Prediction of Software Defect Using Ensemble Learning Based Improved Sparrow Search Algorithm To Optimize Extreme Learning Machine

Yu Tang  
North China University of Science and Technology

Qi Dai  
China University of Petroleum Beijing

Mengyuan Yang  
North China University of Science and Technology

Lifang Chen (chenlifang@ncst.edu.cn)  
North China University of Science and Technology

Research Article

Keywords: Improved sparrow search algorithm, Extreme learning machine, Bagging, Software defect prediction

Posted Date: November 30th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1100298/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
Prediction of software defect using ensemble learning based improved sparrows search algorithm to optimize extreme learning machine

Yu Tang$^1$ · Qi Dai$^2$ · Mengyuan Yang$^1$ · Lifang Chen$^{1,3}$

$^1$ College of Science, North China University of Science and Technology, Hebei Tangshan 063210, China

$^2$ Department of Automation, College of Information Science and Engineering, China University of Petroleum-Beijing, Beijing, China

$^3$ Hebei Key Laboratory of Data Science and Application, Hebei Tangshan 063210, China

Corresponding author
Email address:  chenlifang@ncst.edu.cn (Lifang Chen)
Postal address: No. 21 Bohai Avenue, Caofeidian New City, North China University of Science and Technology, Tangshan Hebei, 063210.
China
Tel.: +86 132 3082 9011
Prediction of software defect using ensemble learning based improved sparrow search algorithm to optimize extreme learning machine

Yu Tang¹ · Qi Dai² · Mengyuan Yang¹ · Lifang Chen¹,³

Abstract

For the traditional ensemble learning algorithm of software defect prediction, the base predictor exists the problem that too many parameters are difficult to optimize, resulting in the optimized performance of the model unable to be obtained. An ensemble learning algorithm for software defect prediction that is proposed by using the improved sparrow search algorithm to optimize the extreme learning machine, which divided into three parts. Firstly, the improved sparrow search algorithm (ISSA) is proposed to improve the optimization ability and convergence speed, and the performance of the improved sparrow search algorithm is tested by using eight benchmark test functions. Secondly, ISSA is used to optimize extreme learning machine (ISSA-ELM) to improve the prediction ability. Finally, the optimized ensemble learning algorithm (ISSA-ELM-Bagging) is presented in the Bagging algorithm which improve the prediction performance of ELM in software defect datasets. Experiments are carried out in six groups of software defect datasets. The experimental results show that ISSA-ELM-Bagging ensemble learning algorithm is significantly better than the other four comparison algorithms under the six evaluation indexes of Precision, Recall, F-measure, MCC, Accuracy and G-mean, which has better stability and generalization ability.

Keywords

Improved sparrow search algorithm · Extreme learning machine · Bagging · Software defect prediction

1 Introduction

There are various application softwares in our life, which provide great convenience for people's life. The quality of software has become the focus of attention. Most researchers believe that software defects are the main problem affecting software quality and existing risk of leaking user information, resulting in serious information security problem¹,². Therefore, it is issue of this paper is to propose an efficient software defect prediction algorithm to improve software quality and reduce software development and maintenance costs.

Software defect prediction is an important method to measure software quality. Economic losses and information leakage risks will fall with accurately predicting software defect module. If we predict the potential defects of software in time, the work efficiency of testers can be improved³. The existing software defect prediction can be divided into two categories. One is to predict whether the module in the software has defects, and the other is to predict the trend and distribution of defects in the software development process⁴. Efficient software defect prediction algorithm can not only guarantee the quality of software, but also reduce the risk of information leakage⁶,⁷. Therefore, the key issue of this paper is to propose an efficient software defect prediction algorithm to improve software quality and reduce costs of software development and maintenance.

Aiming at the problems of difficult parameter selection, low prediction accuracy and poor stability of base predictor in existing software defect prediction ensemble learning algorithms, the paper proposes a software defect prediction algorithm based on ensemble learning algorithm to optimize extreme learning machine. The main contributions of this paper include the following aspects:

1. An improved sparrow search algorithm is proposed by using the reverse learning of pinhole imaging and the foraging strategy of flip bucket to optimize the sparrow search algorithm.

2. The improved sparrow search algorithm improves the prediction accuracy and stability of extreme learning machine by optimizing the parameters of extreme learning machine.

3. In the Bagging ensemble learning algorithm, an optimized ensemble learning algorithm (ISSA-ELM-Bagging) is proposed to improve the generalization ability of extreme learning machine on software defect prediction datasets.

The rest of this paper is organized as follows. Section 2 introduces the related work of software defect prediction. Section 3 gives optimize the extreme learning machine and ensemble learning algorithm to
improve sparrow search algorithm, and proposes an ensemble learning algorithm of ISSA-ELM-Bagging. Section 4 introduces the experimental setup in detail and analyzes the experimental results. Finally, conclusions and further work are given in Section 5.

2 Related work

With the expansion of software scale and increasing complexity, the quality of software has become the focus of attention. As an important way to guarantee software quality, software defect prediction have been apply many method of data mining and machine learning\[^8\] including, for example, fuzzy logic\[^9\] artificial neural network\[^10\], semi-supervised learning\[^4\], multiple kernel learning\[^11\], extreme learning machine\[^12\], naive bayes\[^13\], support vector machine\[^14\], and so on. In software defect prediction, ensemble learning algorithms, such as Bagging\[^15\] and Boosting\[^16\], have been widely used to improve the prediction performance of software defect prediction algorithm. Reference \[^17\] improved the performance of software defect prediction algorithm used coding-based ensemble learning. Reference \[^18\] verified that Bagging algorithm has better overall performance in ensemble learning.

With the increase of software scale and test cost, researchers have proposed many software defect prediction algorithms. At present, as a machine learning model, extreme learning machine (ELM) has a high prediction ability, but the performance of ELM is easily affected by input weights and hidden layer neuron thresholds. To solve this problem, researchers have used swarm intelligence optimization algorithm to optimize ELM parameters. Liu et al.\[^1\] used cuckoo search algorithm to optimize extreme learning machine parameters (ICS-ELM) to improve the accuracy of junction temperature prediction of insulated-gate bipolar transistor (IGBT). Ding et al.\[^20\] used PSO algorithm to optimize extreme learning machine and improve the accuracy of clinical cancer diagnosis results. Li et al.\[^21\] used the whale optimization algorithm to optimize the extreme learning machine, which has higher accuracy and generalization ability. In recent years, ensemble learning has been widely used in the field of software defect prediction. Ensemble learning can improve the generalization ability of the algorithm by combining multiple weak learners to build strong learners. Related ensemble learning algorithms have also been proposed in the fields of industrial chloride prediction\[^22\], high-dimensional data classification\[^23\], and disease prediction\[^24\]. Reference \[^22\] proposed a new ensemble learning model by combining artificial neural network and stepwise clustering analysis, but did not optimize the model parameters. Reference \[^23\] proved that the multi-view ensemble learning performance (MEL) was superior to the classical ensemble method. When MEL used the particle swarm algorithm for multi-objective optimization, it could quickly search the optimal solution, but the performance of the particle swarm algorithm was not obvious. Reference \[^24\] used four prediction algorithms to predict gas emissions and established an ensemble deep learning model for new type of exhaust gas emissions.

The problem of parameter optimization in the base prediction period is not well solved by the above ensemble learning methods, which leads to obvious performance fluctuation and poor stability of the model. We improve the extreme learning machine (ELM) optimized by sparrow search algorithm (ISSA) as the base predictor of ensemble learning algorithm, and build an ensemble learning algorithm with better prediction accuracy and generalization ability. The experimental results show that the optimized base predictor has better performance than other prediction algorithms on the software defect prediction data sets, and its prediction performance is more stable and generalization ability is stronger.

3 Algorithm design

3.1 Sparrow Search Algorithm

Sparrow search algorithm is a swarm intelligence optimization algorithm proposed by XUE et al.\[^25\] according to a series of behaviors of sparrow groups in the process of foraging. Compared with the traditional swarm intelligence algorithms such as particle swarm optimization (PSO)\[^20\], grey wolf optimizer algorithm (GWO)\[^27\] and whale optimization algorithm (WOA)\[^28\], sparrow search algorithm has stronger optimization ability and faster convergence speed. Sparrow search algorithm can quickly converge near the optimal value, and has good global optimization ability and stability. Sparrow search algorithm consists of three parts: the producers are responsible for foraging and providing direction for the group, the scroungers follow the
producers and the reporters who alert predators. The producers are the best positioned part in the sparrow population and closest to the food; scroungers will update their positions according to the producers to improve the probability of obtaining food; when reporters detect predator’s attacks, they alert the sparrow community, in which the sparrows approach each other to reduce the probability of being preyed and act against predators[29].

