IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 30, 2021, accepted April 7, 2021, date of publication April 13, 2021, date of current version April 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072993

Software Defects Prediction Based on Hybrid Particle Swarm Optimization and Sparrow Search Algorithm

LIU YANG[1,2], **ZHEN LI**[1,2], **DONGSHENG WANG**[1,3], **HONG MIAO**[2,3], **AND ZHAOBIN WANG**[1,2]

[1]School of Electronics and Information, Jiangsu University of Science and Technology, Zhenjiang 212003, China
[2]Reliability and Systems Engineering Open Group, Jiangsu University of Science and Technology, Zhenjiang 212003, China
[3]School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang 212003, China

Corresponding author: Zhen Li (justlz@just.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702234 and Grant 51977101, and in part by the Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant SJCX20_1466.

ABSTRACT Software defects reflect software quality, and software failures can be predicted through software reliability models. Aiming at the problem that the parameters of software reliability model are difficult to estimate, this paper used the hybrid algorithm for model parameter estimation to software defect prediction. As a typical swarm intelligence algorithm, PSO (Particle Swarm Optimization) has fast convergence but low solution accuracy. SSA (Sparrow Search Algorithm) not only has high search accuracy and fast convergence speed, but also has the advantages of good stability and strong robustness. Based on the characteristic that the fitness function proposed in this paper, this paper hybrid PSO and SSA to accelerate the convergence before the individual update of the SSA. At the same time, this paper also constructed a new fitness function based on the maximum likelihood estimation of the parameters, and used it for parameter initialization. Through the analysis of the experimental results of five sets of actual data sets, the optimization performance of the hybrid algorithm (SSA-PSO) was better than that of a single algorithm with higher convergence speed and more stable, accurate results. Moreover, with the support of the new fitness function, it effectively solved the problems of slow convergence speed and low accuracy of solution. The experimental results showed that the hybrid SSA-PSO could obtain the better solution, convergence speed and stability than single SSA and PSO in software defections estimation and prediction.

INDEX TERMS Defects prediction, software reliability, parameter estimation, swarm intelligence, sparrow search algorithm, particle swarm optimization.

I. INTRODUCTION

Software defect prediction based on software reliability model is an essential issue in determining software quality. The problem consists of two major tasks. One is to estimate parameters of proper software reliability model that have the best fitness to software failure data, the other is to predicate the occurrence time of software defects.

The non-homogeneous Poisson process model is the mainstream software reliability growth model (SRGM) describing software reliability, and the Goel-Okumoto model (G-O) is a classic software reliability growth model derived from it [1], [2]. In the G-O model, the software fault function and fault detection rate are fixed values, which simplifies the SRGM parameters and solves the problem that too many parameters are not conducive to parameter optimization. G-O model has a reasonable description of the software failure process and is in line with the actual situation, so it has become a model often used by testers [3].

When estimating model parameters, in addition to numerical optimization and evolutionary algorithm methods, swarm intelligence algorithms are also used to estimate these model parameters [4], [5]. The swarm intelligence optimization is a new type of biological heuristic calculation method that is inspired by observing various behaviors of social biological groups, such as ants, birds, fish, etc. Based on those swarm intelligence optimization, some new [6]–[8] and hybrid algorithm [9], [10] comes out to get a better solution.

Sparrow Search Algorithm (SSA) is a new type of swarm intelligence algorithm proposed by Jiankai [11] in 2020.

The associate editor coordinating the review of this manuscript and approving it for publication was Jian Guo.

It uses foraging behavior and anti-predation behavior in sparrow populations to iteratively optimization. It has the characteristics of high search accuracy, fast convergence speed, good stability, strong robustness and strong global search ability, and provides a brand new method for solving complex global optimization problems. Li Yali and others had studied and compared the SSA with other new swarm intelligence optimizations. From the comprehensive comparison of experimental results, it was found that the performance of the sparrow search algorithm was much better than traditional optimization algorithms. It was a swarm intelligence algorithm with good optimization performance [12]. In order to improve the global search ability of SSA in further, Lu Xin *et al.* proposed a Chaos Sparrow Optimization Algorithm (CSSA). The algorithm firstly used the Tent chaotic map to initialize the population, so that the initial individuals evenly distributed as possible and introduced Gaussian mutation with chaotic disturbance. When there was ''clustering'' or ''divergence'' in the population, it adjusted individuals to help them jump out of the local optimum [13]. Not only that, they later combined the idea of Bird Swarm Algorithm (BSA) to proposed an Improved Sparrow Search Algorithm (ISSA), which changed the position update formula of the SSA to the update formula of the BSA. In order to shorten the running time of the algorithm, it improved its search ability and development ability [14].

PSO (Particle Swarm Optimization) was proposed by Eberhart and Kennedy in 1995, which refers to the foraging behavior of birds. The advantage of the PSO is fewer control parameters in its model, and it is easier to realize. In the early stage of finding solution, PSO has a faster convergence speed. However, it is easy to fall into local optimum during the search process, resulting in low accuracy of the solution.

In order to study the pyrolysis of a typical agricultural residue, Li Xu *et al.* used PSO to estimate the parameters of the pyrolysis kinetic model [15]. Yanming Ding *et al.* used Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) to estimate the reaction kinetic parameters of biomass pyrolysis. The results showed that the accuracy and efficiency of PSO were higher [16]. Laxmi A. Bewoor *et al.* proposed hybridization of PSO with simulated annealing for planning and scheduling issues [17]. In the literature [18], Helbert Eduardo Espitia *et al.* presented the statistical analysis of vortex particle swarm optimization (VPSO) which was a boost algorithm based on self-propelled particle swarms. In the literature [19], İbrahim Berkan Aydilek proposed a hybrid algorithm combining firefly and particle swarm optimization (HFPSO). The proposed algorithm was able to exploit the strongpoints of both particle swarm and firefly algorithm mechanisms.

This paper used a hybrid algorithm to estimate the parameters of software reliability model and predicted the number and occurence time of software failures. The improvement of the algorithm and the hybrid strategy in the literatures mentioned can prevent the algorithm from falling into the local optimum to a certain extent, but there are still defects such as insufficient search accuracy of the algorithm, slow convergence speed and weak development ability. Although the particle swarm algorithm converges quickly, its solution accuracy is low, and the sparrow algorithm not only has high search accuracy and fast convergence speed, but also has the advantages of good stability and strong robustness. Mixing the two algorithms can improve the solution accuracy and convergence speed of the particle swarm algorithm, and at the same time improve the stability of the algorithm.

In order to further accelerate the algorithm convergence and ensure the stability of the problem solution, this paper proposes a new fitness function for the initialization of parameter b. At the same time, as a new type of swarm intelligence algorithm, this paper is the first to propose a mixture of sparrow algorithm and particle swarm algorithm, which provides new ideas for the improvement of sparrow algorithm. The hybrid algorithm ((SSA-PSO) firstly used the PSO to perform the initial search of the population, so that the initial individuals converge more quickly, and then used the SSA to perform global rapid convergence. The Section II introduced the basic theories related to this paper: software reliability and its models, the basic principles of PSO and SSA, the construction of fitness function and the realization of hybrid algorithms. Section III conduct an experimental simulation on data sets, and compared the results of different algorithms to verify the feasibility and effectiveness of the SSA-PSO. Finally, discussion was in Section IV and there came to conclusion in Section V.

II. MATERIALS AND METHODS

A. SOFTWARE RELIABILITY AND MODEL

IEEE defines software reliability as the probability that software will not fail within a specified time under specified conditions. The software reliability model refers to the reliability block diagram and mathematical model established to predict or estimate the reliability of the software [20], [21].

Modeling software reliability is a mathematical method, and the choice of model parameters will directly affect the accuracy of software reliability and defects prediction. In this paper, a representative model called G-O in the software reliability model was selected, and its parameters will be estimated. The estimated function of the cumulative failure number in the software system was as follows:

$$m(t) = a(1 - e^{bt}) \tag{1}$$

where: m (t) represents the expected function of the cumulative number of failures until time t; a represents the total number of failures the software expects to be detected after the end of the test; b represents the probability that the remaining failures are found, and is a proportional constant with a range of (0, 1).

The problem to be solved in this paper is nonlinear, and there will be multiple local optimal solutions. It is worth noting that in terms of the algorithm structure, the swarm intelligence algorithm designs multiple agents to search the solution space and search in different local solution spaces, which can well solve the problems of multiple local solutions and global solutions. So it is necessary to use this meta-heuristic method to solve the problem.

B. PARTICLE SWARM OPTIMIZATION

PSO realized the optimal solution search based on the mutual cooperation and information sharing among particles in the group. First, the initial population was randomly generated, and the fitness value of each particle was determined by the objective function. The fitness value represented the quality of the particle position. In each iteration of the search for optimal solution, each particle will adjust its position following two extreme values. One was the optimal fitness value found so far by the particle itself, which was called the individual best value. The other was the optimal fitness value found so far by all other particles in the population, called the global best value.

