INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 18, Issue: 2, Month: April, Year: 2023
Article Number: 5049, https://doi.org/10.15837/ijccc.2023.2.5049

Directed Search Based on Improved Whale Optimization Algorithm for Test Case Prioritization

Bin Yang, Huilai Li, Ying Xing, Fuping Zeng*,
Chengdong Qian, Youzhi Shen, Jiongbo Wang

Bin Yang
China Unicom Research Institute
No. 9 Shouti South Road, Haidian District, Beijing 100048, China,
researcher_yang@outlook.com

Ying Xing, Huilai Li
School of Artificial Intelligence
Beijing University of Posts and Telecommunications, China
No. 10, Xitucheng Road, Haidian District, Beijing 100876, China
xingying@bupt.edu.cn, lihuilai@bupt.edu.cn

Fuping Zeng*
School of Reliability and Systems Engineering
Beijing University of Aeronautics and Astronautics, China
No. 37, Xueyuan Road, Haidian District, Beijing 100191, China
*Corresponding author: zfp@buaa.edu.cn

Chengdong Qian, Youzhi Shen, Jiongbo Wang
Phytium Technology Co., Ltd., China
7th Floor, Quantum Core Block, Zhichun Road, Haidian District, Beijing 100086, China
qianchengdong@phytium.com.cn, shenyouzhi@phytium.com.cn, wangjiongbo@phytium.com.cn

Abstract

With the advent of the information age, the iterative speed of software update is gradually accelerating which makes software development severely limited by software testing. Test case prioritization is an effective way to accelerate software testing progress. With the introduction of heuristic algorithm to this task, the processing efficiency of test cases has been greatly improved. However, to overcome the shortcomings of slow convergence speed and easy fall into local optimum, the improved whale optimization algorithm is proposed for test case prioritization. Firstly, a model called n-dimensional directed search space is established for the swarm intelligence algorithm. Secondly, the enhanced whale optimization algorithm is applied to test case prioritization while the backtracking behavior is conducted for individuals when hitting the wall. In addition, a separate storage space for Pareto second optimization is also designed to filter the optimal solutions of the multi-objective tasks. Finally, both single-objective and multi-objective optimization experiments are carried out for open source projects and real-world projects, respectively. The results show that the improved whale optimization algorithm using n-dimensional directed search space is more conducive to the decisions of test case prioritization with fast convergence speed.

Keywords: software testing, test case prioritization, swarm intelligence algorithm, whale optimization algorithm, directed search space.

1 Introduction

Under the diversified user requirements, the software system is updated iteratively more frequently[1]. To discover new defects after software upgrading, necessary tests must be carried out[2]. However, every time a software is upgraded, new test cases need to be added to the original test case set, which leads to a gradual increase in test costs due to its large scale[3, 4]. Especially for large-scale software regression testing, the process may take a long time. Enterprises often need to set up pretreatment modules to complete production more efficiently[5, 6]. Wong et al.[7] first proposed research on test case prioritization (TCP). By handling the test cases and determining the test order according to the priority, the test cases that can cover more nodes can be executed early so that the internal defects can be found faster, and the time consumed for software testing can be significantly reduced[8, 9].

Test case prioritization is a pre-decision process for software testing. The test cases that can achieve faster full coverage of the system will be prioritized first. TCP technology plays a vital role in reducing the cost of software testing and speeding up the progress of digital testing[10]. Most of the early studies focused on prioritizing itself. For example, Fan Hui et al.[11] gave methods of total prioritization algorithm and additional prioritization algorithm according to the definition of test case prioritization. However, it has been proved that the prioritization of test cases belongs to a NP-hard problem[12], so the search without information or search with information can hardly meet this requirement. As we know, the overall prioritization scale of test cases is much larger than the scale of the test cases themselves, which is the main problem to be faced in TCP.

With the advance of the research on test case prioritization, the solution to this problem has gradually transited to intelligent algorithms. For example, Dharmveer et al.[13] proposed a fuzzy inference system, which uses the fault detection rate and program execution time as indicators to verify its effectiveness[14]. Zhang et al.[15] proposed an intelligent test case prioritization method based on the genetic algorithm. They designed corresponding coding strategies, crossover operators, mutation operators, and fitness functions for program coverage, which improved the automation level of software testing. Xu et al.[16] designed an intelligent test case prioritization algorithm based on the artificial immune algorithm (IA), which has great global search performance. Among these intelligent algorithms, the performance of swarm intelligence algorithm is more obvious. The swarm intelligence algorithm with the advantages of good convergence and comprehensive optimization can effectively avoid falling into the problem of local optimization.

The so-called swarm intelligence algorithm is an optimal search that simulates the intelligent activities of natural creatures. After setting the population and specific rules, it can completes the search task in the search space [17–20]. It can give consideration to both the purpose and randomness of the search. At present, more and more researchers the use swarm intelligence algorithms to solve the TCP problem and have achieved good results. For example, Andreea et al.[21] proposed a test case prioritization strategy based on the ant colony algorithm (ACO). They tried to find the most significant fault with the highest severity first according to the number and severity of defects. Xing et al.[22] used the artificial fish school algorithm (AFSA) to optimize the test case prioritization by using the swarm behavior, foraging behavior, and tail-chasing behavior and verified its effectiveness through experiments. Gouda et al.[23] combined the Crow search algorithm and chaotic Drosophila optimization algorithm to enhance the optimization results of test case prioritization. Anu and Sangwan[24] use the bat algorithm to research the TCP problem, and their results are greatly improved compared to traditional methods. Manar et al.[25] designed an improved Harris Hawk optimization algorithm to improve prioritizing efficiency in many ways. Bajaj et al.[26] used the discrete cuckoo search algorithm and introduced the adaptive strategy of asexual reproduction to reasonably improve the solutions. Through reasonable search principles, the swarm intelligence algorithm can complete the global search[27]. At the same time, it has improved in solving the problems of falling into the local optimum (LO) and converging too fast.

However, in the past, when the swarm intelligence algorithm was used to solve the TCP problem, it often only improved the solution of the optimal agent without establishing a clear search space. This paper applies a whale optimization algorithm (WOA) further improve the solution to the TCP problem through reasonable modeling. The whale optimization algorithm is a new method after the Grey Wolf algorithm which has the advantages of few input parameters, strong independence of the search mechanism and fast solving speed[28, 29]. However, the initial whale optimization algorithm converges quickly and sometimes falls into local optimization[30]. Therefore, a whale optimization algorithm with an enhanced search mechanism is proposed subsequently[31]. In addition, the usual TCP process based on an intelligent algorithm only partially updates the solution, which is challenging to solve the problem with enormous complexity. Therefore, this paper establishes an n-dimensional directed search space and applies reinforced exploration mechanism whale optimization algorithm (REM-WOA) to optimize test case prioritization decisions. Finally, our experiments show that REM-WOA can achieve better results in less iterations when dealing with TCP problems.