3.1.1 The mathematical model of producers
The location of the producers is updated as follows [29]:

\[
X_{i,j}^{t+1} = \begin{cases} 
X^*_i \cdot \exp\left(-\frac{t}{\alpha \cdot \text{iter}_{\max}}\right) & \text{if } R_i < ST \\
X_{i,j}^t + Q \cdot L & \text{if } R_i \geq ST 
\end{cases}
\]

Where \(X_{i,j}^t\) represents the position of the ith sparrow in j-dimensional at the t-iteration, \(\text{iter}_{\max}\) represents the maximum number of iterations, and \(\alpha\) is a random number in the range of (0,1). \(R_i\) is a warning value, and if it is reached, the sparrow population is already in danger and needs to take action (\(R_i \in [0,1]\)). \(ST\) is a safe value, if within the safe value range Sparrow group can move normally (\(ST \in [0.5,1]\)). \(Q\) is a random number that obeys the standard normal distribution. \(L\) represents a matrix of 1\(\times d\), each element in the matrix is 1.

3.1.2 The mathematical model of the scroungers
The location of the scroungers is updated as follows [29]:

\[
X_{i,j}^{t+1} = \begin{cases} 
X_{i,j}^t + \frac{1}{2} |X_{i,j}^t - X_{\text{worst}}^t| & \text{if } f_i > f_u \\
X_{i,j}^t + K \left(\frac{X_{i,j}^t - X_{\text{worst}}^t}{f_u - f_i + \varepsilon}\right) & \text{if } f_i \leq f_u 
\end{cases}
\]

Where \(X_{\text{worst}}\) is currently the global optimal location. \(\beta\) is a random number that obeys the standard normal distribution. \(K\) is a random number from -1 to 1 that represents the direction of the sparrow’s movement and parameters of the step length control. \(f_i\) is the fitness value of the current individual sparrow. \(f_u\) is the current global optimal fitness value, and \(f_u\) is the current worst global fitness value. \(\varepsilon\) is an Infinitesimal constant, avoiding zeros in denominator.

3.2 Pore imaging reverse learning
Aiming at the problem that the sparrow search algorithm is easy to fall into local extremum in the later iteration, Tian et al.[30] applied reverse learning to the sparrow search algorithm, effectively expanding the search range of the algorithm, but the candidate solution obtained by traditional reverse learning is far from the current optimal solution. Aiming at this problem, this paper adopts the inverse learning of pinhole imaging. This method combines the principle of pinhole imaging with the inverse learning, so that the current solution can obtain the candidate solution through the principle of pinhole imaging, which effectively improves the quality of the candidate solution, further avoids the algorithm falling into local optimal solution, and improves the generalization ability of the algorithm. The principle of pinhole imaging reverse learning is shown as follows:

![Fig. 1. Reverse learning principle of pinhole imaging](image-url)
The flame in the $x$-axis is the global optimal position $X_s$, and the upper and lower bounds of the coordinate axis are $a_j, b_j$. The flame $P$ passes through the hole screen at $O$ point, and a flame image $P'$ with height $h'$ can be obtained on the receiving screen. The projection of flame image $P'$ on the $x$ axis is $X'_s$, and the candidate solution corresponding to the global optimal position of $X_s$ is $X'_s$.

According to the principle of pinhole imaging:

$$\frac{(a_j + b_j) / 2 - X'_{best}}{X'_{best} - (a_j + b_j) / 2} = \frac{h}{h'} \quad (4)$$

Substituting $n = h / h'$, the formula becomes:

$$X'_{best} = \frac{a_j + b_j}{2} + \frac{a_j + b_j}{2n} - \frac{X'_{best}}{n} \quad (5)$$

When $n = 1$, we can obtain:

$$X'_{best} = (a_j + b_j) - X'_{best} \quad (6)$$

When $n = 1$, formula (7) is the traditional reverse learning. At this time, the distance between the small hole screen and the receiving screen can be adjusted to obtain a better candidate solution, which effectively improves the quality of the candidate solution and greatly reduces the situation of falling into local optimal solution in the late iteration. A more dynamic candidate solution can be obtained reverse learning of pinhole imaging, which enhances the ability of sparrow search algorithm to jump out of local optimal solution and can find the optimal solution more efficiently.

### 3.3 Fighting foraging strategy

The scroungers find food by following the producers and update their location through the producers, so this also makes the search of the scroungers have certain blindness. This paper introduces the flip foraging strategy and applies it to the location update of the joiner. This strategy can make the joiner update the location more effectively.

$$X_{i}(t+1) = X_{i}(t) + S \cdot (r_1 X_{b} - r_2 X_{i}(t)) \quad i=1,2,\ldots,n \quad (7)$$

where $S$ represents the flip factor that determines the position of the sparrow after the tumbling foraging strategy. According to the reference [31], $S = 2$, $n$ is the number of sparrow, $r_1, r_2$ are $[0,1]$ random numbers. $X_{b}(t)$ is the optimal position of the $t$th iteration, and $X_{b}(t)$ is the position of the $t$th iteration of the $ith$ sparrow. Sparrow flip bucket diagram is as follows:

![Foraging strategy sketch of flip bucket](image)

Introducing the flip bucket foraging strategy in location update of the scroungers, which effectively improves the search ability of the scroungers in the high-dimensional space. The flip bucket foraging strategy is mainly carried out around the optimal solution of each iteration, which also increases the search efficiency of the scroungers in the space. In the early stage of the iteration, the distance between each sparrow is relatively large, and the sparrow can increase the search range of the sparrow through the flip bucket foraging strategy. In the late stage of the iteration, the distance between each solution and the optimal solution is small. When the flip bucket foraging strategy is carried out in a relatively small range, it can not only quickly jump out of the local optimal solution, but also reduce the corresponding time cost.

### 3.4 ISSA algorithm design

The initial number of sparrows in the traditional sparrow search algorithm is small. After a certain number of iterations, the sparrows in the group will gradually approach a certain optimal solution. If the previous generation of sparrow location is not ideal, then continuing to update the individual location of the sparrow will affect the final result, making the algorithm easy to fall into local optimal solution. In order to solve these problems, this paper introduces the inverse learning of pinhole imaging. After each iteration, the candidate solution of the inverse learning of pinhole imaging is obtained according to the producers. Then the fitness value of the producers is compared with the fitness value of the candidate solution. The sparrow with better fitness value is taken as the producers of the current iteration number, which effectively improves the
local search ability, the efficiency of the algorithm and
the diversity of the population. In the traditional sparrow
search algorithm, the adder updates the location
according to the producers, which has certain blindness.
In the early iteration, the search range of the adder is not
wide enough, and it is easy to lose high quality solutions.
Therefore, the flip bucket foraging strategy is introduced
into the location update process of the scroungers. The
flip bucket foraging strategy is a location change around
the optimal solution obtained by the current iteration
number. In the early iteration, the sparrow is relatively
far from the optimal solution, which can increase the
search range of the scroungers and obtain more
high-quality solutions. In the late iteration, the sparrow
individuals are close to the optimal solution, and the
probability of replacing the current optimal solution by
the candidate solution obtained by the foraging strategy
of the flip bucket will gradually increase. Therefore, the
strategy can effectively improve the ability of the
algorithm to jump out of the local optimal solution.
ISSA algorithm can effectively reduce the situation that
the sparrow search algorithm falls into the local optimal
solution, and find the optimal sparrow location and the
optimal fitness value more quickly and efficiently.