The mathematical expression of the particle swarm optimization is: in a d-dimensional search space, there is a population of m particles, that is, $X = \{X_1, \ldots X_i, \ldots X_m\}$. The position of the i-th particle is expressed as $X_i = \{X_{i1}, X_{i2}, \ldots X_{id}\}^T$, and its velocity is expressed as $V_i = \{V_{i1}, V_{i2}, \ldots V_{id}\}^T$. The individual extreme value of the i-th particle is expressed as $P_{bi} = \{P_{bi1}, P_{bi2}, \ldots P_{bid}\}^T$. The global extremum of the population is expressed as $g_b = \{g_{b1}, g_{b2}, \ldots g_{bd}\}^T$. The particles update their velocity and position according to formulas (2) and (3):

$$v_{id}^{(t+1)} = wv_{id}^t + c_1 rand_1()\left(P_{bid}^t - x_{id}^t\right) + c_2 rand_2()\left(g_{bd}^t - x_{id}^t\right) \quad (2)$$

$$x_{id}^{(t+1)} = x_{id}^t + v_{id}^{(t+1)} \quad (3)$$

In the formula, w is the weight of inertia, which is a non-negative number between [0, 1], indicating the ability of the particle to inherit the current velocity; learning factors c_1, c_2, generally $c_1 = c_1 = 2$, indicating the ability of particle learning; $rand_1()$ and $rand_2()$ are random numbers between (0, 1). The update speed of particle is composed of three parts. The first part is the previous speed of the particle, which represents the current state of the particle is inertial moved by its own speed, balancing the global exploration and local development capabilities. The second part is the "cognition" part. P_{bid} represents the optimal position that the particle has experienced so far. The difference between it and the current position of the particle represents the impact of the particle's own experience on its next behavior. The third part is the "society" part. g_{bd} is the optimal position found by all particles so far. The cooperation and information sharing between particles make the particles search in a better direction. Because particles are random during flight, particles may fly out of the solution space. For particle speed, generally the value of v_{max} is between the range of particle position width, if x_{max} is limited to the range of $[-x_{max}, x_{max}]$, then $v_{max} = k \times x_{max}, 0.1 \le k \le 1.0$. If the updated velocity is less than $-v_{max}$ or greater than v_{max}, the current position velocity is not updated, and $-v_{max}$ and v_{max} are used instead.

C. SPARROW SEAR ALGORITHM

In the process of foraging for food, sparrows are divided into explorers and followers. Explorers usually have higher energy reserves and are responsible for finding food in the population and providing foraging areas and directions for the entire sparrow population, while followers use explorers to obtain food. In the model, the level of energy reserve depends on the fitness value of the individual sparrow. During each iteration, the position update formula of the explorer is as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^t \cdot \exp\left(-\dfrac{i}{\alpha \cdot iter_{max}}\right), & \text{if } R_2 < ST \\ X_{i,j}^t + Q \cdot L, & \text{if } R_2 \ge ST \end{cases} \quad (4)$$

In the formula, t represents the current iteration number, and $iter_{max}$ is a constant, which represents the maximum iteration number. $X_{i,j}$ represents the position information of the i-th sparrow in the j-th dimension. $\alpha (\alpha \in (0, 1])$ is a random number. R_2 $(R_2 \in [0, 1])$ and ST $(ST \in [0.5, 1])$ represents the warning value and the safety value respectively. Q is a random number that obeys a normal distribution. L represents a matrix of $1 \times d$, where each element in the matrix is all 1. When $R_2 < ST$, it means that there are no predators around the foraging environment at this time, and the explorer can perform a wide range of search operations. When $R_2 \ge \acute{Y}ST$, it means that some sparrows in the population have found a predator and have issued an alarm to other sparrows in the population. At this time, all sparrows need to fly to other safe places quickly for food.

At the same time, in order to increase their predation rate, some followers may constantly monitor the explorers and compete for food resources. The lower the energy of the followers, the worse their foraging position in the entire population. The follower's location update is described as follows:

$$X_{i,j}^{t+1} = \begin{cases} Q \cdot \exp\left(\dfrac{X_{worst} - X_{i,j}^t}{i^2}\right), & \text{if } i > n/2 \\ X_P^{t+1} + \left|X_{i,j}^t - X_P^{t+1}\right| \cdot A^+ \cdot L, & \text{otherwise} \end{cases} \quad (5)$$

In the above formula, X_P is the best position currently occupied by explorers, and X_{worst} represents the worst position currently. A represents a matrix of $1 \times d$, where each element is randomly assigned a value of 1 or −1, and $A^+ = A^T\left(AA^T\right)^{-1}$. When $i > n/2$, this indicates that the i-th follower with a lower fitness value has no food and is very hungry. At this time, it needs to fly to other places for food to get more energy.

The identities of explorers and followers change dynamically. As long as they can find a better source of food, every sparrow can become an explorer, but the proportion of explorers and followers in the entire population remains unchanged. In other words, if one sparrow becomes an explorer, another sparrow becomes a follower. In the process of foraging, followers can always search for the explorer, who provides the best food, and then obtain food from the best food or forage around the explorer.

Suppose that 10% to 20% of the sparrows aware of danger are randomly generated in the population. When they are aware of the danger, the sparrow population will make anti-predation behaviors, and the sparrows at the edge of the group will quickly move to a safe area to obtain a better position. The sparrows in the middle of the population will walk around randomly to get close to other sparrows. The mathematical expression is as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{best}^{t} + \beta \cdot \left|X_{i,j}^{t} - X_{best}^{t}\right|, & \text{if } f_i > f_g \\ X_{i,j}^{t} + K \cdot \left(\dfrac{\left|X_{i,j}^{t} - X_{worst}^{t}\right|}{(f_i - f_w) + \varepsilon}\right), & \text{if } f_i = f_g \end{cases} \tag{6}$$

In the formula, X_{best} is the current global optimal position. β as the step control parameter, is a random number that obeys a normal distribution with a mean of 0 and a variance of 1. $K(K \in [-1, 1])$ is a random number, and f_i is the fitness value of the current individual sparrow. f_g and f_w are the current global best and worst fitness values, respectively. ε is the smallest constant to avoid zeros in the denominator. When $f_i > f_g$ means that the sparrow is at the edge of the population at this time and is extremely vulnerable to predators. When $f_i = f_g$, this indicates that the sparrows in the middle of the population are aware of the danger and need to be close to other sparrows to minimize their risk of predation. K represents the direction in which the sparrow moves and is also a step length control parameter.

D. CONSTRUCTION OF FITNESS FUNCTION

The key to using intelligent optimization algorithms to estimate the software reliability model is to construct a suitable fitness function, that is, the objective optimization function, and turn the parameter estimation problem into a function optimization problem. Aiming at the characteristics of the software reliability model, referring to the principle of the least square method, the fitness function constructed is as follows:

$$f_{it} = \frac{\sqrt{\sum_{i=1}^{n} \left[m(t_i) - m^0(t_i)\right]^2}}{n} \tag{7}$$

where: f_{it} represents the relative difference between the actual number of software failures and the number of failures estimated by the model. The smaller the value of f_{it}, the higher the accuracy of model fitting, that is, the better the result of parameter estimation. $m(t_i)$ represents the cumulative number of failures actually found in the test period $(0, t_i]$. $m^0(t_i)$ represents the cumulative number of failures estimated by the model in the test period $(0, t_i]$. t_i is the moment when the i-th failure occurs. $i = 1, 2, 3, \ldots n$. n represents the total number of failures that occurred at the end of the test.

Before using formula (7) to control the parameter optimization, in order to speed up the algorithm convergence speed, this paper constructed the fitness function of b according to the maximum likelihood estimation formula of the software reliability G-O model parameter to initialize the parameter b. The maximum likelihood estimation formulas of a and b were as follows:

$$\begin{cases} a = \dfrac{n}{1 - e^{-bt_n}} \\ \dfrac{n}{b} = at_n e^{-bt_n} + \sum\limits_{i=1}^{n} t_i \end{cases} \tag{8}$$

Substituted the expression of the first equation a into the second equation and perform mathematical transformation to construct an equation only related to the b, in which all parameters except b are known, as shown below:

$$f = \left| b - \frac{n\left(1 - e^{-bt_n}\right)}{nt_n e^{-bt_n} + \left(1 - e^{-bt_n}\right)\sum\limits_{i=1}^{n} t_i} \right| \tag{9}$$

f can be used as the fitness function for the initialization of the control parameter b. The method proposed in this paper was to first use f as the fitness function, and performed an iterative search of parameter b through the swarm intelligence algorithm. When the algorithm end condition was met, then f_{it} was used as the fitness function to optimize the algorithm. The algorithm was used to iteratively search the two fitness functions, and when the algorithm stop criterion was reached, the optimal parameter b was obtained.

E. IMPLEMENTATION OF SSA-PSO

PSO determines the direction of finding new solutions through all the solutions found by the current known populations, that is, new solutions are generated depends on all the solutions found by these populations. Because the entire search and update process follows the current optimal solution process, all particles may converge to the optimal solution faster. Although the convergence speed of the particles is very fast, this one-way information flow will lead to the lack of randomness in the change of the particle position. Therefore, the PSO has the defect that it is easy to fall into local optimization and cause premature convergence in the late iteration.