The main contributions of this paper are as follows:

•For the test case prioritization problem based on swarm intelligence algorithm, a general modeling method of n-dimensional directed search space is proposed.

•This paper introduces different variants of the whale optimization algorithm. We select the best one through experiments, and apply it to the solution of the TCP problem.

•The backtracking problem of the swarm intelligence algorithm in directed search space is proposed to avoid the search limitation caused by individuals hitting the wall.

•The prioritization effects of different heuristic algorithms are compared through experiments, and the performance is analyzed for single-objective and multi-objective optimization.

•Different heuristic algorithms are applied to the TCP problem of the real-world gateway project. Their decision effects and convergence rates are compared and analyzed.

The rest of this paper is arranged as follows. The second section introduces the basic technical background. The third section gives the specific modeling process and algorithm principle. The fourth section introduces the specific settings of the experiment. In the fifth section, the effectiveness experiment, single-objective experiment and multi-objective experiment are shown, respectively. Section 6 prioritizes the test cases of the real-world project. Section 7 summarizes the full text and points out the future research direction.

2 Technical background

This section reviews the basic definition of test case prioritization. The whale optimization algorithm, the WOA of population updating, convergence-weighted WOA and REM-WOA are also introduced.

2.1 Test case prioritization

To better illustrate the problem of test case prioritization, this paper introduces a specific example. In Table 1, there are 5 test cases and 9 elements to be covered. Assuming that T_1-T_3 is selected, all elements can be covered as quickly as possible. Obviously, the execution sequence of T_1-T_3-T_2-T_5-T_4 is better than the original sequence of T_1-T_2-T_3-T_4-T_5. The tester can handle the problems in the program more quickly.

Table 1: Element coverage of test case set

Test case	Element								
	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$	$f8$	$f9$
T_1	1	1		1	1	1			1
T_2		1	1	1			1		1
T_3			1				1	1	
T_4	1	1	1					1	1
T_5	1				1	1	1		

Test case prioritization is frequently used in software testing, which can significantly improve the efficiency of regression testing. The test case prioritization problem is defined as[32]: The given test case set is T, all possible prioritizing sets of test cases in T are Pt, and the mapping from Pt to the real number set is F. The prioritization problem of test cases is to find $T' \epsilon Pt$, so that for any $T'' \epsilon Pt$ and $T' \neq T''$, there is $|f(T') \geqslant f(T'')|$.

Where, f is a quantitative description of the objectives to measure the effectiveness of the prioritization. Here, we define that the larger the f, the better the effect. In practical applications, testers set different test objectives, like the coverage speed of test points and the detection rate of faults[33, 34].

2.2 Selection of test objectives

The swarm intelligence algorithm requires determining the fitness function, that is, the f value in the above TCP definition. It can be divided into single-objective and multi-objective optimization according to requirements.

The effective execution time (EET) of the test case sequence represents the time consumed by the test case when the test case sequence reaches the maximum statement coverage for the first time. In addition, during single-objective optimization, many optimization objectives for code coverage have been developed[35]. For the black box testing, if an element is covered for enough time, it will nearly have no mistakes[36]. Therefore, this paper selects the average percentage of test point coverage (APTC) as one of the optimization objectives to indicate the coverage speed to test points in the program.

The specific definitions of EET and APTC are as follows:

$$EET = \sum_{i=1}^{N'} ET_i \tag{1}$$

$$APTC = 1 - \frac{TT_1 + TT_2 + \ldots + TT_M}{M \cdot N} + \frac{1}{2 \cdot N} \tag{2}$$

where N represents the number of test cases, M represents the number of statements of the program, T_i represents the position of the test case in the execution sequence where the test point is detected for the first time, N' represents the number of test cases executed when the maximum statement coverage is reached for the first time, and ET_i represents the time consumed by the execution of the ith test case.

2.3 Whale optimization algorithm and its variants

To solve the problem of test case prioritization, we introduced the algorithms of whale optimization, including the original WOA, PR-WOA(population redistribution based whale optimization algorithm), CAW-WOA(convergence adaptive weighting based whale optimization algorithm) and REM-WOA. These algorithms concern both randomness and certainty, and can obtain better optimization results.

2.3.1 Whale optimization algorithm

WOA is a swarm intelligence algorithm inspired by whales preying. According to the random agents and best agents in the population, the method simulates the spiral bubble attack when whales surround prey. It is mainly divided into three search mechanisms:

(1) Surround prey: The information of individuals in the population is shared, which enables each individual to move towards the position of the prey and continuously reduce the enclosure.

(2) Random search: To improve the global search ability of whales, each agent has a random search behavior, which slows down the search speed and prevents the results from local optimization.

(3) Spiral search: When a certain probability is reached, the whale will update the spiral position and spit out bubbles to spiral upward to surround the prey.

The above three strategies are executed according to the probability parameters. The fitness function of the algorithm is $f(x_i)$, and the individual position is determined by the fitness.

2.3.2 WOA of population renewal strategy

Although the WOA strategy has good optimization ability, it often converges searching quickly and easily falls into local optimization. When updating the individual position of the population in each iteration, the following three position redistribution strategies need to be combined.

(1) Random substitution method: This population redistribution strategy aims to protect the best solution while enhancing other stochastic strategies. The strategy points out that the position of the current agent will be replaced by the position of the best agent.

(2) Small-scale redistribution method: This method aims to enhance the algorithm's global optimization ability and significantly avoid omitting the best value in the process of the whale moving toward the prey.

(3) Population inverse solution: The main idea is to generate an inverse solution for each individual in the process of optimization, so as to update the population information and obtain the obest individuals as the new population position[37]. The essence of this method is to set up a new individual at a position symmetrical to the current individual's position.

2.3.3 Convergent adaptive weighted WOA

When the search space is large, the adaptive weighting strategy can refine the search process, accelerate the convergence speed of the algorithm, and find the best solution faster. When the search space is large, this strategy can enhance the global search ability and avoid solving the LO problem. The essence of this method is to adjust the speed at which other individuals approach a certain point. So it can balance the local search and the global search.

2.3.4 Reinforced exploration mechanism whale optimization algorithm

The REM-WOA introduces the population redistribution strategy and the adaptive weighting strategy into the whale optimization algorithm, which enhances the global optimization ability and convergence ability, and it also avoids falling into the local optimization. By changing the individual moving speed and the updating process of the fitness function, the optimal population of each iteration is determined, which ensures the quality of the final solutions.