3.5 ISSA Optimized Extreme Learning Machine

For dataset $D = \{(x_i, y_i)\}, i = 1, \ldots, l$, the
extreme learning machine model with $n$ hidden layer nodes excitation
function sigmoid can be expressed as[32]:

$$f(x) = \sum_{i=1}^{n} \beta_i g(w_i, b_i, x_i) = \beta \cdot H(x) \tag{8}$$

where $H$ is the hidden layer output matrix and $\beta$ is the output weight of the
hidden layer node. The output weight can be obtained by least square solution of
linear formula (8)[33]:

$$\beta = H^+ \cdot Y \tag{10}$$

where $H^+$ is the Moore-Penrose Pseudoinverse of the hidden layer output matrix $H$. The ELM
structure is shown in the following figure:

![ELM structure](image)

In this paper, ISSA is used to optimize ELM
parameters, and the maximum absolute error of the sum
of sample expected output and actual output in ELM is
selected as the adaptation function of ISSA algorithm.
The specific steps of ISSA algorithm to select the
optimal input weight and hidden layer neuron threshold
are as follows:

step1: Initialize sparrow population and related
parameters;

step2: Calculate the fitness of each sparrow, select
the current optimal location and the optimal fitness
value corresponding to the optimal location, select
the current worst location and the worst location
corresponding to the worst fitness value;

step3: Select the better fitness value of the spa
rrow population as the producers, according to the
formula (1) update the location of the producers;

step4: According to formula (6), the candidate
solution of the reverse learning of the producers' pi
nhole imaging is obtained;

step5: The fitness values of the producers and the
candidate solution are compared, and the better
fitness value is selected as the scroungers;

step6: In addition to the producers sparrow as
scroungers, according to the formula (2) update the
location of the scroungers;

step7: The candidate solution is obtained by the
foraging strategy of the adder according to formula
(7);

step8: Compare the fitness value of the candi
te solution and the scroungers', select the better fitn
ess value as the scroungers;
step 9: Randomly select sparrows as reporters among scroungers and producers, and update the reporters location according to formula (3);
step10: judging whether the algorithm runs to the maximum number of iterations, if it reaches the end of the cycle, if it does not reach step3;
step11: Output global optimal location and optimal fitness value, get the optimal input weights and hidden layer neuron threshold. The ISSA-ELM flowchart is as follows:

3.6 Ensemble learning algorithm Bagging
According to reference [35], the superiority of Bagging algorithm in the field of software prediction is verified. This paper adopts the Bagging ensemble learning algorithm. The Bagging algorithm uses the prediction algorithm with low accuracy as the base predictor, and then performs parallel processing on K base predictors. Finally, the voting method is used to obtain the final prediction results[36]. In this paper, the model first uses ISSA to optimize the parameters of ELM as the base predictor of homogeneous integration, and then uses Bagging ensemble learning algorithm to improve the prediction performance of ISSA-ELM base predictor. The Bagging ensemble learning algorithm is shown in the figure:
4 Experimental results and analysis

The experiments are implemented with MATLAB 2014b running on a PC with Intel (R) Core (TM) i7-8565U 1.80 GHz CPU, 16 GB RAM.

4.1 Experimental introduction

This paper compares sparrow search algorithm (SSA), whale algorithm (WOA), grey wolf algorithm (GWO), particle swarm optimization (PSO) with improved sparrow search algorithm (ISSA) for comparison. In order to ensure the fairness of the experiment, the population size of all algorithms are set to 30, and the maximum number of iterations are set to 500[37]. The experimental results retain four decimals, and other parameters are shown in table 1[37].

For the software defect datasets, SSA algorithm and ISSA algorithm select 30 sparrows, and the maximum number of iterations is 100[38]. The number of neurons in the hidden layer is 100, and the activation function is sigmoid[33]. The number of base predictor is the same as the datasets dimension. In this paper, 25 datasets from six groups of NASA, SOFTLAB, AEEEM, AUDI, MORPH and ReLink were selected for simulation experiments. Of these, 80 percent are training sets and 20 percent are test sets. Due to the high data dimension of the software defect datasets, PCA is used to reduce the dimension. The number of principal components after dimensionality reduction is shown in Table 2. The basic information of the dataset is shown in Table 2.

| Algorithm | Parameters |
|-----------|------------|
| ISSA      | ST=0.8, PD=0.2, SD=0.2 |
| SSA       | ST=0.8, PD=0.2, SD=0.2 |
| WOA       | h=1         |
| GWO       | a decline from 2 to 0 |
| PSO       | c1=c2=2, ωmin=0.2, ωmax=0.9 |
Table 2. Introduction of datasets

| group         | characteristic | dataset | number of samples | attribute number | defective number | defect rate | number of principal components |
|---------------|----------------|---------|-------------------|------------------|------------------|-------------|---------------------------------|
| NASA          |                | CM1     | 344               | 2                | 42               | 0.1221      | 10                              |
| NASA          |                | MC2     | 125               | 2                | 44               | 0.3520      | 10                              |
| NASA          |                | MW1     | 263               | 2                | 27               | 0.1027      | 10                              |
| NASA          |                | PC1     | 735               | 2                | 61               | 0.0830      | 11                              |
| NASA          |                | PC3     | 1099              | 2                | 138              | 0.1256      | 12                              |
| NASA          |                | PC2     | 1493              | 2                | 16               | 0.0107      | 9                               |
| NASA          |                | PC4     | 1379              | 2                | 178              | 0.1291      | 13                              |
| SOFTLAB       |                | ar1     | 121               | 2                | 9                | 0.0744      | 8                               |
| SOFTLAB       |                | ar4     | 107               | 2                | 20               | 0.1869      | 8                               |
| SOFTLAB       |                | ar6     | 101               | 2                | 15               | 0.1485      | 7                               |
| AEEEM         |                | EQ      | 324               | 2                | 129              | 0.39814     | 20                              |
| AEEEM         |                | JDT     | 997               | 2                | 206              | 0.2066     | 23                              |
| AEEEM         |                | ML      | 1862              | 2                | 245              | 0.1316      | 24                              |
| AEEEM         |                | PDE     | 1497              | 2                | 209              | 0.1396      | 22                              |
| AUDI          |                | ProjectA| 1908              | 2                | 85               | 0.0445      | 6                               |
| AUDI          |                | ProjectK| 2515              | 2                | 375              | 0.1491      | 6                               |
| MORPH         |                | ant-1.3 | 125               | 2                | 20               | 0.1600      | 10                              |
| MORPH         |                | arc     | 234               | 2                | 27               | 0.1154      | 9                               |
| MORPH         |                | camel-1.0| 339              | 2                | 13               | 0.0383      | 11                              |
| MORPH         |                | pos-1.5 | 237               | 2                | 136              | 0.5738      | 11                              |
| MORPH         |                | rektor  | 176               | 2                | 27               | 0.1534      | 10                              |
| MORPH         |                | velocity-1.4| 196           | 2                | 146              | 0.7449      | 11                              |
| MORPH         |                | xerces-1.2| 440            | 2                | 71               | 0.1614      | 11                              |
| ReLink        |                | apache  | 194               | 2                | 98               | 0.5052      | 7                               |
| ReLink        |                | zxing   | 399               | 2                | 118              | 0.2957      | 7                               |

4.2 ISSA algorithm performance tests

The experiment selects eight different benchmark functions to compare the five optimization algorithms. The benchmark test function information is shown in Table 3. In order to highlight the performance and stability of the algorithm in this paper, the five optimization algorithms are run independently for 30 times on eight benchmark test functions. The optimal values for each function is shown in bold. The specific results are shown in Table 4.

Table 3. Test function table

| Function | DIM | Section | Best |
|----------|-----|---------|------|
| $F_1(x) = \sum_{i=1}^{n} x_i^2$ | 30  | [-100,100] | 0    |
| $F_2(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$ | 30  | [-100,100] | 0    |
| $F_3(x) = \sum_{i=1}^{n} [100(x_i - x_i^2)^2 + (x_i - 1)^2]$ | 30  | [-30,30] | 0    |
| $F_4(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$ | 30  | [-100,100] | 0    |
| $F_5(x) = ta_x^d + random(0,1)$ | 30  | [-1.28,1.28] | 0    |
\[
F_6(x) = \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})
\]

\[
F_7(x) = -20\exp(-0.2 \sqrt{\sum_{i=1}^{n} x_i}) - \exp(-\frac{1}{n} \cos(2\pi x_n)) + 20 + e
\]

\[
F_8 = \frac{\pi}{n} \left( 10\sin(\pi y_i) + \sum_{j=1}^{n} (y_j - 1)^2 [1+10\sin^2(\pi y_j)] + (y_n - 1)^2 \right) + \sum_{j=1}^{n} u(x_j, 10, 100, 4)
\]

\[y_i = 1 + \frac{x_i + 1}{4}\]

\[u(x_i, a, k, m) = \begin{cases} 
  k(x_i - a)^m & x_i > a \\
  0 & -a < x_i < a \\
  k(-x_i - a)^m & x_i < -a
\end{cases} \]