The unimodal test function has only one maximum value in the search interval, which can reflect the ability of the algorithm to develop and converge locally. The SSA has superior performance when dealing with unimodal test functions. Its optimal solution and average solution are higher than other algorithms, and the convergence speed is sharp. At the same time, when dealing with multimodal test functions, its convergence speed is much higher than other algorithms, so the SSA has higher performance global search capabilities than other algorithms, and it can find more areas meaningful in solution space.

The fitness function fit mentioned above is a unimodal function. Therefore, in order to improve the convergence speed of the algorithm, this section proposed an algorithm combining PSO and SSA, namely SSA-PSO. The algorithm used the SSA to continue searching around the new location after the PSO completes the location update to determine the result. The algorithm flow was shown in Figure 1:

Input:

✧ the actual number of software failures n-10 and the time t_i of each failure;

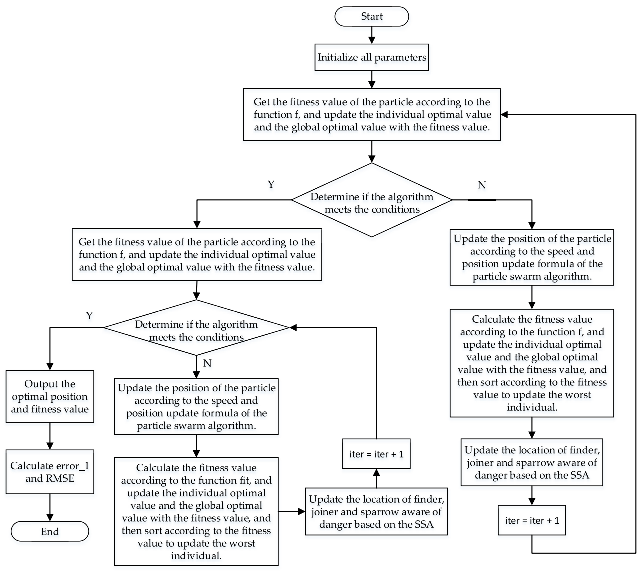
✧ the parameters of the hybrid algorithm.

Output:

✧ the optimal parameter b and the corresponding fitness value;

✧ the error after prediction according to the parameter b: error_1 and RMSE, where error_1 is the difference of the predicted value and the actual value of the n-9th failure, and RMSE is the root mean square error between the predicted value and the actual value of the last 10 failures(n-9 to n).

This paper will constrain the parameters a, b and the y value of the fitness function f. a represents the total number of failures that the software expects to be detected after the test, so the value of parameter a should be greater than the known number of failures. b represents the probability that the remaining failure is found, so its range is (0,1). The accuracy of the y value of the fitness function f cannot be less than 1e-4.



Start
Initialize all parameters
Get the fitness value of the particle according to the function f, and update the individual optimal value and the global optimal value with the fitness value.
Determine if the algorithm meets the conditions
Y
N
Get the fitness value of the particle according to the function f, and update the individual optimal value and the global optimal value with the fitness value.
Update the position of the particle according to the speed and position update formula of the particle swarm algorithm.
Determine if the algorithm meets the conditions
Calculate the fitness value according to the function f, and update the individual optimal value and the global optimal value with the fitness value, and then sort according to the fitness value to update the worst individual.
Output the optimal position and fitness value
Update the position of the particle according to the speed and position update formula of the particle swarm algorithm.
Update the location of finder, joiner and sparrow aware of danger based on the SSA
iter = iter + 1
Calculate error_1 and RMSE
Calculate the fitness value according to the function fit, and update the individual optimal value and the global optimal value with the fitness value, and then sort according to the fitness value to update the worst individual.
Update the location of finder, joiner and sparrow aware of danger based on the SSA
iter = iter + 1
End

FIGURE 1. Implementation of the SSA-PSO algorithm.

1) Initialize all parameters. The number of ppapers: pop=10; the maximum number of iterations: M=10; the proportion of explorers is 20%; the proportion of sparrows aware of danger is 20%; the safety threshold is 0.8; the learning factor: $c_1 = c_2 = 1.5$; the inertia weight: w =0.9; the position of each ppaper, the parameter b of the G-O model, is initialized to a random number between (0, 1),and the speed is initialized to a random number between [−1, 1].
2) Substituting b into the fitness function (9) to obtain the fitness value of each ppaper, and update the individual optimal value and the global optimal value.
3) Determine whether the algorithm stop condition is met, that is, the number of iterations is less than or equal to M and the global optimal fitness value accuracy is greater than or equal to 1e-4, if it is satisfied, go to step (10), otherwise go to step (4).
4) Update the speed and position of each ppaper according to formulas (2) and (3).
5) Substitute b into the fitness function (Equation 9) to obtain the fitness value of each ppaper, update the individual and global optimal values, sort by fitness value, and update the worst individual.
6) Update the position of the explorer according to formula (4).
7) Update the position of followers according to formula (5).
8) Update the position of the sparrow that is aware of danger according to formula (6).
9) iter=iter+1, go to step (2).
10) Substitute b into the fitness function (Equation 7) to obtain the fitness value of each ppaper, and update the individual optimal and global optimal.
11) Determine whether the algorithm stop condition is met, that is, the number of iterations is less than or equal to M, if it is satisfied, go to step (18), otherwise go to step (12).
12) Update the speed and position of each ppaper according to formulas (2) and (3).
13) Substitute b into the fitness function (Equation 7) to obtain the fitness value of each ppaper, update the individual and global optimal values, sort by fitness value, and update the worst individual.
14) Update the position of the explorer according to formula (4).
15) Update the position of followers according to formula (5).
16) Update the position of the sparrow that is aware of danger according to formula (6).
17) iter=iter+1, go to step (10).
18) The algorithm satisfies the end condition and outputs the optimal fitness value y_min and the optimal position b_best.
19) Substitute the estimated parameter b_best into the function expression of the G-O model for prediction, calculate the difference (error_1) between the predicted value of the n-9th failure and the actual value, and the root mean square error (RMSE) between the predicted and actual values of the last 10 failures.

In order to show the steps of the hybrid algorithm more clearly, the algorithm architecture is expressed in interpretable code as follows:

```
Start
Initialize all parameters；
for i=1: pop
    Substitute b into the fitness function (9) to get the fitness value of each ppaper;
end for
Update individual optimal value and global optimal value;
iter=1;
while (iter≤ M &&y_min>=1e-4)
    for i=1: pop
        Update the speed and position of each ppaper by using formulas (2) and (3);
        Substitute b into the fitness function (9) to get the fitness value of each ppaper;
    end for
    Update individual optimal value and global optimal value;
    Sort by fitness value, update the worst individual value;
    Update the position of the explorer according to formula (4)；
    Update the position of followers according to formula (5);
    Follow formula (6) to update the location of sparrows aware of danger；
    for i=1: pop
        Substitute b into the fitness function (9) to get the fitness value of each ppaper;
    end for
    Update individual optimal value and global optimal value;
    iter=iter+1；
end while
for i=1: pop
    Substitute b into the fitness function (7) to get the fitness value of each ppaper;
end for
Update individual optimal value and global optimal value;
iter=1;
while (iter≤ M)
    for i=1: pop
        Update the speed and position of each ppaper by using formulas (2) and (3);
        Substitute b into the fitness function (7) to get the fitness value of each ppaper;
    end for
    Update individual optimal value and global optimal value;
    Sort by fitness value, update the worst individual value;
    Update the position of the explorer according to formula (4)；
    Update the position of followers according to formula (5);
    Follow formula (6) to update the location of sparrows aware of danger；
    for i=1: pop
        Substitute b into the fitness function (7) to get the fitness value of each ppaper;
    end for
    Update individual optimal value and global optimal value;
    iter=iter+1；
end while
Substitute the estimated parameter b_best into the function expression of the G-O model to predict the number of failures, and calculate error_1 and RMSE.
```

III. RESULTS

In this paper, five sets of software defects data (SYS1, SS3, CSR1, CSR2 and CSR3) obtained in an actual industrial project were used. The address of data downloaded is http://www.cse.cuhk.edu.hk/lyu/book/reliability/data.html.

Assuming that the actual number of failures in each data set is n, the algorithm used in this paper was to estimate parameters based on a total of n-10 data, and predict the last 10 data based on the estimated results, and then calculated the prediction performance.

A. EFFECTIVENESS OF THE FITNESS FUNCTION

In the Section II, it was proposed to add the fitness function of b to initialize the parameter b, which can improve the convergence speed of the algorithm. In order to verify its effectiveness, in this section, we took PSO as an example to compare the experimental results of two methods that not use f(f-0) to control parameter b and use f(f-1) to control parameter b.