3 Test case prioritization based on WOA

Since the data processed by the whale optimization algorithm are numerical values, the simple test case information cannot be directly input into the model. Therefore, the corresponding search space should be designed according to the prioritization requirements. This chapter introduces the preprocessing process of search space, the whale optimization algorithm, the edge problem, and the secondary selection of the optimal solution.

3.1 Establishment of search space

After the upper and lower limits of the search space are set, the population can move for searching tasks. However, if you want to combine the TCP problem with it, you need to manually bind a certain sequence to a specific location in the search space. Since there may be hundreds of test cases, and the number of them in the whole set is even more challenging to count, it is necessary to select them randomly.

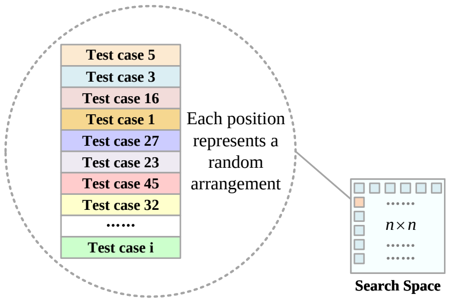

Figure 1: Search space establishment.

Assuming that there are two optimization objectives, take the two-dimensional search space as an example. First, we should extract an appropriate number of test cases and randomly generate n^2 possible orders. As shown in Figure 1, each sort is taken as an element, and these elements are arranged in a square to constitute a global search. All whale individuals can only move in the square array, so each position can only be taken as an integer. The comprehensiveness of prioritization depends on the number of test cases extracted and the size of the square array.

3.2 Search space preprocessing

After the search space is determined, the swarm intelligent algorithm can be used to search. But, no matter what kind of swarm intelligence algorithm is used, its essence is random search. This is because there is no association between each element in the search space, and there is no mathematical relationship between the adjacent elements. Therefore, we must preprocess the search globally for a certain trend relationship between each element. We mainly designed two methods, and the specific principle is shown in Figure 2.

The first method is based on the maximum value. For the optimization objective 1, the square array is rearranged according to the maximum value of each row. For optimization objective 2, the square array is rearranged according to the maximum value or minimum value of each column. If the local search ability of the algorithm is strong, the method is more likely to obtain the real optimal solution. But the execution result may be unstable, so multiple experiments are required. The second method is based on the average value. For the optimization objective 1, the square array is rearranged according to the average value of each row. For the optimization objective 2, the square array is rearranged according to the average value of each column. This method may not find the real best, but the execution result is more stable, and the trend relationship between various elements is more obvious.

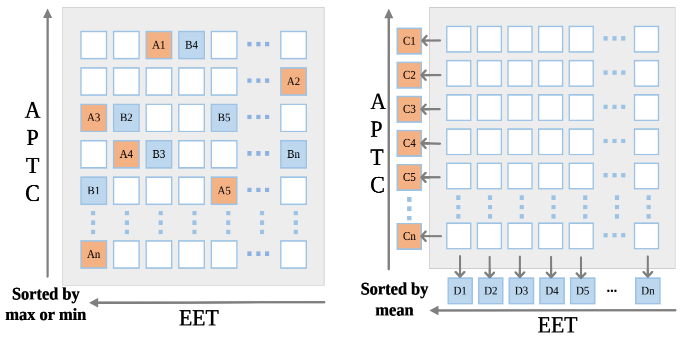

Figure 2: Search space preprocessing.

3.3 Whale optimization model

After completing the establishment of the global search and data processing, the population is redistributed to the search space, so that the swarm intelligence algorithm can be used in it. We take the REM-WOA as an example to give a complete solution process.

3.3.1 Search strategy

REM-WOA has three optimized search mechanisms: surrounding prey, random search and spiral search[38]. Each iteration executes one search strategy according to the random variables.

(1) Surrounding prey: When the random variable p is greater than 0.5, and the absolute value of the random iteration variable A is less than 1, the strategy of surrounding prey is implemented, and the search agent approaches the best individual. The position update equation is:

$$B = \left| C \cdot \vec{X}_{best}(t) - \vec{X}_{local}(t) \right| \tag{3}$$

$$\vec{X}_{local}(t+1) = \vec{X}_{best}(t) - A \cdot B \tag{4}$$

where $\vec{X}_{best}(t)$ represents the position of the best individual of the tth iteration in the population and $\vec{X}_{local}(t)$ represents the position of one individual in the tth iteration. To adjust its convergence speed, after adding the convergence weighting strategy, Eq.4 becomes:

$$\vec{X}_{local}(t+1) = v \cdot \vec{X}_{best}(t) - A \cdot B \tag{5}$$

$$v = 2(rand - 0.5)/exp(tan(\pi \cdot t/t_{max})) \tag{6}$$

where, the size of the weight v changes with the number of iterations, and A and B are coefficient vectors. Their expression methods are:

$$h = 2 - 2 \cdot t/t_{max} \tag{7}$$

$$A = 2h \cdot rand - h \tag{8}$$

$$C = 2 \cdot rand \tag{9}$$

where $rand$ represents a random number between 0 and 1. The size of h is related to the current number of iterations, and It decreases linearly from 2 to 0. And t_{max} is the maximum number of iterations.

(2) Random search: When the random variable p is greater than 0.5 and the absolute value of the random iteration variable A is greater than 1, the random search strategy is executed to make the agent approach the random individual. The location update equation is:

$$B = C \cdot \vec{X}_{rand}(t) - \vec{X}_{local}(t) \tag{10}$$

$$\vec{X}_{local}(t+1) = \vec{X}_{rand}(t) - A \cdot B \tag{11}$$

where $\vec{X}_{local}(t)$ is the position of a random agent in the population at the tth iteration. Similar to the surrounding prey strategy, after the convergence weighting mechanism is added, Eq.11 becomes:

$$\vec{X}_{local}(t+1) = v \cdot \vec{X}_{rand}(t) - A \cdot B \tag{12}$$

(3) Spiral search: When the random variable p is less than 0.5, the spiral search strategy is executed. Here, the position information of the current individual and the prey needs to be determined. Then a spiral formula to simulate the spiral movement of the whale can be established. The expression is:

$$\vec{X}_{local}(t+1) = \vec{X}_{best}(t) + B_p \cdot e^{bl} cos(2\pi l) \tag{13}$$

$$B_p = \left| \vec{X}_{best}(t) - \vec{X}_{local}(t) \right| \tag{14}$$

where b is a constant used to determine the shape of the helix, and l is a random number between -1 and 1.