| Function | Algorithm | Best | Worse | Ave | Std |
|----------|-----------|------|-------|-----|-----|
| $F_1$    | ISSA      | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
|          | SSA       | 0.0000 | 3.5969E-175 | 1.1603E-176 | 0.0000 |
|          | GWO       | 2.3995E-27 | 1.8489E-25 | 3.8428E-26 | 4.2843E-26 |
|          | WOA       | 3.3075E-85 | 3.6537E-70 | 2.3076E-71 | 8.1484E-71 |
|          | PSO       | 0.0007 | 1.4262E-05 | 0.0002 | 0.0002 |
| $F_2$    | ISSA      | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
|          | SSA       | 0.0000 | 7.3887E-277 | 2.4629E-278 | 0.0000 |
|          | GWO       | 0.0016 | 9.4610E-08 | 8.9965E-05 | 0.0003 |
|          | WOA       | 15157.9349 | 67242.3333 | 40889.3950 | 10846.4721 |
|          | PSO       | 20.3739 | 177.2701 | 86.3908 | 35.1518 |
| $F_3$    | ISSA      | 1.4533E-07 | 0.0003 | 4.0544E-05 | 5.9781E-05 |
|          | SSA       | 1.3640E-05 | 0.0013 | 0.0002 | 0.0003 |
|          | GWO       | 25.9146 | 27.9571 | 26.8444 | 0.6194 |
|          | WOA       | 27.2115 | 28.7874 | 27.9858 | 0.4924 |
|          | PSO       | 22.1368 | 249.1101 | 83.2573 | 54.8628 |
| $F_4$    | ISSA      | 5.6861E-13 | 1.8820E-06 | 1.4071E-07 | 3.8628E-07 |
|          | SSA       | 1.8713E-09 | 3.8206E-06 | 8.7093E-07 | 9.5760E-07 |
|          | GWO       | 6.1621E-07 | 1.5093 | 0.5886 | 0.3408 |
|          | WOA       | 0.0858 | 1.6502 | 0.4335 | 0.3084 |
|          | PSO       | 4.9346E-06 | 0.0005 | 0.0001 | 0.0001 |
| $F_5$    | ISSA      | 1.1559E-05 | 0.0004 | 0.0001 | 0.0001 |
|          | SSA       | 4.9654E-05 | 0.0008 | 0.0003 | 0.0002 |
|          | GWO       | 0.0003 | 0.0047 | 0.0020 | 0.0010 |
|          | WOA       | 0.0124 | 8.2148E-05 | 0.0031 | 0.0032 |
|          | PSO       | 0.0749 | 0.3896 | 0.1792 | 0.0613 |
| Function | Algorithm | Best | Worse | Ave | Std |
|----------|-----------|------|-------|-----|-----|
| $F_6$    | ISSA      | -39166.9401 | -26683.1355 | -33826.5691 | 3441.9490 |
|          | SSA       | -12569.2938 | -4921.5749 | -8846.3963 | 2448.0671 |
|          | GWO       | -7303.4504  | -3531.3164 | -6117.2140 | **762.5930** |
|          | WOA       | -12569.4866 | -6925.8467 | -10793.3021 | 1926.5821 |
|          | PSO       | -7476.3798  | -3034.9355 | -5307.5740 | 1108.6601 |
|          |           | **8.8818E-16** | **8.8818E-16** | **8.8818E-16** | 0 |
| $F_7$    | ISSA      | 8.8818E-16  | 8.8818E-16 | 8.8818E-16 | 0 |
|          | SSA       | 8.8818E-16  | 8.8818E-16 | 8.8818E-16 | 0 |
|          | GWO       | 9.6811E-14  | 2.6024E-13 | 1.6100E-13 | 3.8009E-14 |
|          | WOA       | 8.8818E-16  | 7.9936E-15 | 4.4409E-15 | 2.4270E-15 |
|          | PSO       | 0.0025      | 1.1610     | 0.1537     | 0.3541   |
| $F_8$    | ISSA      | 3.3155E-13  | 1.6293E-07 | 1.6517E-08 | 3.1809E-08 |
|          | SSA       | 1.6624E-09  | 1.5204E-06 | 1.3821E-07 | 3.0227E-07 |
|          | GWO       | 0.0066      | 0.0975     | 0.0364     | 0.0203   |
|          | WOA       | 0.0051      | 0.0618     | 0.0239     | 0.0128   |
|          | PSO       | 1.0695E-07  | 0.2073     | 0.0138     | 0.0443   |

Fig. 6. Convergence effect diagram of each algorithm
Fig. 6. Continued

Fig. 7. Three-Dimensional diagram of test function
Table 5. Wilcoxon rank sum test P value

| Function | ISSA-SSA | ISSA- WOA | ISSA- GWO | ISSA-PSO |
|----------|----------|-----------|-----------|----------|
| F1       | N/A      | 1.5299E-11| 1.5299E-11| 1.5299E-11|
| F2       | N/A      | 1.2117E-12| 1.2117E-12| 1.2117E-12|
| F3       | 6.0459E-01| 3.019E-11 | 3.019E-11 | 3.019E-11 |
| F4       | 3.2323E-06| 6.6591E-12| 9.3030E-12| 2.0707E-12|
| F5       | 1.0385E-04| 9.1288E-10| 4.5311E-11| 2.0701E-11|
| F6       | 2.0701E-11| 3.019E-11 | 2.0701E-11| 2.0701E-11|
| F7       | N/A      | 3.6292E-09| 1.1863E-12| 1.2117E-12|

Fig. 7. Continued
It can be seen from Table 4 that the ISSA algorithm is significantly better than the other four comparison algorithms in eight test functions. For F1 and F2, ISSA algorithm and SSA algorithm can find the theoretical optimal value, but the average value of ISSA algorithm is smaller, which shows that the optimization effect of ISSA algorithm is superior. For F3, F4, F8, the optimal value of ISSA algorithm is higher than the other four algorithms by several orders of magnitude or even a dozen orders of magnitude, and the average and standard deviation are the smallest. For F6, the average value of ISSA algorithm is the smallest, indicating that the convergence rate of ISSA algorithm is faster. However, in this function, the standard deviation of GWO algorithm is the smallest, indicating that the stability of GWO algorithm is better than that of ISSA algorithm in function F6. For F7, the optimal results of ISSA algorithm and SSA algorithm are the same, and the average value and standard deviation are the same, indicating that the two algorithms have the same effect on function F7. In the seven groups of test functions except F6, the average and standard deviation of ISSA algorithm are the minimum values, indicating that ISSA algorithm has better stability and stronger robustness, and has stronger global optimization ability than the other four algorithms.

As shown in Fig. 6, the convergence curves of eight groups of test functions are plotted. The abscissa and ordinate represent the number of iterations and the fitness function value respectively. It can be seen from Fig. 6 that among the convergence curves of eight groups of test functions, ISSA algorithm is at the bottom, indicating that ISSA algorithm has faster convergence speed and higher solution accuracy than the other four algorithms. In the convergence curve of F7, the ISSA algorithm and SSA algorithm almost coincide after 50 iterations, indicating that the convergence accuracy of the two algorithms is the same for the test function, but the ISSA algorithm converges faster in the early iteration. Integrating the convergence curves of eight test functions, the convergence curves of ISSA algorithm finally tend to be stable, indicating that the algorithm has the fastest convergence speed and the highest convergence accuracy.

In order to more comprehensively reflect the performance of ISSA algorithm, this paper, Wilcoxon rank sum test is used to verify whether ISSA algorithm has significant difference with other algorithms at \( P = 5 \) percent level. As shown in Table 5, \( P < 5 \) percent indicates that the difference between the two algorithms is obvious, \( P > 5 \) percent shows that the two algorithms have little difference, that is, the algorithm performance is close. N/A indicates that the performance of the two algorithms is close.