The parameters of the ppaper swarm algorithm were set as follows:

the number of ppapers: pop=10;
the maximum number of iterations: M=10/20/40/60/80;
the learning factor: $c_1 = c_2 = 1.5$;
the inertia weight: w =0.9;
the position of each ppaper, the parameter b of the G-O model, is initialized to a random number between (0, 1), and the speed is initialized to a random number between [−1, 1].

Each algorithm runs 20 times, and the best and worst results and the average value of 20 times of y value are shown in Table 1∼5:

In Tables 1 to 5:

✧ M is the maximum number of iterations in the algorithm.

TABLE 1. **The best and worst values of different iterations (SYS1).**

The number of iterations(M)	f(0/1)	Fitness value(y)			Difference (error_1)		Root mean square error (RMSE)	
		the worst	the best	average	the worst	the best	the worst	the best
10	f-0	6.10	1.41	4.74	1.00	0.98	6.20	5.71
	f-1	5.85	0.64	2.04	1.00	0.90	6.20	3.48
20	f-0	5.54	0.80	3.88	1.00	0.93	6.20	4.46
	f-1	5.00	0.61	1.92	1.00	0.88	6.20	3.03
40	f-0	5.08	0.65	2.46	1.00	0.90	6.20	3.59
	f-1	4.25	0.61	1.57	1.00	0.87	6.20	2.85
60	f-0	4.27	0.70	2.30	1.00	0.91	6.20	3.98
	f-1	1.31	0.62	0.86	0.97	0.88	5.60	3.16
80	f-0	2.81	0.61	1.13	1.00	0.88	6.20	2.90
	f-1	1.19	0.61	0.70	0.97	0.87	5.43	2.78

TABLE 2. **The best and worst values of different iterations (SS3).**

The number of iterations(M)	f(0/1)	Fitness value(y)			Difference (error_1)		Root mean square error (RMSE)	
		the worst	the best	average	the worst	the best	the worst	the best
10	f-0	9.04	4.13	8.40	1.00	0.94	6.21	5.54
	f-1	8.93	1.30	5.63	1.00	0.65	6.21	2.12
20	f-0	8.96	1.35	6.76	1.00	0.66	6.21	2.21
	f-1	7.19	0.90	3.39	1.00	0.57	6.20	1.40
40	f-0	8.67	0.69	5.72	1.00	0.50	6.21	1.23
	f-1	6.46	0.66	3.15	1.00	0.49	6.19	1.23
60	f-0	7.70	0.65	4.30	1.00	0.40	6.21	1.26
	f-1	5.11	0.61	1.96	0.98	0.39	5.98	1.26
80	f-0	6.11	0.65	2.36	1.00	0.45	6.17	1.47
	f-1	4.62	0.62	1.60	0.96	0.42	5.80	1.22

TABLE 3. **The best and worst values of different iterations (CSR1).**

The number of iterations(M)	f(0/1)	Fitness value(y)			Difference (error_1)		Root mean square error (RMSE)	
		the worst	the best	average	the worst	the best	the worst	the best
10	f-0	11.2	2.81	9.64	1.00	1.00	6.21	6.20
	f-1	10.9	2.15	6.23	1.00	0.99	6.21	6.17
20	f-0	11.0	1.84	7.82	1.00	0.98	6.21	6.14
	f-1	9.44	1.45	5.13	1.00	0.96	6.21	6.06
40	f-0	8.64	1.41	5.09	1.00	0.95	6.21	6.04
	f-1	5.90	1.20	2.65	1.00	0.90	6.21	5.84
60	f-0	8.53	1.24	3.94	1.00	0.92	6.21	5.92
	f-1	3.87	1.18	1.66	1.00	0.88	6.20	5.78
80	f-0	6.36	1.21	2.29	1.00	0.83	6.21	5.58
	f-1	3.13	1.18	1.46	1.00	0.87	6.20	4.24

TABLE 4. **The best and worst values of different iterations (CSR2).**

The number of iterations(M)	f(0/1)	Fitness value(y)			Difference (error_1)		Root mean square error (RMSE)	
		the worst	the best	average	the worst	the best	the worst	the best
10	f-0	6.25	2.81	5.60	1.00	1.00	6.21	6.20
	f-1	6.15	1.38	4.51	1.00	0.89	6.21	6.03
20	f-0	6.22	1.97	4.99	1.00	0.98	6.21	6.17
	f-1	5.60	0.80	3.65	1.00	0.48	6.21	5.25
40	f-0	6.07	1.08	3.34	1.00	0.77	6.21	5.81
	f-1	4.01	0.75	1.76	1.00	0.30	6.21	4.89
60	f-0	5.38	0.80	2.60	1.00	0.46	6.21	5.23
	f-1	1.52	0.74	0.93	0.92	0.11	6.09	4.52
80	f-0	3.08	0.74	1.40	1.00	0.08	6.20	4.45
	f-1	1.73	0.74	0.90	0.96	0.07	6.14	4.43

✧ f-0 means that the function f is not used to control the parameter b.

✧ f-1 means using the function f to control the parameter b.

✧ y value is the optimal fitness value output by the algorithm, which is the minimum value calculated by the fitness function fit. Therefore, the smaller the y value, the better the fit to the data set, the better the effect of parameter estimation, and the better the accuracy of prediction.

✧ error_1 and RMSE were described in part E of Section II. They can represent the error between the predicted value and the true value to a certain extent, and were positively correlated with the y value, so their values should be as small as possible. In the table, error_1 and RMSE had similar expressions to the y value, so in the following we will focus on the change of the y value.

Observing the data changes of the y value, it can be clearly found that regardless of whether the function f was used to control the parameter b for initialization or not, as the number of iterations increases, the optimal, worst, and average values of y began to gradually decrease. This was because the more iterations, the algorithm naturally converges finally.

But when the number of iterations is the same, it can be found that the result of the algorithm using f was smaller, and as the number of iterations increases, its convergence speed was also faster.

And in most cases, the y value of f-1 is not only smaller than the current y value of f-0, but also smaller than the value of f-0 after increasing the number of iterations. This showed

TABLE 5. The best and worst values of different iterations (CSR3).

The number of iterations(M)	f(0/1)	Fitness value(y)			Difference (error_1)		Root mean square error (RMSE)	
		the worst	the best	average	the worst	the best	the worst	the best
10	f-0	5.10	0.79	3.15	1.00	0.95	6.21	5.46
	f-1	3.54	0.57	1.69	1.00	0.88	6.21	4.38
20	f-0	4.29	0.57	2.49	1.00	0.89	6.21	4.65
	f-1	3.03	0.57	1.34	1.00	0.89	6.21	4.61
40	f-0	3.38	0.58	1.30	1.00	0.90	6.21	4.76
	f-1	2.92	0.58	0.89	1.00	0.89	6.21	4.67
60	f-0	1.50	0.57	0.86	0.99	0.89	6.09	4.55
	f-1	1.00	0.57	0.62	0.97	0.88	5.77	4.48
80	f-0	1.22	0.57	0.72	0.98	0.89	5.95	4.60
	f-1	0.73	0.57	0.59	0.94	0.88	5.35	4.47

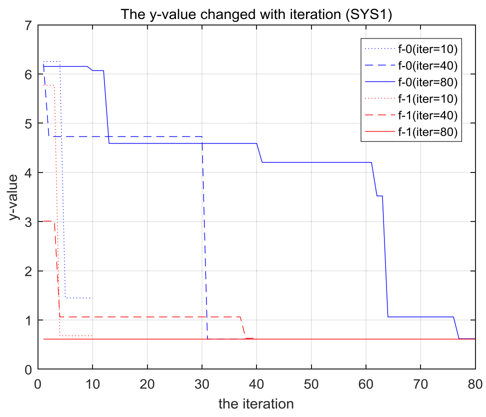


FIGURE 2. The y-value changed with iteration (SYS1).

that the use of the f function greatly improved the accuracy and convergence speed of the algorithm, and verified the effectiveness of the function f.

1) ANALYSIS OF CONVERGENCE

In order to show the specific situation of the convergence of the y value in the algorithm search, and further analyze the influence of the y value on the number of iterations and the function f, the iterative process with the optimal y value in Tables 1 to 5 is recorded and analyzed.

This section showed the iterations of the algorithm when the maximum number of iterations is 10, 40, and 80, respectively. For example, when the number of iterations is 10, take the result of 10 changes in the y value when the algorithm ends and reaches the optimal value. At the same time, the convergence results of adding function f(f-1) and not adding function f(f-0) are also compared, as shown in Figures 2∼6:

It can be seen from Figures 2∼6 that at the beginning of convergence, the y value of f-1 is already much smaller than f-0. This showed that using the function f to control the parameter b can indeed optimize the initialization of b and

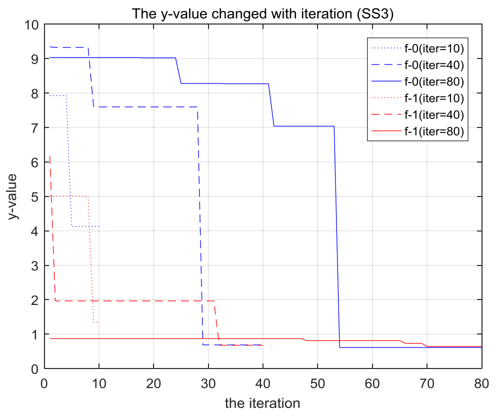


FIGURE 3. The y-value changed with iteration (SS3).