3.3.2 Fitness update strategy

The fitness updating process of REM-WOA mainly includes three parts: random substitution, population inverse solution and small-scale redistribution. Then three times the number of individuals is generated for further search.

(1) Random substitution: When the following probabilities are met, the solution of the current individual is replaced by the best individual to accelerate the convergence speed of the algorithm[39].

$$tan(\pi \cdot (rand - 0.5)) < (1 - t/t_{max}) \tag{15}$$

(2) Population inverse solution: In order to improve the global search ability of WOA, this strategy can be applied. Assuming that our search space is multidimensional, we know that our existing

population is $(x_1, x_2, \ldots, x_n)$, and the upper bound and lower bound of the search space are $lb(i)$ and $ub(i)$. Then the inverse solution is:

$$\vec{X}_{iop} = lb(i) + ub(i) - \vec{X}_i \tag{16}$$

(3) Small-range redistribution: To improve the global search ability of the model, certain randomness should be added to it, so that all individuals can move freely in the maximum range without crossing. The expression is:

$$\vec{X}_{local}(t+1) = \vec{X}_{local}(t) + r \cdot rand \cdot sign(rand - 0.5) \tag{17}$$

$$r = \frac{|ub - lb|}{2 \cdot s} \tag{18}$$

where r is the moving range and s is the number of individuals.

3.4 Backtracking of whale individuals

Since the newly set search space is not a continuous space, the real-world range of a single dimension of the search space is small. The individuals in the population are very likely to encounter an edge, resulting in some individuals being unable to move. To solve this problem, it is necessary to trace back the whale individuals who hit the edge. The specific principle is shown in Figure 3. Assuming that the side length of the square matrix of the search space is n, the number of backtracking is set to a random integer between 0 and $n/3$. After this operation step, the algorithm's random search ability is further improved, and the problem of population concentration toward the boundary is avoided.

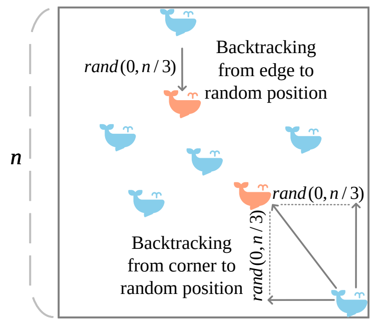
n
rand(0, n / 3)
Backtracking from edge to random position
rand(0, n / 3)
rand(0, n / 3)
Backtracking from corner to random position

Figure 3: Individual whale back-tracing.

3.5 Selection of optimal solution

Although there are certain relations between the adjacent elements, they are not prioritized strictly according to the objective value. So the best solution obtained in the next iteration may not be better than the best solution obtained in the previous iteration. To speed up the solution speed and avoid the waste of the best solution obtained in each iteration, a new set should be created to store all the best solutions in the iteration process. Finally, the Pareto optimal solution set should be obtained. The specific principle is shown in Figure 4.

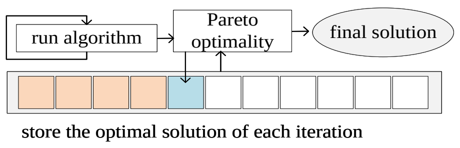

Figure 4: Quadratic selection of optimal solution set.

4 Experimental settings

After modeling, specific experimental configurations are required to solve the TCP problem. This chapter introduces the benchmark functions, tested programs, compared algorithms and experimental parameters.

4.1 Benchmark functions

Before using REM-WOA to prioritize test cases, we first verify its effectiveness and compare the performance of different variants of the whale optimization algorithm. Before the experiment, it is necessary to select a suitable reference function. Because the whale optimization algorithm and its variants converge so fast, selecting a single peak benchmark function will make the contrast effect inapparent. Therefore, four multi-peak reference functions are chosen in this paper[40]. More details are in Table 2.

Table 2: Relevant information of benchmark function

	Benchmark Function	Formula	Range
F1	Rastrigin's Function	$f_1(x) = \sum_{i=1}^{D-1}(x_i^2 - 10cos(2\pi x_i) + 10)$	[-100,100]
F2	High Conditioned Elliptic Function	$f_2(x) = \sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}} x_i^2$	[-800,800]
F3	Ackley's Function	$f_3(x) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i))$	[-800,800]
F4	Griewank's Function	$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{i}) + 1$	[-800,800]

4.2 Tested programs

In this paper, the data set of Siemens open-source test cases are selected for comparative experiments[41]. To ensure the reliability of the test, this paper selects several test suites with the highest utilization rate. We conduct static software measurements on them respectively to meet the diversity of the tested programs as much as possible. We find that in the alternative programs, the number of lines of code is positively related to such indicators as Files, Functions, and code complexity, but not closely related to Percent Branch Statement and Percent Lines with Comments. Therefore, under the principle of meeting the diversity of the two indicators of Percent Branch Statement and Percent Lines with Comments, we selected two groups of tested programs with different scales to avoid the accidental impact of the tested programs on the experimental results as much as possible. Among them, the number of code lines for Grep and Flex is large, while the number of code lines for Printtokens and Schedule is small. Specific software metrics are shown in Table 3.

Table 3: Static metrics of the tested program

	Grep	Flex	Printtokens	Schedule
Files	46	51	3	1
Lines	50784	79200	725	412
Statements	13991	23185	366	207
Percent Branch Statement	24.1%	11.9%	26.0%	17.9%
Percent Lines with Comments	27.7%	6.8%	28.8%	20.6%
Functions	192	137	0	7
Average Statements per Function	28.6	36.4	0.0	4.7
Complexity of Most Complex Function	158	220	0	3
Maximum Block Depth	9+	8	5	5
Average Block Depth	2.30	0.94	1.72	1.48
Average Complexity	12.16	11.44	0.00	1.86

4.3 Compared algorithms

To test the effect of REM-WOA in solving TCP problems clearly, this paper selects two heuristic algorithms as comparative experiments: the artificial fish school algorithm and immune algorithm.

The specific reasons are as follows:

First of all, the ant colony algorithm does show excellent performance in the test case prioritization problem, but with more and more intelligent algorithms being applied in software engineering, the effectiveness of ACO is challenged. After testing, we found that the artificial fish school algorithm has better convergence characteristics and global search ability[22]. However, the artificial fish school algorithm has a strong random search ability and is more suitable for the search method based on n-dimensional directed space. These problems make AFSA effectively avoid falling into local optimal problems.