### 4.3 Software defect prediction contrast experiment

In order to verify the superiority of ISSA-ELM-Bagging algorithm, four prediction algorithms, ELM, SSA-ELM, ISSA-ELM and SSA-ELM-Bagging, are used as comparison algorithms. In order to ensure the accuracy of the experiment, the ISSA-ELM-Bagging algorithm uses the average of the six indicators of precision (P), recall (R), F-measure, MCC, Accuracy and G-mean as the evaluation index to verify the prediction performance of the algorithm. The higher the six evaluation indexes are, the better the software defect prediction effect is. The optimal results of each evaluation index have been bold. The evaluation index calculation formula is as follows \([39]\):

\[
P = \frac{TP}{(TP + FP)}
\]

\[
R = \frac{TP}{(TP + FN)}
\]
\[ F - measure = \frac{2 \times P \times R}{P + R} \]  
(13)  

\[ MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \]  
(14)  

\[ Accuracy = \frac{TP + FN}{TP + FP + TN + FN} \]  
(15)  

\( TP \) is the number of samples correctly classified by minority samples; \( FP \) is the number of correctly classified majority class samples; \( FN \) is the number of misclassified minority samples; \( TN \) is the number of samples misclassified by most class samples\(^\text{(40)}\).

\[ G - mean = \sqrt{TPR \times TNR} \]  
(16)  

\( TPR = \frac{TP}{TP + FN} \), \( TNR = \frac{TN}{FP + TN} \). \( TPR \) is the proportion of predicted correct samples in the actual minority samples; \( TNR \) is the proportion of predicted correct samples in most class samples to actual majority class samples\(^\text{(41)}\).

| Dataset   | ELM   | SSA-ELM | ISSA-ELM | SSA-ELM-Bagging | ISSA-ELM-Bagging |
|-----------|-------|---------|----------|-----------------|-----------------|
| CM1       | 0.111 | 0.2727  | 0.3977   | 0.6054          | 0.6249          |
| MC2       | 0.2774| 0.6353  | 0.4736   | 0.9000          | 1.0000          |
| MW1       | 0.1818| 0.3333  | 0.3749   | 0.5714          | 0.9999          |
| PC1       | 0.0909| 0.2962  | 0.3333   | 0.4292          | 0.5714          |
| PC3       | 0.1791| 0.1702  | 0.2045   | 0.3777          | 0.3962          |
| PC2       | 0.0541| 0.0909  | 0.1667   | 0.1999          | 0.2500          |
| PC4       | 0.3139| 0.0478  | 0.4999   | 0.4571          | 0.5416          |
| ar1       | 0.1250| 0.5000  | 0.9999   | 0.5000          | 0.9999          |
| ar4       | 0.3333| 0.7999  | 0.7500   | 0.5000          | 0.8333          |
| ar6       | 0.2857| 1.0000  | 0.7500   | 0.7500          | 1.0000          |
| EQ        | 0.5882| 0.8064  | 0.8125   | 0.8928          | 0.9545          |
| JDT       | 0.5172| 0.6249  | 0.6551   | 0.6862          | 0.7297          |
| ML        | 0.2574| 0.3296  | 0.3703   | 0.6249          | 0.6888          |
| PDE       | 0.2986| 0.3571  | 0.3714   | \textbf{0.4492} | 0.4489          |
| ProjectA  | 0.5641| 0.7000  | 0.6764   | 0.6176          | 0.6571          |
| ProjectK  | 0.4924| 0.5423  | 0.5193   | 0.5362          | 0.5942          |
| ant-1.3   | 0.5714| 0.6667  | 0.5000   | 1.0000          | 1.0000          |
| arc       | 0.3333| 0.5000  | \textbf{0.5833} | 0.5000          | 0.5000          |
| camel-1.0 | 0.1111| 0.1999  | 0.2857   | 0.6666          | 0.9999          |
| peis-1.5  | 0.6956| 0.8333  | 0.8695   | \textbf{0.8888} | 0.8695          |
| rediktor  | 0.3333| 0.4444  | 0.5000   | 0.6666          | 0.8000          |
| velocity-1.4 | 0.2727 | 0.6666 | 0.9999 | 0.9166 | \textbf{1.0000} |
| xerces-1.2| 0.1851| 0.3636  | 0.4500   | 0.4782          | 0.6842          |
| apache    | 0.6470| 0.7368  | 0.8333   | 0.8571          | 0.9500          |
| zxing     | 0.2692| 0.5217  | 0.5000   | 0.5416          | 0.7500          |
| sum       | 8.0897| 12.8007 | 13.8783  | 15.6123         | 18.8449         |
| Mean value| 0.3236| 0.5120  | 0.5551   | 0.6245          | 0.7538          |
| Standard deviation | 0.1822 | 0.2277 | 0.2273 | 0.1944 | 0.2194 |
Fig. 8. Standard deviation of Precision for each algorithms

### Table 7. Prediction Recall of algorithms

| Dataset   | ELM        | SSA-ELM    | ISSA-ELM | SSA-ELM-Bagging | ISSA-ELM-Bagging |
|-----------|------------|------------|----------|-----------------|------------------|
| CM1       | 0.3333     | 0.5000     | 0.7778   | 0.8333          | 1.0000           |
| MC2       | 0.4545     | 0.7272     | 0.8181   | 1.0000          | 1.0000           |
| MW1       | 0.3333     | 0.5000     | 0.5000   | 0.5000          | 0.5714           |
| PC1       | 0.2500     | 0.6666     | 0.8333   | 0.8081          | 0.3000           |
| PC3       | 0.7500     | 0.5000     | 0.5625   | 0.7083          | 0.8750           |
| PC2       | 0.6666     | 0.6666     | 0.7500   | 1.0000          | 1.0000           |
| PC4       | 0.6923     | 0.7948     | 0.7692   | 0.8205          | 0.7647           |
| ar1       | 0.3333     | 0.3333     | 0.3333   | 1.0000          | 0.6666           |
| ar4       | 0.5000     | 0.6666     | 0.7500   | 1.0000          | 1.0000           |
| ar6       | 0.5000     | 0.5000     | 0.7500   | 0.7500          | 0.6000           |
| EQ        | 0.6250     | 0.7812     | 0.812    | 0.8620          | 0.9545           |
| JDT       | 0.6383     | 0.7446     | 0.8085   | 0.7608          | 0.7941           |
| ML        | 0.6190     | 0.5882     | 0.7142   | 0.7894          | 0.7948           |
| PDE       | 0.5000     | 0.5434     | 0.5652   | 0.6326          | 0.6470           |
| ProjectA  | 0.8800     | 0.8400     | 0.9200   | 0.9544          | 0.9583           |
| ProjectK  | 0.9154     | 0.9014     | 0.9436   | 0.9610          | 0.9647           |
| ant-1.3   | 0.5714     | 0.8571     | 1.0000   | 1.0000          | 1.0000           |
| arc       | 0.5000     | 0.6666     | 0.5833   | 0.6666          | 0.7500           |
| camel-1.0 | 0.3333     | 0.5000     | 0.6666   | 0.6666          | 0.6666           |
| poi-1.5   | 0.7272     | 0.9090     | 0.9090   | 0.8888          | 0.9523           |
| reldktor  | 0.5000     | 0.6666     | 0.6666   | 0.6666          | 0.8000           |
| velocity-1.4 | 0.4285 | 0.5714     | 0.6666   | 0.8461          | 0.8181           |
| xerces-1.2 | 0.3333 | 0.5333     | 0.6000   | 0.7857          | 0.8666           |
| apache    | 0.5238     | 0.8750     | 0.8333   | 0.9000          | 0.9047           |
| zxing     | 0.3888     | 0.5454     | 0.6666   | 0.6842          | 0.8181           |
| Sum       | 13.2980    | 16.3793    | 18.2010  | 20.4859         | 20.968           |
| Mean value| 0.5319     | 0.6552     | 0.7280   | 0.8194          | 0.8387           |
| Standard deviation | 0.1727 | 0.1513     | 0.1488   | 0.1361          | 0.1324           |
Fig. 9. Standard deviation of Recall for each algorithms