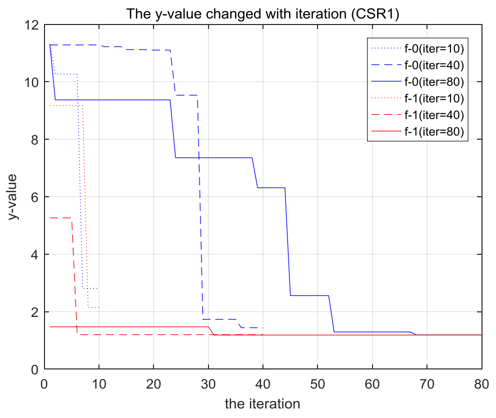


FIGURE 4. The y-value changed with iteration (CSR1).

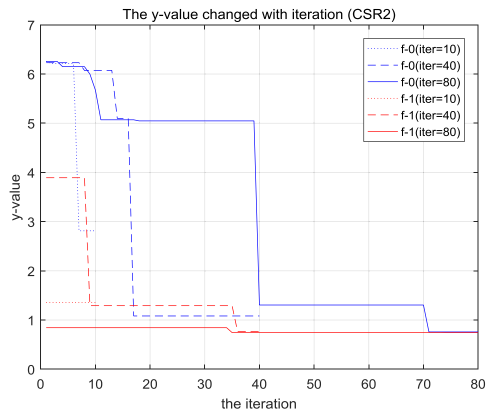


FIGURE 5. The y-value changed with iteration (CSR2).

bring the solution space more closer to the optimal solution. Function f speed up the convergence of the y value and improves the accuracy and stability of the result.

However, the y value of f-0 was the same at the initial stage, and it can only converge to the optimal value by increasing the number of iterations. When the maximum number of iterations is 80, the y value of f-1 changes very little, and some do not even continue to converge. This was because the solution space was already at the optimal solution. This

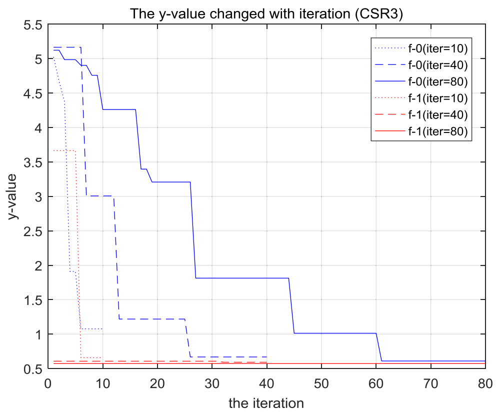


FIGURE 6. The y-value changed with iteration (CSR3).

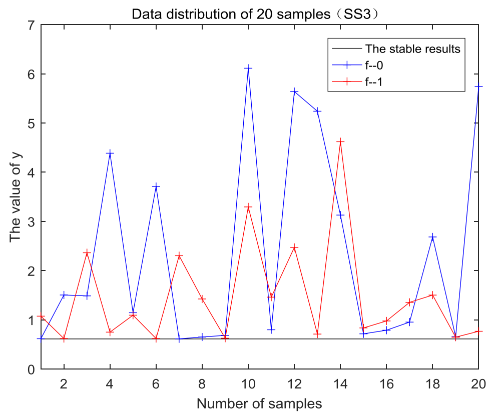


FIGURE 8. Data distribution-y of 20 samples (SS3).

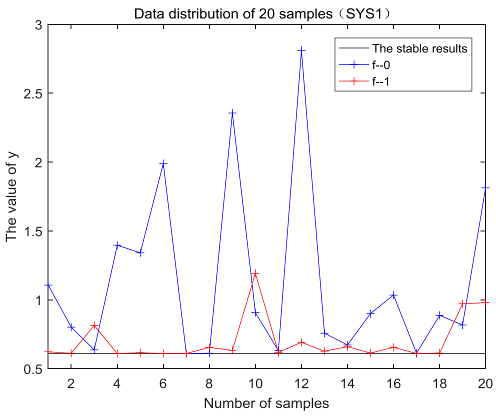


FIGURE 7. Data distribution-y of 20 samples (SYS1).

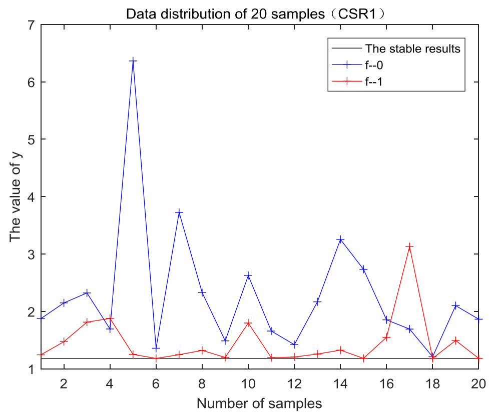


FIGURE 9. Data distribution-y of 20 samples (CSR1).

further showed that the use of the function f greatly improve the convergence of the algorithm results.

2) ANALYSIS OF STABILITY

The previous section analyzed the convergence of the y value, and speculated that the use of the function f can improve the accuracy and stability of the results. In order to make a stable comparison intuitively and conveniently, this section compared the y-value results of the algorithm running 20 times when the maximum number of iterations is 80, and also divided the two cases of whether to use the function f. The statistical distributions of each data set were shown in Figures 7∼11:

In Figures 7∼11, the horizontal black line was at the bottom, and its corresponding y value represented the optimal value that the algorithm can achieve. The gap between it and other lines represented the difference between the y value calculated by the algorithm and the optimal value. Therefore, the smaller the distance change, the more stable the y value obtained by the algorithm. Observing the five figures, it was obvious that the red line using the function f was much closer to the black line, and the distance between the blue line and the black line fluctuated much more. Therefore, the result of the algorithm using the function f was more stable.

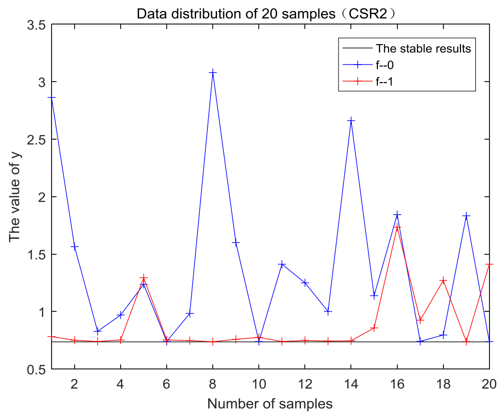


FIGURE 10. Data distribution-y of 20 samples (CSR2).

This showed that the use of f indeed improve the stability of the solution to parameter estimation to G-O model.

B. COMPARISON OF THREE ALGORITHMS

This paper had verified the necessity and effectiveness of using the function f. This section compared the experimental results of the three algorithms PSO, SSA and SSA-PSO based on the use of f. The parameters of the SSA-PSO algorithm

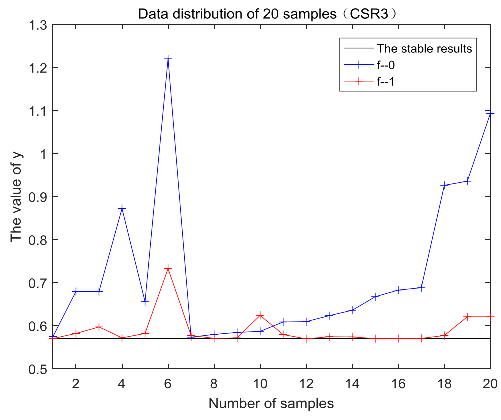


FIGURE 11. Data distribution-y of 20 samples (CSR3).

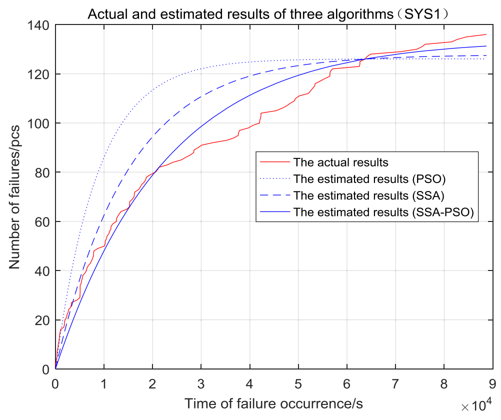


FIGURE 12. Actual and estimated results of three algorithms (SYS1).

were set as shown in part E of Section II, and the parameters of SSA, PSO and hybrid algorithms were set as follows:

- SSA:
  - ✧ the population number: pop=10;
  - ✧ the maximum number of iterations: M=10;
  - ✧ the proportion of explorers is 20%;
  - ✧ the proportion of sparrows who are aware of danger is 20%;
  - ✧ the safety threshold is 0.8;
  - ✧ the position of each ppaper, the parameter b of the G-O model, is initialized to a random number between (0, 1).
- PSO:
  - ✧ the number of ppapers: pop = 10;
  - ✧ the maximum number of iterations: M = 10;
  - ✧ the learning factor: $c_1 = c_2 = 1.5$;
  - ✧ the inertia weight: w = 0.9;
  - ✧ the position of each ppaper, the parameter b of the G-O model, is initialized to a random number between (0, 1);
  - ✧ the speed is initialized to a random number between [−1, 1].