Secondly, we found that the performance of the immune algorithm in this field is significantly better than that of the genetic algorithm[16]. In theory, according to the comparative analysis of the characteristics of the two, genetic algorithm is dominated by cross operation, and its solving behavior is unitary and exclusive, while AI is dominated by mutation operation whose global search ability is stronger[42]. As shown in Figure 5, for the prioritization problem, small range of cross can not significantly improve the output results, but rearranging all test cases can effectively avoid falling into the local optimal problem.

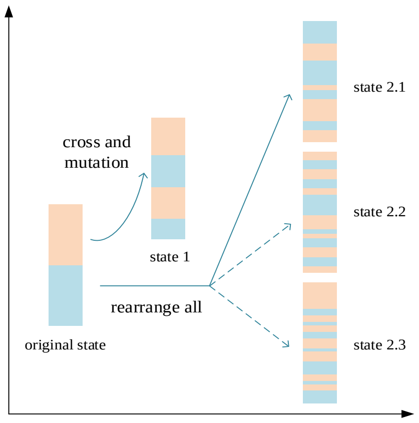

Figure 5: Cross mutation and rearrangement.

In addition, there will be different test requirements according to different test scenarios. For single-objective optimization, the objective can be directly used as the fitness function, which is more convenient to solve problems. For multi-objective optimization, Pareto optimality should be adopted when selecting solutions. During data preprocessing, elements in each row are rearranged in ascending order according to the maximum value of APTC, when elements in each column are rearranged in descending order according to the maximum value of EET.

4.4 Experimental parameters settings

The parameters used in this paper are the default values because previous studies have shown that using default parameters is also a reasonable choice[43, 44]. In addition, the parameter adjustment process of the algorithm is time-consuming and costly, and the results obtained in the TCP problem may not be better[36]. However, the side length of the square array is particular. Its value is set to maximize the use of computing memory. The backtracking size of the population was obtained through trial and error experiments. The specific parameter settings are shown in Table 4. The number of test cases randomly selected during prioritizing is 50. The experiments are conducted 30 times respectively, and the experimental data are recorded.

5 Analysis of experimental results

5.1 Effectiveness of REM-WOA

Before the formal TCP task study, we first researched the effectiveness of different WOA variants. The WOA with weighted search strategy is CAW-WOA, the WOA with three search strategies is

Table 4: Experimental parameter settings

	REM-WOA	AFSA	IA

Algorithm parameters	b=1	Visual range=0.8
		Visual attenuation=0.98
		Congestion threshold=0.66
			a=0.6
			b=0.4
			Variation rate=0.6

Population size	50		-
Number of iteration	100		-
Side length of square array	300		-
Population backtracking	80		-

PR-WOA, and REM-WOA is a combination of the two.

Take the single-objective optimization algorithm as an example and use the four benchmark functions as the fitness. Compare the optimization effects of WOA, PR-WOA, CAW-WOA and REM-WOA. Draw the changes in the fitness of the four algorithms with the number of iterations.

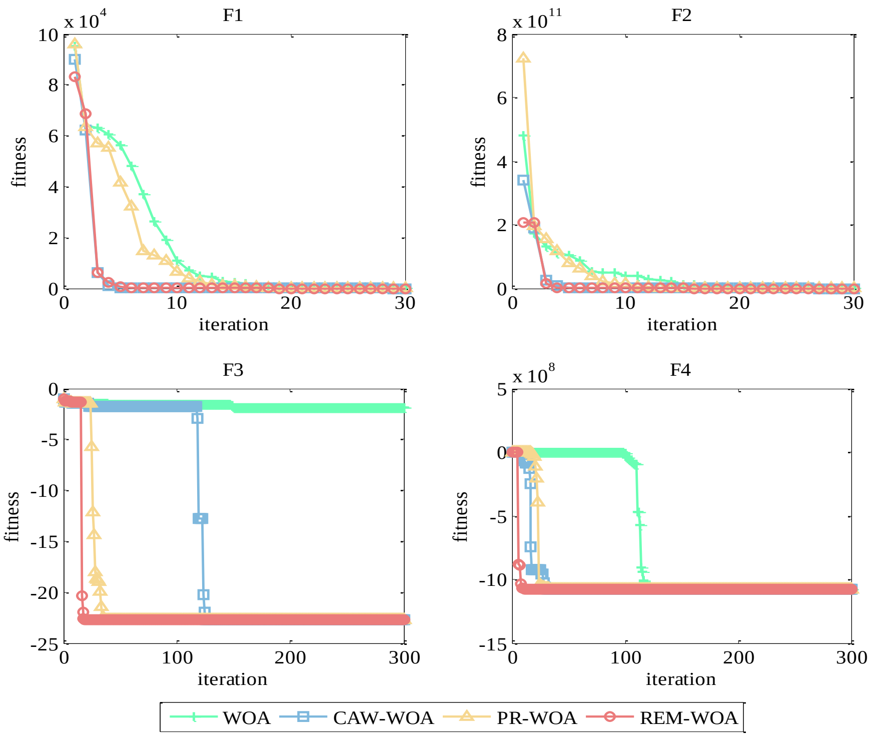

Figure 6: Convergence curves of different algorithms.

Figure 6 shows the convergence of four kinds of algorithms for different benchmark functions. It can be seen from the figure that the convergence speed of the conventional WOA is the slowest, and the fitness obtained by using the benchmark function F3 falls into the local optimization, which is far from reaching the optimization effect. PR-WOA is better than WOA and has been able to get rid of Lo problems. However, it has little impact on improving the convergence speed because the essence of the algorithm is to increase the population number and redistribute in space without improving the algorithm itself. The effect of CAW-WOA is obviously better than that of WOA and PR-WOA. It has not only fast convergence speed but also strong optimization ability. REM-WOA combines the advantages of PR-WOA and CAW-WOA. Compared with the other three algorithms, REM-WOA has the fastest convergence speed and will not fall into LO. Therefore, REM-WOA is selected in the subsequent test case prioritization.

5.2 Test case prioritization based on single objective

We take APTC and EET as the objectives, summarize the results of 30 experiments, and draw violin figures as shown in Figure 7 and Figure 8. It can be seen that when the improved whale optimization algorithm is applied to the single-objective test case prioritization, its output results are more concentrated, and has no abnormal values appear. First, except the median EET of the immune algorithm is better when executing Flex, the median APTC output by REM-WOA is slightly higher than the other two algorithms. And the median EET output is slightly lower than the other two algorithms. Second, there are two abnormal data in the APTC value output by the AFSA after executing Printtokens. In addition, in most cases, REM-WOA is more concentrated in the box graph, and there is little difference in the outputs of multiple experiments. This shows that REM-WOA runs more stably after reasonable modeling, and the final solution is closer to the real optimal solution.