Table 8. Prediction F-measure of algorithms

| Dataset   | ELM    | SSA-ELM | ISSA-ELM | SSA-ELM-Bagging | ISSA-ELM-Bagging |
|-----------|--------|---------|----------|-----------------|-----------------|
| CM1       | 0.1666 | 0.3529  | 0.5263   | 0.7000          | 0.7692          |
| MC2       | 0.3445 | 0.6782  | 0.6000   | 0.9473          | 1.0000          |
| MW1       | 0.2352 | 0.4000  | 0.4285   | 0.5333          | 0.7272          |
| PC1       | 0.1333 | 0.4102  | 0.4761   | 0.5606          | 0.6666          |
| PC3       | 0.2891 | 0.2539  | 0.3000   | 0.4927          | 0.5454          |
| PC2       | 0.1000 | 0.1600  | 0.2727   | 0.3333          | 0.4000          |
| PC4       | 0.4320 | 0.5391  | 0.6060   | 0.5871          | 0.6341          |
| ar1       | 0.1818 | 0.4000  | 0.5000   | 0.6666          | 0.8000          |
| ar4       | 0.4000 | 0.7272  | 0.7500   | 0.6666          | 0.9091          |
| ar6       | 0.3636 | 0.6666  | 0.7500   | 0.7500          | 0.7500          |
| EQ        | 0.6060 | 0.7936  | 0.8125   | 0.8772          | 0.9546          |
| JDT       | 0.5714 | 0.6796  | 0.7238   | 0.7216          | 0.7605          |
| ML        | 0.3636 | 0.4225  | 0.4878   | 0.6976          | 0.7381          |
| PDE       | 0.3739 | 0.4310  | 0.4482   | 0.5254          | 0.5301          |
| ProjectA  | 0.6875 | 0.7636  | 0.7796   | 0.7500          | 0.7796          |
| ProjectK  | 0.6403 | 0.6772  | 0.6700   | 0.6883          | 0.7354          |
| ant-1.3   | 0.5714 | 0.7500  | 0.6666   | 1.0000          | 1.0000          |
| arc       | 0.4000 | 0.5714  | 0.5833   | 0.5714          | 0.6000          |
| camel-1.0 | 0.1666 | 0.2857  | 0.4000   | 0.6666          | 0.8000          |
| poi-1.5   | 0.7111 | 0.8695  | 0.8888   | 0.8888          | 0.9091          |
| redaktor  | 0.4000 | 0.5333  | 0.5714   | 0.6666          | 0.8000          |
| velocity-1.4 | 0.3333 | 0.6153  | 0.8000   | 0.8800          | 0.9000          |
| xerces-1.2 | 0.2381 | 0.4324  | 0.5142   | 0.5945          | 0.7647          |
| apache    | 0.5789 | 0.8000  | 0.8333   | 0.8780          | 0.9268          |
| zxing     | 0.3181 | 0.5333  | 0.5714   | 0.6046          | 0.7826          |
| Sum       | 9.6073 | 13.7474 | 14.9613  | 17.2491         | 19.1835         |
| Mean value| 0.3843 | 0.5499  | 0.5985   | 0.6900          | 0.7673          |
| Standard deviation | 0.1751 | 0.1862  | 0.1633   | 0.1542          | 0.1456          |
Fig. 10. Standard deviation of F-measure for each algorithms

Table 9. Prediction MCC of algorithms

| Dataset   | ELM   | SSA-ELM | ISSA-ELM | SSA-ELM-Bagging | ISSA-ELM-Bagging |
|-----------|-------|---------|----------|-----------------|-----------------|
| CM1       | 0.0509| 0.2871  | 0.5080   | 0.6542          | 0.7715          |
| MC2       | 0.1134| 0.8314  | 0.8461   | 0.9185          | 1.0000          |
| MW1       | 0.1108| 0.3142  | 0.3483   | 0.4581          | 0.7324          |
| PC1       | 0.0182| 0.3718  | 0.4654   | 0.5456          | 0.6486          |
| PC3       | 0.2705| 0.1951  | 0.2533   | 0.4366          | 0.5188          |
| PC2       | 0.1659| 0.2286  | 0.3376   | 0.4348          | 0.4949          |
| PC4       | 0.3334| 0.4717  | 0.5423   | 0.5285          | 0.5842          |
| ar1       | 0.0000| 0.3418  | 0.5516   | 0.6915          | 1.0000          |
| ar6       | 0.2300| 0.6420  | 0.6911   | 0.6915          | 0.8856          |
| EQ        | 0.2009| 0.6000  | 0.6306   | 0.7817          | 0.9312          |
| JDT       | 0.4244| 0.5728  | 0.6333   | 0.6772          | 0.7099          |
| ML        | 0.2787| 0.3190  | 0.4292   | 0.6231          | 0.6712          |
| PDE       | 0.2364| 0.3115  | 0.3334   | 0.4227          | 0.4678          |
| ProjectA  | 0.6799| 0.7490  | 0.7722   | 0.7514          | 0.7778          |
| ProjectK  | 0.6017| 0.6379  | 0.6379   | 0.6543          | 0.6977          |
| ant-1.3   | 0.4047| 0.6458  | 0.6756   | 0.7514          | 1.0000          |
| arc       | 0.1409| 0.4031  | 0.4404   | 0.5053          | 0.5687          |
| camel-1.0 | 0.1274| 0.2844  | 0.3985   | 0.6512          | 0.8101          |
| pois-1.5  | 0.4568| 0.7526  | 0.7916   | 0.8222          | 0.8324          |
| redaktor  | 0.2527| 0.4262  | 0.4745   | 0.6354          | 0.7677          |
| velocity-1.4 | 0.1584| 0.5435 | 0.7785 | 0.8249          | 0.8738          |
| xerces-1.2 | 0.02606| 0.2966 | 0.4031 | 0.5190 | 0.7169 |
| apache    | 0.1914| 0.6471  | 0.6904   | 0.7364          | 0.8469          |
| zxing     | 0.0735| 0.3478  | 0.4311   | 0.4679          | 0.6964          |
| Sum       | 5.7194| 11.8637 | 13.7574  | 16.1330         | 18.7357         |
| Mean value| 0.2288| 0.4745  | 0.5503   | 0.6453          | 0.7494          |
| Standard deviation | 0.1712| 0.1813 | 0.1631 | 0.1516 | 0.1496 |
Fig. 11. Standard deviation of MCC for each algorithms

Table 10. Prediction Accuracy of algorithms

| Dataset | ELM | SSA-ELM | ISSA-ELM | SSA-ELM-Bagging | ISSA-ELM-Bagging |
|---------|-----|---------|----------|-----------------|------------------|
| CM1     | 0.7101 | 0.8406 | 0.8696 | 0.8986 | 0.9265 |
| MC2     | 0.6800 | 0.8000 | 0.8400 | 0.9600 | 1.0000 |
| MW1     | 0.7547 | 0.8302 | 0.8491 | 0.8679 | 0.9434 |
| PC1     | 0.7347 | 0.8435 | 0.8503 | 0.8966 | 0.9456 |
| PC2     | 0.7306 | 0.7854 | 0.8082 | 0.8356 | 0.8364 |
| PC3     | 0.8792 | 0.9295 | 0.9463 | 0.9463 | 0.9799 |
| PC4     | 0.7428 | 0.8080 | 0.8582 | 0.8370 | 0.8913 |
| ar1     | 0.6250 | 0.8750 | 0.9167 | 0.9583 | 0.9583 |
| ar4     | 0.7143 | 0.8636 | 0.9048 | 0.9583 | 0.9546 |
| ar6     | 0.6667 | 0.8500 | 0.9048 | 0.9167 | 0.9048 |
| EQ      | 0.6000 | 0.8000 | 0.8154 | 0.8615 | 0.9692 |
| JDT     | 0.7739 | 0.8341 | 0.8550 | 0.8650 | 0.9150 |
| ML      | 0.7554 | 0.7802 | 0.8306 | 0.8700 | 0.8794 |
| PDE     | 0.7425 | 0.7793 | 0.7860 | 0.8100 | 0.8700 |
| ProjectA | 0.9476 | 0.9660 | 0.9660 | 0.9634 | 0.9660 |
| ProjectK | 0.8549 | 0.8787 | 0.8688 | 0.8608 | 0.8748 |
| ant-1.3 | 0.7600 | 0.8400 | 0.9200 | 1.0000 | 1.0000 |
| arc     | 0.6170 | 0.7447 | 0.7872 | 0.8723 | 0.9149 |
| camel-1.0 | 0.8529 | 0.9265 | 0.9118 | 0.9706 | 0.9851 |
| poi-1.5 | 0.6875 | 0.8333 | 0.8541 | 0.9166 | 0.8936 |
| redtlor | 0.7428 | 0.8000 | 0.8285 | 0.9428 | 0.9444 |
| velocity-1.4 | 0.6500 | 0.8500 | 0.8974 | 0.8974 | 0.9487 |
| xerces-1.2 | 0.6363 | 0.7613 | 0.8068 | 0.8295 | 0.9091 |
| apache | 0.5897 | 0.8205 | 0.8461 | 0.8684 | 0.9231 |
| zxing   | 0.6250 | 0.7342 | 0.7750 | 0.7875 | 0.8750 |
| Sum     | 18.0737 | 18.0737 | 18.0737 | 18.0737 | 18.0737 |
| Mean value | 0.7229 | 0.8310 | 0.8599 | 0.8957 | 0.9284 |
| Standard deviation | 0.0893 | 0.0551 | 0.0408 | 0.0549 | 0.0428 |
Fig. 12. Standard deviation of Accuracy for each algorithms

Table 11. Prediction G-mean of algorithms

| Dataset | ELM     | SSA-ELM | ISSA-ELM | SSA-ELM-Bagging | ISSA-ELM-Bagging |
|---------|---------|---------|----------|-----------------|-----------------|
| CM1     | 0.4986  | 0.6606  | 0.8529   | 0.8520          | 0.9759          |
| MC2     | 0.6241  | 0.8217  | 0.8017   | 0.9682          | 1.0000          |
| MW1     | 0.5191  | 0.6604  | 0.6684   | 0.6831          | 0.7559          |
| PC1     | 0.4409  | 0.7568  | 0.842    | 0.8481          | 0.8746          |
| PC3     | 0.7394  | 0.6355  | 0.6822   | 0.7788          | 0.8556          |
| PC2     | 0.7665  | 0.7883  | 0.843    | 0.9724          | 0.9898          |
| PC4     | 0.7210  | 0.8024  | 0.8194   | 0.8300          | 0.8337          |
| ar1     | 0.4714  | 0.5634  | 0.5773   | **0.9780**      | 0.8165          |
| ar4     | 0.6183  | 0.7905  | 0.8401   | **0.9780**      | 0.9701          |
| ar6     | 0.5940  | 0.7071  | 0.8401   | **0.8441**      | 0.7746          |
| EQ      | 0.5998  | 0.7995  | 0.8153   | 0.8889          | 0.9655          |
| JDT     | 0.7216  | 0.8011  | 0.8383   | 0.8257          | 0.8638          |
| ML      | 0.6916  | 0.6905  | 0.7771   | 0.8377          | 0.8516          |
| PDE     | 0.6271  | 0.6684  | 0.6833   | 0.7327          | 0.7624          |
| ProjectA| 0.9154  | 0.9048  | 0.9442   | 0.9592          | 0.9624          |
| ProjectK| 0.8794  | 0.8881  | 0.8990   | 0.9036          | 0.9140          |
| ant-1.3 | 0.6901  | 0.8451  | 0.9555   | **1.0000**      | 1.0000          |
| arc     | 0.5732  | 0.7171  | 0.7071   | 0.7756          | 0.8352          |
| camel-1.0| 0.5406  | 0.6853  | 0.7844   | 0.8102          | 0.8165          |
| poi-1.5 | 0.7290  | 0.8770  | 0.8967   | 0.9108          | 0.9178          |
| redaktor| 0.6297  | 0.7427  | 0.7581   | 0.8036          | 0.8798          |
| velocity-1.4| 0.5698  | 0.7326  | 0.8165   | 0.9020          | 0.9045          |
| xerces-1.2| 0.4825  | 0.6565  | 0.7138   | 0.8113          | 0.8918          |
| apache  | 0.5909  | 0.8275  | 0.8451   | 0.8660          | 0.9243          |
| zxing   | 0.5193  | 0.6634  | 0.7332   | 0.7488          | 0.8564          |
| Sum     | 15.7544 | 18.6875 | 19.9369  | 21.5096         | 22.1938         |
| Mean value | 0.6302  | 0.7475  | 0.7975   | 0.8604          | 0.8878          |
| Standard deviation | 0.1186  | 0.0867  | 0.0884   | 0.0833          | 0.0723          |
By analyzing Table 6 to Table 11, ISSA-ELM-Bagging algorithm is compared with ELM, SSA-ELM, ISSA-ELM and SSA-ELM-Bagging for twenty-five datasets. Among the six evaluation indexes of Precision, Recall, F-measure, MCC, Accuracy and G-mean, ISSA-ELM-Bagging algorithm can get the optimal value for most datasets. The mean value of ISSA-ELM-Bagging algorithm is the maximum among the six evaluation indexes, which fully shows the efficiency of ISSA-ELM-Bagging algorithm. It can be concluded from Table 6 that for PDE and poi-1.5 datasets, the precision value of SSA-ELM-Bagging prediction algorithm is the best. About arc datasets, the precision value of ISSA-ELM-Bagging algorithm is the highest. In these three datasets, the precision value of ISSA-ELM-Bagging algorithm is the suboptimal value. In Table 7, the Recall values of ISSA-ELM-Bagging algorithm and SSA-ELM-Bagging algorithm for MC2, PC2 and ar4 datasets reach the optimal value. For Ant-1.3 and camel-1.0 datasets, the three prediction algorithms of ISSA, SSA-ELM-Bagging and ISSA-ELM-Bagging can reach the optimal value. About PC4, ar1, ar6 and velocity-1.4, the data volume is generally small and the number of features is few. The experimental results show that the Recall value of SSA-ELM-Bagging prediction algorithm is better, and ISSA-ELM-Bagging algorithm is suboptimal. The Recall value of ISSA-ELM is superior to the other four algorithms on PC1 dataset, but the difference between ISSA-ELM-Bagging algorithm and the optimal value is only 0.03. For Table 8, in the dataset ant-1.3, ProjectA, ar6, SSA-ELM-Bagging algorithm and SSA-ELM-Bagging prediction algorithm, ISSA-ELM prediction algorithm, ISSA-ELM and SSA-ELM-Bagging prediction algorithm respectively reach the optimal F-measure value. In the rest of datasets, the F-measure values of ISSA-ELM-Bagging algorithm are the best. For the datasets with low defect rate and large amount of data, the prediction effect of ISSA-ELM-Bagging algorithm is good. It can be seen from Table 9 that the ISSA-ELM-Bagging algorithm and SSA-ELM-Bagging prediction algorithm are superior to the other three comparative prediction algorithms on ant-1.3. ISSA-ELM-Bagging algorithm has high prediction efficiency in EQ, JDT, ML and PDE datasets. Table 10 shows that SSA-ELM-Bagging prediction algorithm has the highest accuracy value in the ar4, ar6, poi-1.5 three datasets, ISSA-ELM-Bagging algorithm is a suboptimal value. For the dataset poi-1.5, SSA-ELM-Bagging algorithm may not reflect the effect of low defect rate datasets because of the high defect rate of data. It can be seen from Table 11 that ISSA-ELM-Bagging algorithm and SSA-ELM-Bagging prediction algorithm both reach the optimal G-mean value in the dataset ant-1.3. The prediction effect of SSA-ELM-Bagging algorithm is better than that of the other four prediction algorithms for ar1, ar4 and ar6 datasets. ISSA-ELM-Bagging algorithm is a suboptimal prediction algorithm on ar1 and ar4.
datasets. For ar6 dataset, the difference between the proposed algorithm and the optimal G-mean value is 0.1095. For this dataset, the number of data features and the number of samples are relatively few, resulting in the poor effect of ISSA-ELM-Bagging algorithm.

The ISSA-ELM-Bagging algorithm can achieve the optimal evaluation index value in most datasets. By analyzing Table 6 to Table 11, it can be seen that the ISSA-ELM-Bagging algorithm can achieve good prediction results for high data dimension, large data volume and low data defect rate, which shows that the ISSA-ELM-Bagging algorithm has higher prediction accuracy and stronger generalization ability.

Standard deviation is an important indicator to measure the effect and stability of the algorithm. In the experimental process, the lower the standard deviation is, the better the stability of the algorithm is, and vice versa. It can be seen from Fig. 8 that the standard deviation of ISSA-ELM-Bagging algorithm is not the optimal value. Since the accuracy of ELM prediction algorithm for 25 datasets is generally low, the final standard deviation is lower than that of ISSA-ELM-Bagging algorithm. It can be seen from Fig. 9 to Fig. 13 that the standard deviation of ISSA-ELM-Bagging algorithm is the minimum compared with the standard deviation of the other four prediction algorithms on five evaluation indexes. Especially under the four evaluation indexes of F-measure, MCC, Accuracy and G-mean, the standard deviation of ISSA-ELM-Bagging algorithm is significantly better than that of the other four prediction algorithms. It shows that ISSA-ELM-Bagging algorithm has better overall stability on the software defect dataset than the other four prediction algorithms.

The traditional ELM is a single hidden layer feedforward neural network with poor stability and random selection of parameters, resulting in low prediction accuracy of the model. The traditional sparrow search algorithm has excellent optimization ability, which can obtain the optimal parameters of ELM and improve the stability, but it is easy to fall into the local optimal solution in the late iteration. The improved sparrow search algorithm has faster convergence rate and the ability to jump out of the local optimal solution by using the reverse learning of pinhole imaging and the foraging strategy of flip bucket. Using improved sparrow search algorithm to optimize ELM as the base predictor of ensemble learning algorithm can further improve the prediction accuracy and generalization ability of ELM.

5 Conclusions
In this paper, ISSA-ELM-Bagging algorithm is proposed to improve the stability of ELM in datasets with low defect rate and high data dimension. The optimization ability of sparrow search algorithm is improved by using small hole imaging reverse learning and flip bucket foraging strategy. Then, the ISSA algorithm is used to optimize the random selection of ELM parameters, and the optimal parameters of ELM are obtained to ensure the prediction accuracy of ISSA-ELM. Finally, ISSA-ELM is used as the base predictor of Bagging ensemble learning algorithm to improve the stability of ISSA-ELM. The experimental results show that the optimization performance of ISSA algorithm is significantly better than other prediction algorithms. For 25 software defect datasets, the six evaluation indexes of ISSA-ELM-Bagging ensemble prediction algorithm are significantly superior to other prediction algorithms. In addition, the weighted integration strategy can be explored to further improve the performance and stability of the model.

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.

Acknowledgment This work is supported by the National Natural Science Foundation of China under Grant No. 52074126. We would like to thank the editor and anonymous reviewers for their valuable comments and suggestions to improve the paper.

Author Statement Yu Tang: Writing-original draft, Editing and Visualization. Qi Dai: Methodology and Review. Mengyuan Yang: Data curation, Visualization. Lifang Chen: Review, Conceptualization.
References

[1] Xu Z, Li L, Yan M (2021) A comprehensive comparative study of clustering-based unsupervised defect prediction models. Journal of Systems and Software 172(3): 110862.

[2] Jayanthi R, Florence L (2019) Software defect prediction techniques using metrics based on neural network classifier. Cluster Computing.

[3] Limsettho N, Bennin KE, Keung JW (2018) Cross Project Defect Prediction Using Class Distribution Estimation and Oversampling. Information and Software Technology 100: 87-102.

[4] Wang J, Zhang C (2018) Software reliability prediction using a deep learning model based on the RNN encoderdecoder. Reliability Engineering & System Safety. 170(feb.): 73-82.

[5] Chen X, Zhang D, Zhao Y, et al (2018) Software defect number prediction: Unsupervised vs supervised methods. Information and Software Technology. 106.

[6] Jin C (2020) Software defect prediction model based on distance metric learning. Soft Computing (3).

[7] Kz A, Shi YA, Nz A (2021) Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network. Journal of Systems and Software 180.

[8] Hall T, Beecham S, Bowes D (2012) A Systematic Literature Review on Fault Prediction Performance in Software Engineering. Software Engineering, IEEE Transactions on. 38(6): p.1276-1304.

[9] Yadav HB, Yadav DK (2015) A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Information & Software Technology. 63(july) 44-57.

[10] Wang J, Zhang C (2018) Software reliability prediction using a deep learning model based on the RNN encoderdecoder. Reliability Engineering & System Safety. 170(feb.): 73-82.

[11] Laradji IH, Alshayeb M, Ghouti L (2015) Software defect prediction using ensemble learning on selected features. Information & Software Technology. 58(feb.): 388-402.

[12] Zhou X, Jin L, XI B (2019) Software defect prediction based on kernel PCA and weighted extreme learning machine - ScienceDirect. Information and Software Technology. 106: 182-200.

[13] Menzies T, Greenwald J, Frank A (2007) Data Mining Static Code Attributes to Learn Defect Predictors. IEEE Transactions on Software Engineering. 33: p.2-13.

[14] Peng H, Jie Z (2010) Predicting Defect-Prone Software Modules at Different Logical Levels. IEEE.

[15] Breiman, Leo (1996) Bagging Predictors. Machine Learning. 24(2): 123-140.

[16] Duffy N, Helmbold D (2002) Boosting Methods for Regression. Machine Learning. 4 7(2/3): 153-200.

[17] Sun Z, Song Q, Zhu X (2012) Using coding-based ensemble learning to improve software defect prediction. IEEE Trans Syst Man Cybern. 42 , pp. 1806-1817

[18] Wang S, Yao X (2013) Using Class Imbalance Learning for Software Defect Prediction. IEEE Transactions on Reliability. 62(2): 434-443.

[19] Liu BY, Chen GL, Lin HC (2021) Prediction of IGBT junction temperature using improved cuckoo search-based extreme learning machine. Microelectronics Reliability 124.

[20] Ding L, Zhang XY, Wu DY (2021) Application of an extreme learning machine netw
ork with particle swarm optimization in syndrome classification of primary liver cancer. Journal of Integrative Medicine.

[21] Li LL, Sun J, Tseng ML (2019) Extreme learning machine optimized by whale optimization algorithm using Insulated Gate Bipolar Transistor module aging degree evaluation. Expert Systems with Applications 127.

[22] Zhang QQ, Li Z, Lu Z (2021) Real-time prediction of river chloride concentration using ensemble learning. Environmental Pollution.

[23] Kumar V, Minz S (2015) Multi-view ensemble learning: an optimal feature set partitioning for high-dimensional data classification. Knowledge & Information Systems 2015(1): 1-59.

[24] Han Zhezhe, Jian Li, Md. Moinul Hossain (2021) An ensemble deep learning model for exhaust emissions prediction of heavy oil-fired boiler combustion, Fuel.

[25] Xue JT, Shen B (2020) A novel swarm intelligence optimization approach: sparrow search algorithm. Syst. Sci. Control Eng. 8 (1), pp. 22-34.

[26] Xu J, Tan W, Li T (2020) Predicting fan blade icing by using particle swarm optimization and support vector machine algorithm. Computers & Electrical Engineering. 87(1): 106751.

[27] Keshavdas GP, Shah VA, Lokhande MM (2018) Grey wolf algorithm for multidimensional engine optimization of converted plug-in hybrid electric vehicle. Transportation Research Part D Transport & Environment. 63: 632-648.

[28] Gaganpreet, Kaur, Sankalap (2018) Chaotic whale optimization algorithm. Journal of Computational Design & Engineering.

[29] Zhang C, Ding S (2021) A stochastic configuration network based on chaotic sparrow search algorithm. Knowledge-Based Systems. 220(10): 106924.

[30] Wen L, J JJ, L XM (2021) Pinhole-imaging-based learning butterfly optimization algorithm for global optimization and feature selection. Applied Soft Computing.

[31] Zhao WG, Zhang ZX, Wang LY (2020) Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Engineering Applications of Artificial Intelligence 87(C): 103300-103300.

[32] Zhang DM, Xu H, Wang YR (2021) Whale optimization algorithm embedded Circle mapping and dimension-by-dimension hole imaging reverse learning. Control and decision-making 36(05): 1173-1180.

[33] Liu BY, Chen GL, Lin HC (2021) Prediction of IGBT junction temperature using improved cuckoo search-based extreme learning machine. Microelectronics Reliability 124.

[34] Tong R, Li P (2021) A novel adaptive weighted kernel extreme learning machine algorithm and its application in wind turbine blade icing fault detection. Measurement (12): 110009.

[35] Ding L, Zhang XY, Wu DY (2021) Application of an extreme learning machine network with particle swarm optimization in syndrome classification of primary liver cancer. Journal of Integrative Medicine.

[36] Zza B, Gw A, Chong LB (2021) Bagging-based positive-unlabeled learning algorithm with Bayesian hyperparameter optimization for three-dimensional mineral potential mapping. Computers & Geosciences, 154.

[37] Ouyang CT (2021) Lens Learning Sparrow Search Algorithm. Mathematical Problems in Engineering 2021.

[38] Zhu Y, Yousefi N (2021) Optimal parameter identification of PEMFC stacks using Adaptive Sparrow Search Algorithm. International Journal of Hydrogen Energy. 46(14).
The author’s introduction

Yu Tang, born in 1998. He is currently working toward the M.S. degree in cyberspace security with the college of Science, North China University of Science and Technology. His research interest includes machine learning, software defect prediction and sparrow search algorithm.
Email address: hblg_ty@163.com (Yu Tang)

Qi Dai, born in 1991. He received the M.S. degree in mathematics from North China University of Science and Technology in 2020. He is currently working toward the Ph.D. degree in control theory and control engineering with the department of automation, college of information science and engineering, China University of petroleum, Beijing campus (CUP). His research interests include data mining and machine learning.
Email address: daiq_cup@126.com (Qi Dai)

Mengyuan Yang, born in 2000. He is currently working toward the B.S. degree in computer science with the college of Science, North China University of Science and Technology. His research interest includes artificial intelligence, machine learning and deep learning.
Email address: ymy306340869@163.com (Mengyuan Yang)

Li-fang Chen, born in 1973. She received the Ph.D. degree in China University of Mining & Technology. Her research interest includes data mining and processing, neural network modeling, control theory.
Email address: chenlifang@ncst.edu.cn (Lifang Chen)
Postal address: No. 21 Bohai Avenue, Caofeidian New City, North China University of Science and Technology, Tangshan Hebei, 063210.
Tel.: +86 132 3082 9011