1) FITTING AND PREDICTION

In order to intuitively feel the results of model fitting and prediction, and to compare the estimation effects of the three algorithms on the model, this section used three algorithms to estimate the parameter b. After obtaining the optimal parameter b, use the maximum likelihood formula to solve the parameter a based on formula 8 and 9. Based on the two parameters, the failure number estimation curve of the model can be drawn based formula 1. The model results of the three algorithms and the actual failure numbers were shown in Figures 12∼16:

In Figures 12∼16, the red curve represented the actual number of failures over time. Observing each graph separately, we can find that the estimation and prediction curve of SSA-PSO proposed in this paper was closer to the actual curve than the curve of a single algorithm, which showed that the hybrid algorithm improved the accuracy of the algorithm results. The curve was exponentially distributed, and the slope of the curve was increasing, indicating that the time interval for software failures was increasing.

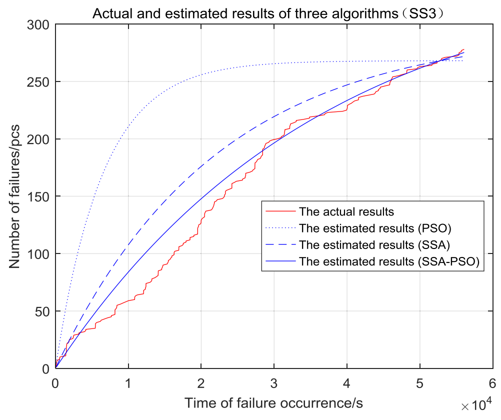


FIGURE 13. Actual and estimated results of three algorithms (SS3).

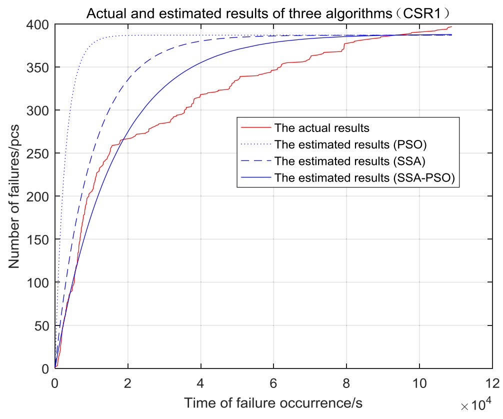


FIGURE 14. Actual and estimated results of three algorithms (CSR1).

2) DATA ANALYSIS

In order to further compare the convergence and stability of the three algorithms, the three algorithms were run 20 times,

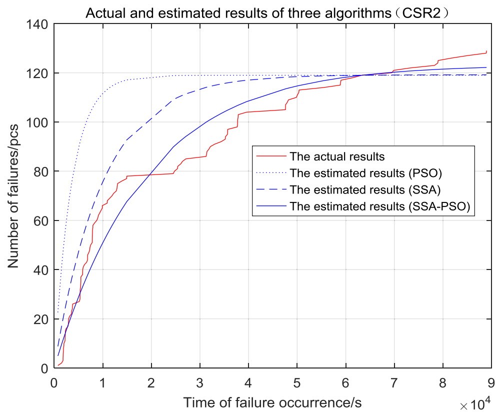


FIGURE 15. Actual and estimated results of three algorithms (CSR2).

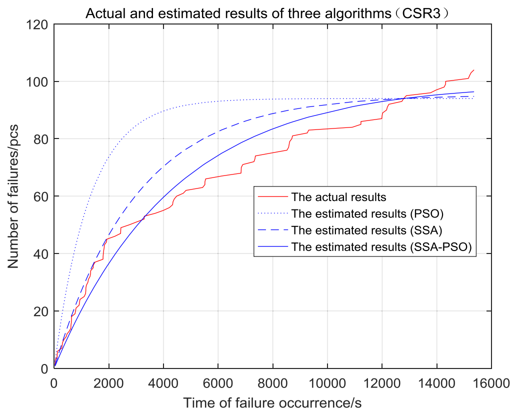


FIGURE 16. Actual and estimated results of three algorithms (CSR3).

and the results of 20 times were recorded. Among them, the best and worst results of each algorithm and the average value of 20 times of y value are shown in Table 6:

The basic meaning of the data in Table 6 had been explained in part A of Section III. The difference was that the maximum number of iterations of the three algorithms is 10.

This section only compared the difference in results between the algorithms. From the data change of the y value in the table, it can be seen that the result of the hybrid algorithm was smaller than that of the single algorithm, whether it was the optimal value, the worst value or the average value, which was, the solution result of the hybrid algorithm was better.

Although the gap between the optimal values was not particularly obvious, it was determined by the randomness of the algorithm itself, and algorithms with poor performance can occasionally solve slightly high-precision results.

However, from the worst value and average value, it can be clearly found that the accuracy of the hybrid algorithm was higher, and the convergence speed was faster.

3) ANALYSIS OF CONVERGENCE

The previous section speculated that the convergence speed of the hybrid algorithm was faster than that of the single algorithm. In order to compare the convergence of the algorithm, this section statistically analyzed the iterative changes of the y value in the three algorithms. Took the iterative process of the optimal y value achieved in Table 6 for comparison, as shown in Figures 17∼21:

With reference to the implementation of the hybrid algorithm in part E of Section II, the three algorithms all use the function f to initialize the parameter b, and all perform iterative search on the function f based on their own algorithms.

From Figures 17∼21, it can be found that at the beginning of convergence, the y value of the hybrid algorithm had already converged to near the optimal value from the

TABLE 6. The best and worst values of different algorithms.

Data Sets	Algorithms	Fitness value(y)			Difference (error_1)		Root mean square error (RMSE)	
		the worst	the best	average	the worst	the best	the worst	the best
SYS1	PSO	6.23	1.39	5.36	1.00	0.98	6.20	5.69
	SSA	5.11	0.61	1.64	1.00	0.87	6.20	2.62
	SSA-PSO	1.72	0.61	0.76	0.99	0.87	5.97	2.78
SS3	PSO	9.28	6.29	8.71	1.00	1.00	6.20	6.18
	SSA	9.11	0.61	2.24	1.00	0.43	6.20	1.66
	SSA-PSO	3.71	0.61	1.04	0.92	0.44	5.25	1.57
CSR1	PSO	11.3	5.81	10.2	1.00	1.00	6.20	6.20
	SSA	7.40	1.19	2.21	1.00	0.86	6.20	5.69
	SSA-PSO	2.87	1.18	1.40	1.00	0.87	6.20	5.73
CSR2	PSO	6.26	1.74	5.43	1.00	0.96	6.20	6.14
	SSA	5.41	0.75	1.59	1.00	0.31	6.20	4.93
	SSA-PSO	1.80	0.74	0.86	0.96	0.06	6.15	4.40
CSR3	PSO	5.31	1.66	4.12	1.00	0.99	6.20	6.13
	SSA	1.53	0.57	0.77	0.99	0.88	6.10	4.49
	SSA-PSO	0.84	0.57	0.61	0.95	0.88	5.55	4.49

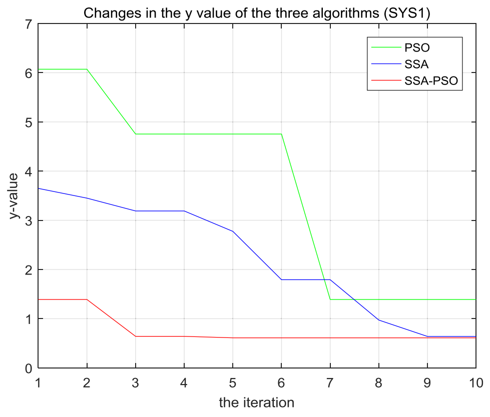


FIGURE 17. Changes in the y value of the three algorithms (SYS1).

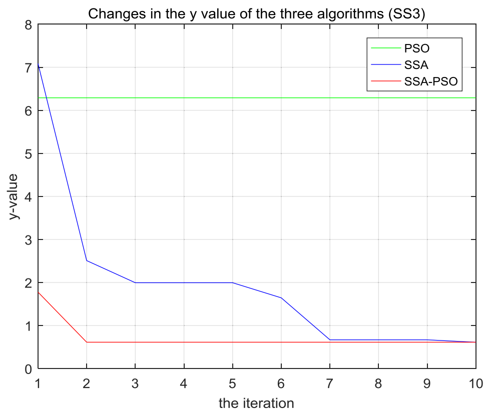


FIGURE 18. Changes in the y value of the three algorithms (SS3).