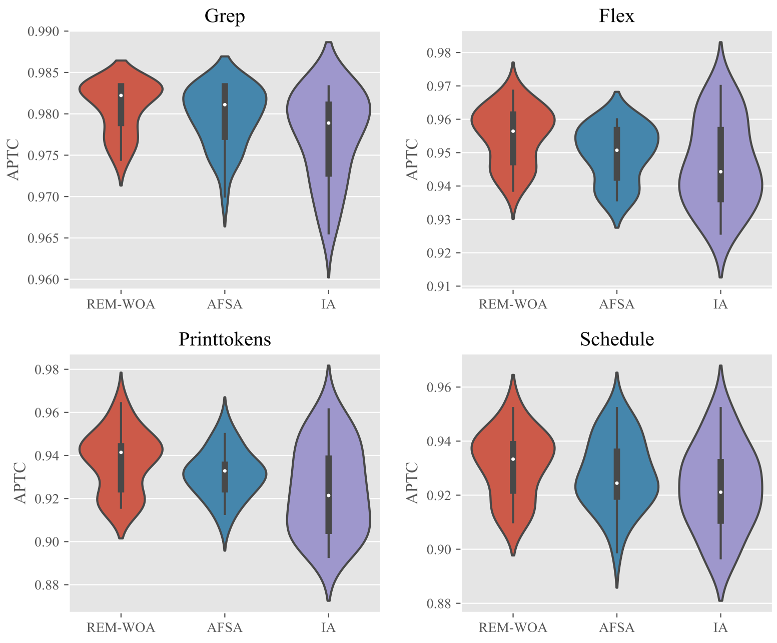

Figure 7: Single-objective comparison diagram of APTC.

We know that REM-WOA has a high efficiency when dealing with a single-objective task. However, we still need to research the stability of the model and the efficiency of the algorithm. As shown in Table 5, the performance of Grep is the best within 5 output results in 30 experiments. In the other 3 programs, different results are output. When these programs reaches the best solution, the average number of iterations varies from 23 to 65. The average value of APTC is similar to the median. In addition, when seeking the optimal value of EET alone, the output of the Grep program is also the most concentrated, but the advantage is not obvious compared with other experiments. When the test case prioritization of these programs reaches the best solution, the average number of iterations is about 40.

To sum up, REM-WOA sacrifices the algorithm's complexity, performs well in the single-objective TCP, and its output is closer to the real optimal solution. In addition, the convergence process of the model is fast, and the output can reach its steady state without too many iterations. However, there are some differences in the number of iterations when reaching the best solution for. Table 6 shows the relationship between the number of iterations of REM-WOA and EET. The average value is about 40, which also shows excellent algorithm stability.

Table 5: Optimal APTC and iteration times

Num	Grep		Flex		Printtokens		Schedule	
	APTC	iteration	APTC	iteration	APTC	iteration	APTC	iteration
1	0.9833	45	0.9628	64	0.9357	27	0.9300	57
2	0.9833	96	0.9542	82	0.9500	28	0.9388	76
3	0.9833	28	0.9585	22	0.9242	10	0.9344	16
4	0.9833	29	0.9585	32	0.9471	54	0.9433	51
5	0.9833	79	0.9542	1	0.9585	83	0.9300	32
6	0.9833	2	0.9557	35	0.9442	94	0.9100	1
7	0.9788	23	0.9414	7	0.9157	7	0.9388	72
8	0.9833	4	0.9457	7	0.9442	23	0.9300	45
9	0.9766	1	0.9557	62	0.9242	32	0.9388	65
10	0.9833	20	0.9442	4	0.9414	59	0.9433	91
11	0.9788	1	0.9628	6	0.9185	69	0.9411	67
12	0.9811	3	0.9600	10	0.9585	1	0.9211	1
13	0.9833	14	0.9442	23	0.9157	14	0.9188	50
14	0.9811	44	0.9600	32	0.9328	1	0.9433	1
15	0.9766	68	09428	79	0.9500	6	0.9144	67
16	0.9833	18	0.9514	63	0.9185	34	0.9522	73
17	0.9744	11	0.9471	24	0.9385	92	0.9388	29
18	0.9788	2	0.9685	23	0.9414	27	0.9522	3
19	0.9811	1	0.9614	88	0.9185	16	0.9122	30
20	0.9766	45	0.9385	14	0.9442	89	0.9166	16
21	0.9833	1	0.9571	13	0.9385	14	0.9388	41
22	0.9744	7	0.9614	14	0.9642	1	0.9300	94
23	0.9811	21	0.9628	94	0.9185	84	0.9100	7
24	0.9833	6	0.9481	19	0.9500	88	0.9277	23
25	0.9766	1	0.9557	54	0.9414	95	0.9100	20
26	0.9833	2	0.9657	1	0.9442	49	0.9344	86
27	0.9766	11	0.9628	63	0.9414	90	0.9388	26
28	0.9833	93	0.9457	12	0.9442	15	0.9144	65
29	0.9833	3	0.9600	90	0.9357	10	0.9322	10
30	0.9811	19	0.9414	33	0.9157	1	0.9233	33
mean	0.9807	23	0.9542	35	0.9371	65	0.9302	41

5.3 Test case prioritization based on multi-objective

The artificial fish school algorithm and immune algorithm are also selected for comparison in multi-objective prioritization. The parameter selection of the three algorithms is the same as that of

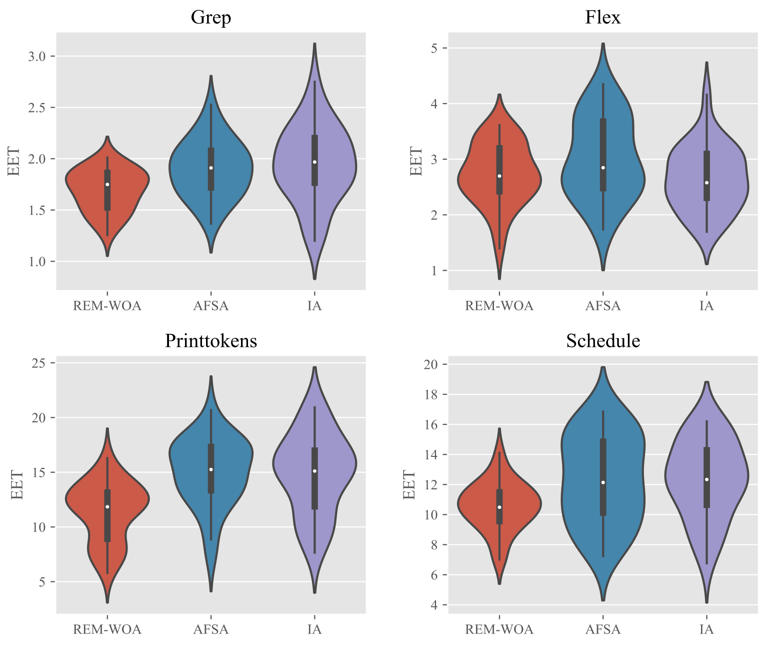

Figure 8: Single-objective comparison diagram of EET.

Table 6: Optimal EET and iteration times

Num	Grep		Flex		Printtokens		Schedule	
	EET	iteration	EET	iteration	EET	iteration	EET	iteration
1	1.5609	12	2.4682	31	14.2876	18	10.4308	64
2	1.8371	8	2.4243	15	11.7309	9	10.6100	37
3	1.8741	51	3.0433	28	7.9485	3	11.4762	25
4	1.4187	94	2.0968	89	15.0273	73	14.1090	1
5	1.6032	68	2.5865	51	13.2166	87	8.5437	87
6	1.7547	64	3.3033	94	12.9345	72	10.3571	31
7	1.7766	44	2.8082	64	7.2060	38	9.4584	44
8	1.8866	36	3.4880	66	13.6794	21	8.3123	25
9	1.8755	50	2.4089	11	11.8613	56	10.4858	41
10	1.3640	20	2,6650	14	8.6725	21	10.4916	85
11	1.2555	1	2.6724	4	9.6084	21	9.2608	82
12	1.6374	38	2.5339	98	8.6374	47	10.9394	52
13	1.7547	9	2.8920	21	12.0604	3	9.9642	46
14	1.9146	94	2.3389	25	11.6528	24	11.4702	40
15	1.7470	73	1.3953	78	5.7971	11	11.9914	76
16	1.5117	7	2.4198	93	13.5992	13	10.4778	41
17	1.6069	82	2.9156	69	7.8503	78	7.0093	19
18	2.0111	12	2.9297	39	12.8290	85	11.9372	61
19	1.6512	62	2.0528	10	14.2332	79	13.2129	33
20	1.6849	37	3.2483	1	12.7407	43	7.2387	7
21	1.8704	89	3.6133	23	12.9655	36	12.5916	72
22	1.5029	30	3.4393	35	13.7013	11	10.2217	29
23	1.7510	11	2.7667	25	16.2932	34	11.0732	12
24	1.3763	57	1.7216	1	12.3183	21	11.8723	69
25	1.9199	47	3.3919	41	11.6285	73	11.2386	25
26	1.8181	42	2.5625	1	6.9967	94	11.9184	60
27	1.8407	42	2.7219	49	11.8222	16	9.3439	48
28	1.4280	14	3.3496	63	11.3882	3	9.3513	6
29	1.7600	21	2.1809	77	11.5110	84	10.9822	11
30	1.8704	48	3.4254	89	7.6206	70	12.5483	82
mean	1.6954	42	2.7287	43	11.3939	41	10.6269	43

single-objective optimization. However, because it is a multi-objective solution, the Pareto optimal solution should be taken when comparing the fitness of individuals. The final result is not necessarily a single solution but a series of solution sets. Taking APTC as the abscissa and EET as the ordinate, the above three algorithms were applied to four programs for some experiments. Record the experimental data and draw the corresponding scatter chart. The closer the point is inclined to the lower right corner, the better the effect is.

It can be seen from Figure 9 that REM-WOA performs better than other algorithms in prioritizing test cases of Flex and Printtokens. While in Grep and Schedule, REM-WOA has obvious advantages over IA. However, compared with AFSA, REM-WOA has no apparent benefits. The latter is slightly better than the former in convergence and optimization objective value.

Finally, to further compare REM-WOA and AFSA, we counted the number of iterations when they reached their best position for the first time, and drew the corresponding three-dimensional scatter diagram. It can be seen from Figure 10 that the iteration number distribution of REM-WOA when finding the best solution is more uniform, which shows its advantages of convergence. However, the number of iterations of AFSA is concentrated in the final stage of the iteration, which indicates that the stability process is slow.

6 Case study of real-world application

To better evaluate the n-dimensional space search and REM-WOA, we selected the real-world project for further testing. The project is the code developed by Phytium for the gateway protocol, which uses C++ to exchange routing information between gateways. In this paper, the immune algorithm and artificial fish school algorithm are still used to test and analyze single-objective and multiple-objective tasks, respectively.

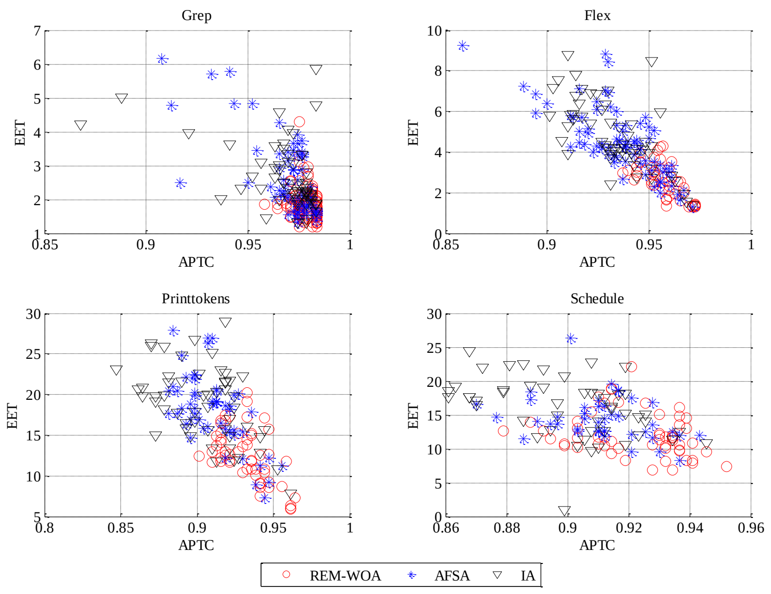

Figure 9: Two-dimensional scatter diagram of multi-objective optimization.

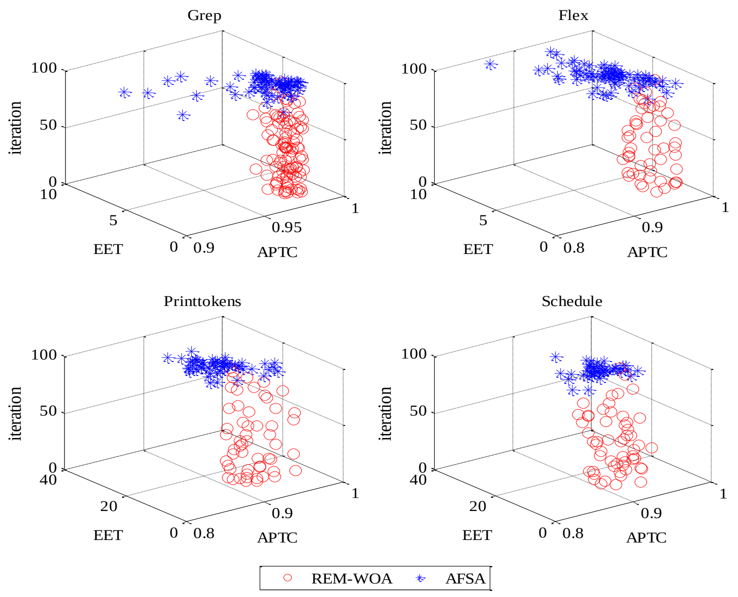

Figure 10: Three-dimensional scatter plot of multi-objective optimization.

6.1 Test case prioritization based on single objective

Because there are many coverage points in the gateway protocol program, the side length of square array needs to be reduced appropriately due to the limitation of computing devices. Considering the memory of the computer, the side length is finally set to 220. The backtracking size of individuals in the population is set to 60. The other parameters are set unchanged, the experiment are repeated 30 times, and the statistical data are collated to draw a violin chart.

We take APTC and EET as the tested objectives respectively for single-objective optimization. More details are showed in Figure 11 and Figure 12. First, it can be seen from the figure that when APTC is used as the fitness function, the median obtained by using REM-WOA is relatively maximum, and its quartile is also the highest among the three algorithms. However, in terms of data concentration, AFSA performs best, and the algorithm effect is between REM-WOA and IA. According to our previous experiments, this is because its iteration takes a long time, and it will not search for a better solution in time. Next, when compared with the swarm intelligence algorithm, the immune algorithm has the worst performance, and its performance is far inferior to the former in terms of data concentration and fitness function.

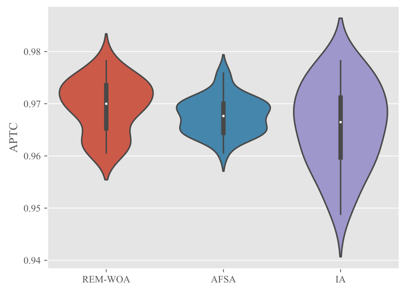

Figure 11: Comparison of APTC of real-world project.

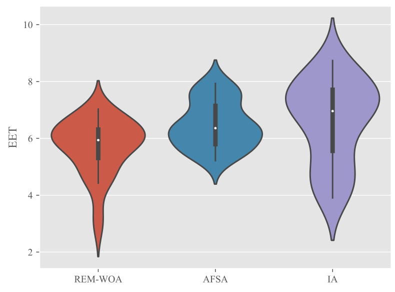

Figure 12: Comparison of EET of real-world project.

When EET is used as the fitness function, the median and quartile obtained by REM-WOA are the smallest. When comparing the concentration of the data distribution, although AFSA is still the best, REM-WOA gets some excellent results. Next, the improved whale optimization algorithm has the best data concentration. In addition, the worst effect is still the immune algorithm, which has the highest median distribution and the most dispersed data, but this also makes its quartile effect not the worst.

To sum up, REM-WOA based on n-dimensional directed space search has the best convergence speed and global search ability when dealing with TCP. AFSA has a weak effectiveness caused by the complex search mechanism. As for the immune algorithm, its effectiveness is far less than that of the swarm intelligence algorithm using n-dimensional directed space search. Because IA has no population and cannot conduct thorough searches. The method based on partial exchange mutation is difficult to solve the problem of high complexity, and the initial state has a significant impact on its final results. In addition, it is also true that the data obtained by IA are seriously scattered.

6.2 Test case prioritization based on multi-objective

We prioritize the multi-objective test cases for Phytium's gateway protocol, and the objectives functions are still APTC and EET. However, through the analysis of algorithm complexity, it can be seen that there is a specific correlation between complexity and performance effect. We repeated the experiment 30 times, and drew a three-dimensional scatter plot as shown in Figure 13.

From the horizontal plane, the REM-WOA distribution is more inclined to on the right, which means that the prioritization effect is better. Second, the data of AFSA are almost on the right in the figure, but they are distributed almost on the top. Similarly, when dealing with multi-objective problems, the immune algorithm has the worst performance and the most decentralized data distribution. In terms of the number of iterations, the improved whale optimization algorithm ranges from 0 to 100, and the immune algorithm has a small number of iterations. However, the final number of

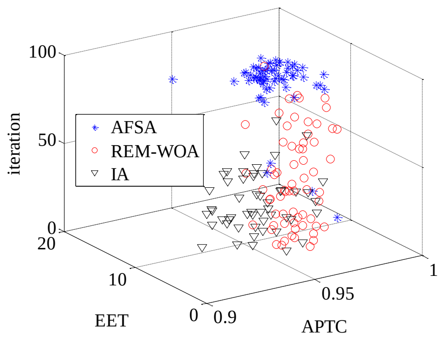

Figure 13: Comparison of multi-objective prioritization of real-world project.

iterations of the artificial fish school algorithm is mostly between 90 and 100. In general, the swarm intelligence algorithm has a high complexity and a relatively high number of iterations, but the effect is better. For the algorithm without population, the algorithm's performance is slightly poor, and the data distribution is not centralized.

7 Conclusion and future work

To reduce the cost of software testing, this paper takes software testing as an example to improve the decision-making method of prioritization for digital systems. We introduce the improved whale optimization algorithm and apply it to test case prioritization. Before using this algorithm, we established an n-dimensional spatial search model and improved the optimization method by processing the global data. The experiment shows that the whale algorithm with an enhanced exploration mechanism has more advantages than other variants. To achieve good results in TCP tasks, the optimization algorithm used should have good global search ability, and it can effectively avoid falling into global optimization. In addition, the performance of the algorithm has different effects on its efficiency. If the search mechanism of the algorithm is added to improve the prioritization effect, it may reduce the efficiency of the model and increase the number of iterations.

In the next stage, we will try other optimization algorithms, integrate the improved whale optimization algorithm with the deterministic algorithm, and avoid the potential problems implied by the fast convergence of the whale optimization algorithm. In addition, in the future, we will further deal with the search space for test case prioritization, making the optimization process of the swarm intelligent algorithm more accurate, and further reducing the operating cost of the digital system.

Author contributions

Bin Yang: Investigation, