). Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. *Expert Syst. Appl., 34*(3), 1905–1913.

El-Fergany, A. A. (2018). Extracting optimal parameters of PEM fuel cells using salp swarm optimizer. *Renew. Energy, 119*, 641–648.

El-Fergany, A. A., & Hasanien, H. M. (2019). Salp swarm optimizer to solve optimal power flow comprising voltage stability analysis. *Neural Comput. Appl.*, 1–17.

Emine, B., & Ülker, E. (2020). AN EFFICIENT BINARY SOCIAL SPIDER ALGORITHM FOR FEATURE SELECTION PROBLEM. Expert Systems with Applications, 113185.

Faris, H., Heidari, A. A., Ala'M, A.-Z., Mafarja, M., Aljarah, I., Eshtay, M., & Mirjalili, S. (2020). Time-varying hierarchical chains of salps with random weight networks for feature selection. *Expert Syst. Appl., 140*, Article 112898.

Faris, H., Mafarja, M. M., Heidari, A. A., Aljarah, I., Ala'M, A.-Z., Mirjalili, S., & Fujita, H. (2018). An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowl.-Based Syst., 154*, 43–67.

Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M., & Heidari, A. A. (2020). Salp swarm algorithm: theory, literature review, and application in extreme learning machines Nature-Inspired Optimizers (pp. 185–199): Springer.

Gleick, J., & Berry, M. (1987). Chaos-making a new science. *Nature, 330*, 293.

Gu, S., Cheng, R., & Jin, Y. (2018). Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft. Comput., 22*(3), 811–822.

Hancer, E., Xue, B., Zhang, M., Karaboga, D., & Akay, B. (2018). Pareto front feature selection based on artificial bee colony optimization. *Inf. Sci., 422*, 462–479.

Hegazy, A. E., Makhlouf, M., & El-Tawel, G. S. (2018). Improved salp swarm algorithm for feature selection. *J. King Saud Univers. –Comput. Inform. Sci.*

Hegazy, A. E., Makhlouf, M., & El-Tawel, G. S. (2019). Feature selection using chaotic salp swarm algorithm for data classification. *Arab. J. Sci. Eng., 44*(4), 3801–3816.

Henschke, N., Everett, J. D., Richardson, A. J., & Suthers, I. M. (2016). Rethinking the role of salps in the ocean. *Trends Ecol. Evol., 31*(9), 720–733.

Ibrahim, A., Ahmed, A., Hussein, S., & Hassanien, A. E. (2018). Fish image segmentation using Salp Swarm algorithm. Paper presented at the International Conference on Advanced Machine Learning Technologies and Applications.

Ibrahim, H. T., Mazher, W. J., Ucan, O. N., & Bayat, O. (2017). Feature selection using salp swarm algorithm for real biomedical datasets. *IJCSNS, 17*(12), 13.

Ibrahim, R. A., Ewees, A. A., Oliva, D., Elaziz, M. A., & Lu, S. (2019). Improved salp swarm algorithm based on particle swarm optimization for feature selection. *J. Ambient Intell. Hum. Comput., 10*(8), 3155–3169.

Jana, N. D., Sil, J., & Das, S. (2018). Continuous fitness landscape analysis using a chaos-based random walk algorithm. *Soft. Comput., 22*(3), 921–948.

Kato, E. R. R., de Aguiar Aranha, G. D., & Tsunaki, R. H. (2018). A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing. *Comput. Ind. Eng., 125*, 178–189.

Kaur, G., & Arora, S. (2018). Chaotic whale optimization algorithm. *J. Comput. Des. Eng., 5*(3), 275–284.

Kaveh, A., & Javadi, S. (2019). Chaos-based firefly algorithms for optimization of cyclically large-size braced steel domes with multiple frequency constraints. *Comput. Struct.*

Khamees, M., Albakry, A., & Shaker, K. (2018). Multi-objective feature selection: hybrid of salp swarm and simulated annealing approach. Paper presented at the International Conference on New Trends in Information and Communications Technology Applications.

Kohli, M., & Arora, S. (2018). Chaotic grey wolf optimization algorithm for constrained optimization problems. *J. Comput. Des. Eng., 5*(4), 458–472.

Li, M., Wang, H., Yang, L., Liang, Y., Shang, Z., & Wan, H. (2020). Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. *Expert Syst. Appl., 150*, Article 113277.

Lin, G., & Guan, J. (2018). A hybrid binary particle swarm optimization for the obnoxious p-median problem. *Inf. Sci., 425*, 1–17.

Liu, W., Luo, N., Pan, G., & Ouyang, A. (2018). Chaos particle swarm optimization algorithm for optimization problems. *Int. J. Pattern Recognit Artif Intell., 32*(11), 1859019.

Mafarja, M., & Mirjalili, S. (2018). Whale optimization approaches for wrapper feature selection. *Appl. Soft Comput., 62*, 441–453.

Mafarja, M., Qasem, A., Heidari, A. A., Aljarah, I., Faris, H., & Mirjalili, S. (2020). Efficient hybrid nature-inspired binary optimizers for feature selection. *Cognit. Comput., 12*(1), 150–175.

Mafarja, M. M., & Mirjalili, S. (2017). Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing, 260*, 302–312.

Majhi, S. K., Mishra, A., & Pradhan, R. (2019). A chaotic salp swarm algorithm based on quadratic integrate and fire neural model for function optimization. *Prog. Artif. Intell.*, 1–16.

Marinakis, Y., Migdalas, A., & Sifaleras, A. (2017). A hybrid particle swarm optimization–variable neighborhood search algorithm for constrained shortest path problems. *Eur. J. Oper. Res., 261*(3), 819–834.
Martarelli, N. J., & Nagano, M. S. (2018). A constructive evolutionary approach for feature selection in unsupervised learning. *Swarm Evol. Comput., 42*, 125–137.
Martarelli, N. J., & Nagano, M. S. (2019). Optimization of the Numeric and Categorical Attribute Weights in KAMILA Mixed Data Clustering Algorithm. Paper presented at the International Conference on Intelligent Data Engineering and Automated Learning.
Martarelli, N. J., & Nagano, M. S. (2020). Unsupervised feature selection based on bio-inspired approaches. *Swarm Evol. Comput., 52*, Article 100618.
Masdari, M., Tahani, M., Naderi, M. H., & Babayan, N. (2019). Optimization of airfoil Based Savonius wind turbine using coupled discrete vortex method and salp swarm algorithm. *J. Cleaner Prod., 222*, 47–56.
Mavrovouniotis, M., Müller, F. M., & Yang, S. (2017). Ant colony optimization with local search for dynamic traveling salesman problems. *IEEE Trans. Cybern., 47*(7), 1743–1756.
Meraihi, Y., Ramdane-Cherif, A., Mahseur, M., & Achelia, D. (2018). A chaotic binary salp swarm algorithm for solving the graph coloring problem. Paper presented at the International Symposium on Modelling and Implementation of Complex Systems.
Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw., 114*, 163–191.
Moradi, P., & Gholampour, M. (2016). A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Appl. Soft Comput., 43*, 117–130.
Neggaz, N., Ewees, A. A., Elaziz, M. A., & Mafarja, M. (2020). Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection. *Expert Syst. Appl., 145*, Article 113103.
Nekkaa, M., & Boughaci, D. (2016). Hybrid harmony search combined with stochastic local search for feature selection. *Neural Process. Lett., 44*(1), 199–220.
Oh, I.-S., Lee, J.-S., & Moon, B.-R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell., 26*(11), 1424–1437.
Ou, T.-C., Su, W.-F., Liu, X.-Z., Huang, S.-J., & Tai, T.-Y. (2016). A modified bird-mating optimization with hill-climbing for connection decisions of transformers. *Energies, 9*(9), 671.
Pei, J., Liu, X., Fan, W., Pardalos, P. M., & Lu, S. (2019). A hybrid BA-VNS algorithm for coordinated serial-batching scheduling with deteriorating jobs, financial budget, and resource constraint in multiple manufacturers. *Omega, 82*, 55–69.
Riahi, V., & Kazemi, M. (2018). A new hybrid ant colony algorithm for scheduling of no-wait flowshop. *Oper. Res. Int. J., 18*(1), 55–74.
Sakamoto, S., Ozera, K., Ikeda, M., & Barolli, L. (2018). Implementation of intelligent hybrid systems for node placement problem in WMNs considering particle swarm optimization, hill climbing and simulated annealing. *Mobile Netw. Appl., 23*(1), 27–33.
Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: theory and application. *Adv. Eng. Softw., 105*, 30–47.
Sayed, G. I., Hassanien, A. E., & Azar, A. T. (2019). Feature selection via a novel chaotic crow search algorithm. *Neural Comput. Appl., 31*(1), 171–188.
Sayed, G. I., Khoriba, G., & Haggag, M. H. (2018). A novel chaotic salp swarm algorithm for global optimization and feature selection. *Appl. Intell., 48*(10), 3462–3481.
Sayed, G. I., Tharwat, A., & Hassanien, A. E. (2019). Chaotic dragonfly algorithm: An improved metaheuristic algorithm for feature selection. *Appl. Intell., 49*(1), 188–205.
Sharma, N., Kaur, A., Sharma, H., Sharma, A., & Bansal, J. C. (2019). Chaotic Spider Monkey Optimization Algorithm with Enhanced Learning Soft Computing for Problem Solving (pp. 149-161): Springer.
Shehab, M., Khader, A. T., Al-Betar, M. A., & Abualigah, L. M. (2017). Hybridizing cuckoo search algorithm with hill climbing for numerical optimization problems. Paper presented at the 2017 8th International Conference on Information Technology (ICIT).
Sindhu, R., Ngadiran, R., Yacob, Y. M., Zahri, N. A. H., & Hariharan, M. (2017). Sine–cosine algorithm for feature selection with elitism strategy and new updating mechanism. *Neural Comput. Appl., 28*(10), 2947–2958.
Sulaiman, N., Mohamad-Saleh, J., & Abro, A. G. (2018). A hybrid algorithm of ABC variant and enhanced EGS local search technique for enhanced optimization performance. *Eng. Appl. Artif. Intell., 74*, 10–22.
Taradeh, M., & Mafarja, M. (2020). Binary Thermal Exchange Optimization for Feature Selection. In *Data Management and Analysis* (pp. 239–260). Springer.
Tawhid, M. A., & Ibrahim, A. M. (2020). Hybrid binary particle swarm optimization and flower pollination algorithm based on rough set approach for feature selection problem. In *Nature-Inspired Computation in Data Mining and Machine Learning* (pp. 249–273). Springer.
Thaher, T., Heidari, A. A., Mafarja, M., Dong, J. S., & Mirjalili, S. (2020). Binary harris hawks optimizer for high-dimensional, low sample size feature selection. In *Evolutionary Machine Learning Techniques* (pp. 251–272). Springer.
Toksari, M. D. (2016). A hybrid algorithm of ant colony optimization (ACO) and iterated local search (ILS) for estimating electricity domestic consumption: case of Turkey. *Int. J. Electr. Power Energy Syst., 78*, 776–782.
Tubishat, M., Idris, N., Shuib, L., Abushariah, M. A., & Mirjalili, S. (2020). Improved Salp Swarm Algorithm based on opposition based learning and novel local search algorithm for feature selection. *Expert Syst. Appl., 145*, Article 113122.
Wang, B., Li, W., Chen, X., & Chen, H. (2019). Improved Chicken Swarm Algorithms Based on Chaos Theory and Its Application in Wind Power Interval Prediction. Mathematical Problems in Engineering, 2019.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Trans. Evol. Comput., 1*(1), 67–82.
Yan, C., Ma, J., Luo, H., & Patel, A. (2019). Hybrid binary coral reefs optimization algorithm with simulated annealing for feature selection in high-dimensional biomedical datasets. *Chemometr. Intell. Labor. Syst., 184*, 102–111.
Zhang, J., Wang, Z., & Luo, X. (2018). Parameter estimation for soil water retention curve using the salp swarm algorithm. *Water, 10*(6), 815.
Zhang, X., & Feng, T. (2018). Chaotic bean optimization algorithm. *Soft. Comput., 22*(1), 67–77.
Zhang, Y., Gong, D.-W., Gao, X.-Z., Tian, T., & Sun, X.-Y. (2020). Binary differential evolution with self-learning for multi-objective feature selection. *Inf. Sci., 507*, 67–85.
Zhao, F., Qin, S., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2019). A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem. *Expert Syst. Appl., 126*, 321–339.