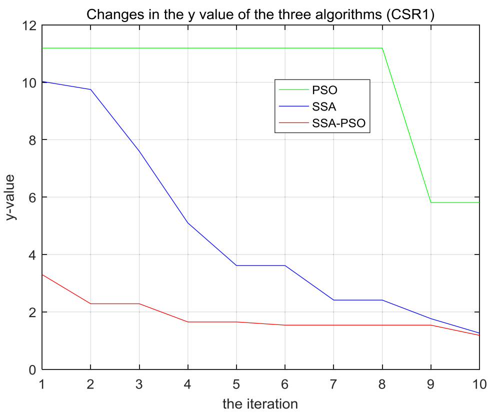


FIGURE 19. Changes in the y value of the three algorithms (CSR1).

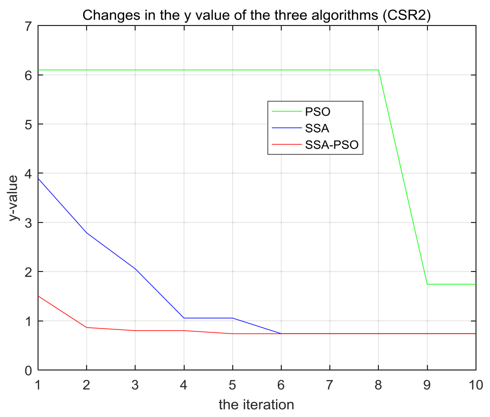


FIGURE 20. Changes in the y value of the three algorithms (CSR2).

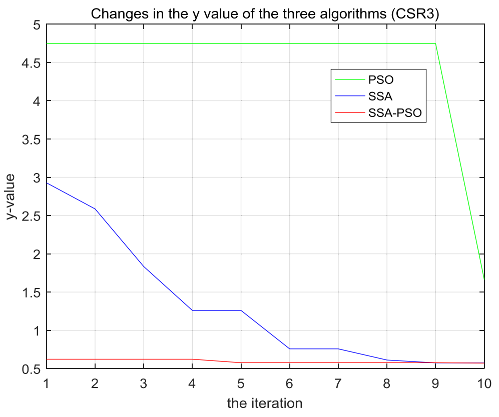


FIGURE 21. Changes in the y value of the three algorithms (CSR3).

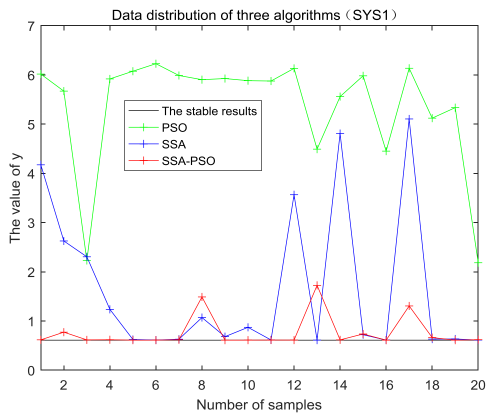


FIGURE 22. Data distribution-20 of three algorithms (SYS1).

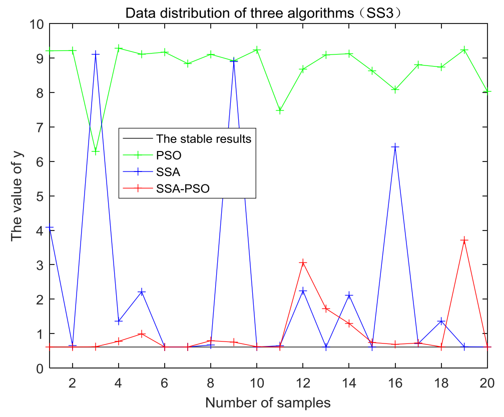


FIGURE 23. Data distribution-20 of three algorithms (SS3).

beginning. Although this was the result of the convergence of the hybrid algorithm on the function f, it also showed that the convergence speed of the hybrid algorithm was indeed higher than the other two algorithms.

Not only that, even if the hybrid algorithm was already near the optimal solution, it can still converge quickly. Since the y value of the hybrid algorithm at the beginning of the iteration was already a relatively optimal value, as the algorithm continues to converge, the result of the hybrid algorithm was stable towards the optimal value.

4) ANALYSIS OF STABILITY

In order to compare the stability intuitively, this section compared the results of the three algorithms running 20 times when the maximum number of iterations is 10. The statistical distribution of each group of data was shown in Figure 22∼26:

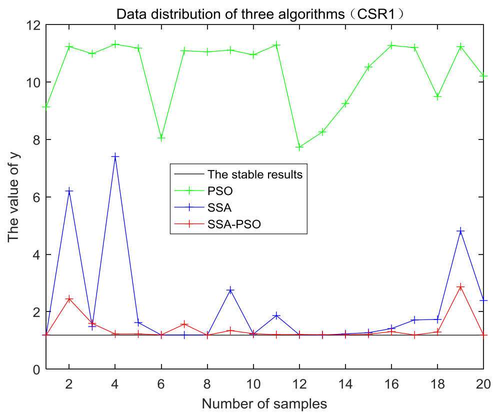


FIGURE 24. Data distribution-20 of three algorithms (CSR1).

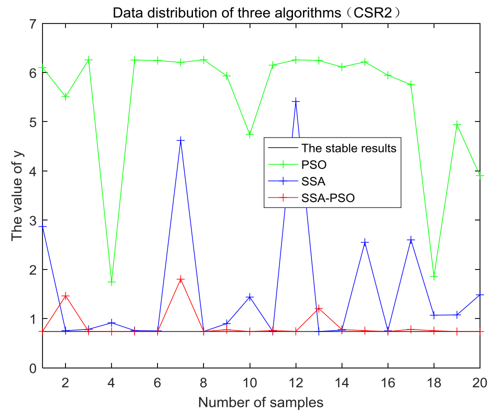


FIGURE 25. Data distribution-20 of three algorithms (CSR2).

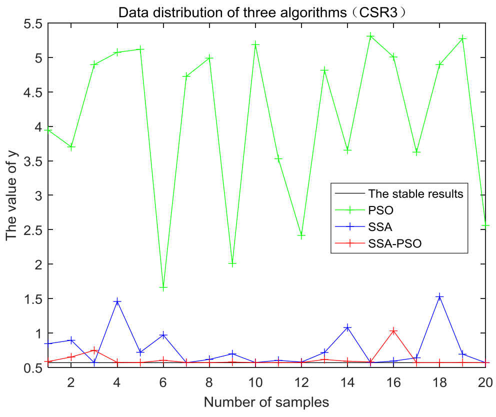


FIGURE 26. Data distribution-20 of three algorithms (CSR3).

In Figures 22∼26, the horizontal black straight line was at the bottom, and its corresponding y value represented the optimal value that the algorithm can achieve. The gap between it and other lines represented the gap between the y value calculated by the algorithm and the optimal value each time. Therefore, the smaller the distance change, the more stable the y value obtained by the algorithm. Observing the Figures 22∼26, it was obvious that the red line of the hybrid algorithm was closer to the black line, and the distance fluctuation was the smallest. The result of the hybrid algorithm was the most stable, which showed that the hybrid algorithm not only converged fast, but also had the best stability.

IV. DISCUSSION

In the experimental simulation of Section III, part A compared the algorithm results of whether to use the control function of parameter b or not. It was found that using the control function can improve the accuracy of the results. The first subsection 1) used the iterative convergence graph to visually show the improvement of the convergence speed by using the control function. The second subsection 2) directly compared the sample results of the algorithm running 20 times and found that the use of control functions can also ensure the stability of the results.

After verifying the effectiveness of the control function, part B compared and analyzed the three algorithms based on the use of the control function. In the first subsection 1), the actual failure graph and the model results of different algorithms were used for intuitive comparison, and it was found that the estimation and prediction of the hybrid algorithm was the closest to the actual results. The second subsection 2) was a statistical analysis of the optimal value, worst value and average value of the algorithm running 20 times, and it was found that the accuracy of the hybrid algorithm was higher than that of the single algorithm. The third and fourth subsection also found that the convergence speed and stability of the hybrid algorithm were better than the single algorithm through the comparison of graphs.

Here, we conducted an in-depth analysis and discussion of the experimental results in section III. The fitness function f proposed in this paper was a unimodal function with only one maximum value. Therefore, the requirement for the algorithm was to converge quickly and to improve the convergence rate, you can start with the initialization of the algorithm and the search process. As we all know, the initialization value of the algorithm was the cornerstone of the algorithm, and an excellent initialization value can make the algorithm quickly converge to the extreme value. Therefore, by using the fitness function to optimize the initialization result of the parameter b, to ensure that the algorithm starts to search for the best in the region near the extreme value, the accuracy and stability of the algorithm result can naturally be improved. After solving the initialization problem, this paper used the extremely fast convergence speed of the PSO to optimize the search process of the SSA, that was, the improved method of PSO was added before the update of SSA.

In fact, hybrid PSO before SSA can also be understood as optimizing the initialization value of SSA. In short, the idea of improving the convergence speed of the algorithm in this paper was to promote the algorithm to be in the extreme region at an early stage to ensure the smooth solution process. One more thing to be noted was that the hybrid algorithm proposed in this paper not only converges quickly on unimodal functions, but also performs well on multimodal functions.

In the convergence graph of the third subsection of part B, the initial y value was the result of the algorithm's convergence to the function f, and from the graph comparison, it can be found that the initial y value of the hybrid algorithm was much smaller than the value of the individual algorithm. This showed that the hybrid algorithm had the best convergence to function f which was a multimodal function.

There are three ways to ensure the stability of the results. First, the multi-role search of the Sparrow Algorithm can ensure the algorithm approaches the global optimal solution based on multiple local optimal solutions, reducing the possibility of inferior solutions. The second is that the value accuracy of the fitness function in this paper is relatively high, so the solution obtained when meeting the accuracy requirements of the fitness function is also close to the optimal solution, and the uncertainty of the result is less. In the third aspect, by observing Tables 1 to 5 in Section 3, it can be found that as the number of iterations increases, the optimal, worst, and average values of y begin to gradually decrease. This is because the more iterations, the more convergent the algorithm naturally, and the result of the algorithm gradually stabilizes. Therefore, the number of iterations can be increased to solve the uncertainty of the algorithm results.

V. CONCLUSION

This paper summarized the advantages and disadvantages of the swarm intelligence algorithm and improvement strategies to get inspiration. In the problem of using swarm intelligence algorithm to estimate the parameters of the typical software reliability model G-O model, a hybrid algorithm based on SSA and PSO was proposed, and the parameter and software defections was estimated and predicted.

Experimental results showed that the hybrid method of SSA and PSO proposed in this paper can improve the accuracy of software reliability model estimation and prediction. Compared with a single algorithm, it had a great improvement both in convergence speed and in stability, so that the algorithm searched quickly converges to the optimal value, and the stability of the algorithm result was guaranteed. In order to speed up the convergence speed and further improve the accuracy and stability of the algorithm results, this paper also constructed a fitness function that can effectively control the initialization of the parameter b through the maximum likelihood formula.

This paper estimated and predicted software defections on the classic G-O model. In the future research, we will consider to make improvement on the strategy of updating ppaper position and more application in other software reliability models.

REFERENCES

[1] F. Ogigau-Neamtiu and H. Moga, "A cyber threat model of a nation cyber infrastructure based on Goel-Okumoto port approach," *Land Forces Acad. Rev.*, vol. 23, no. 1, pp. 75–87, Mar. 2018.

[2] E. Abuta and J. Tian, "Reliability over consecutive releases of a semiconductor optical endpoint detection software system developed in a small company," *J. Syst. Softw.*, vol. 137, pp. 355–365, Mar. 2018.

[3] S. Ameli, F. Naghdy, D. Stirling, G. Naghdy, and M. Aghmesheh, "Chemotherapy-induced fatigue estimation using hidden Markov model," *Biocybernetics Biomed. Eng.*, vol. 39, no. 1, pp. 176–187, Jan. 2019.

[4] C. Jin and S.-W. Jin, "Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization," *Appl. Soft Comput.*, vol. 40, pp. 283–291, Mar. 2016.

[5] T. Yaghoobi, "Parameter optimization of software reliability models using improved differential evolution algorithm," *Math. Comput. Simul.*, vol. 177, pp. 46–62, Nov. 2020.

[6] A. F. Sheta and A. Abdel-Raouf, "Estimating the parameters of software reliability growth models using the grey wolf optimization algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 4, pp. 499–505, 2016.

[7] A. Choudhary, A. S. Baghel, and O. P. Sangwan, "An efficient parameter estimation of software reliability growth models using gravitational search algorithm," *Int. J. Syst. Assurance Eng. Manage.*, vol. 8, no. 1, pp. 79–88, Mar. 2017.

[8] Sangeeta and K. Sharma, "An ecological space based hybrid swarm-evolutionary algorithm for software reliability model parameter estimation," *Int. J. Syst. Assurance Eng. Manage.*, vol. 11, pp. 77–92, Feb. 2020.

[9] M. Zhu and H. Pham, "A software reliability model with time-dependent fault detection and fault removal," *Vietnam J. Comput. Sci.*, vol. 3, no. 2, pp. 71–79, May 2016.

[10] S. Lohmor and B. B. Sagar, "Estimating the parameters of software reliability growth models using hybrid DEO-ANN algorithm," *Int. J. Enterprise Netw. Manage.*, vol. 8, pp. 247–269, Jun. 2017.

[11] X. Jiankai, "Research and application of a new type of swarm intelligence optimization technology," M.S. thesis, Dept. Inf. Sci. Technol., Donghua Univ., Shanghai, China, 2020.

[12] L. Yali and W. Shuqin, "Comparative research on several new swarm intelligence optimization algorithms," *Comput. Eng. Appl.*, vol. 56, pp. 1–12, Oct. 2020.

[13] X. Lv, X. D. Mu, J. Zhang, and Z. Wang, "Chaos sparrow search optimization algorithm," *J. Beijing Univ. Aeronaut. Astronaut.*, vol. 12, pp. 1–10, Aug. 2020.

[14] L. Xin and M. Xiaodong, "Multi-threshold image segmentation based on improved sparrow search algorithm," *Syst. Eng. Electron.*, vol. 2020, pp. 1–12, Nov. 2020.

[15] L. Xu, Y. Jiang, and L. Wang, "Thermal decomposition of rape straw: Pyrolysis modeling and kinetic study via particle swarm optimization," *Energy Convers. Manage.*, vol. 146, pp. 124–133, Aug. 2017.

[16] Y. Ding, W. Zhang, L. Yu, and K. Lu, "The accuracy and efficiency of GA and PSO optimization schemes on estimating reaction kinetic parameters of biomass pyrolysis," *Energy*, vol. 176, pp. 582–588, Jun. 2019.

[17] L. A. Bewoor, V. C. Prakash, and S. U. Sapkal, "Production scheduling optimization in foundry using hybrid particle Swarm Optimization algorithm," *Procedia Manuf.*, vol. 22, pp. 57–64, Jun. 2018.

[18] H. E. Espitia and J. I. Sofrony, "Statistical analysis for vortex particle swarm optimization," *Appl. Soft Comput.*, vol. 67, pp. 370–386, Jun. 2018.

[19] I. B. Aydilek, "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems," *Appl. Soft Comput.*, vol. 66, pp. 232–249, May 2018.

[20] A. K. Shrivastava and R. Sharma, "Generalized modeling for multiple release of two dimensional software reliability growth model," *Int. J. Innov. Res. Develop.*, vol. 5, pp. 22–29, Jan. 2016.

[21] J. Iqbal, "Analysis of some software reliability growth models with learning effects," *Int. J. Math. Sci. Comput.*, vol. 2, no. 3, pp. 58–70, Jul. 2016.

LIU YANG was born in Suqian, Jiangsu, China, in 1995. She received the bachelor's degree from the Jinling Institute of Technology, in 2018. She is currently pursuing the master's degree in electronics and communication engineering with the Jiangsu University of Science and Technology, Zhenjiang.

Her research interests include swarm intelligence and software reliability.

ZHEN LI was born in Xinghua, Taizhou, Jiangsu, China, in 1977. He graduated from Nanjing Normal University, majoring in management. He received the bachelor's degree in computer application from Nanjing University, the M.S. degree in signal and information processing from the Jiangsu University of Science and Technology, Zhenjiang, China, in 2006, and the Ph.D. degree in aerospace system engineering from Beihang University, Beijing, China, in 2011.

From 2011 to 2014, he was a Lecturer with the School of Electronics and Information, Jiangsu University of Science and Technology, where he has been an Assistant Professor, since 2014. He is the author of more than 15 articles and has four inventions. His research interests include reliability and system engineering, safety engineering, swarm intelligence, resilience, and software testing.

DONGSHENG WANG was born in Yancheng, Jiangsu, China, in 1982. He received the master's degree in computer science from the Jiangsu University of Science and Technology, in 2007, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2012. His primary research interests include software engineering, question answering, and natural language understanding.

HONG MIAO was born in Yangzhou, Jiangsu, China, in 1982. She received the master's degree in management science and engineering from the Jiangsu University of Science and Technology, in 2007, and the Ph.D. degree in control engineering from the Nanjing University of Science and Technology, in 2016. Her primary research interest includes the modeling and data analysis of MIS.

ZHAOBIN WANG was born in Qiqihaer, Heilongjiang, China, in 1982. He received the master's and Ph.D. degrees from the Harbin Institute of Technology, in 2007 and 2013, respectively. He held a postdoctoral position with the Faculty of Mechanical Engineering and Automation, Zhejiang Sci-Tech University, from 2014 to 2018. He is currently an Associate Professor with the Jiangsu University of Science and Technology. His primary research interests include reliability theory and test in electronics, storage reliability, reliability, and PHM.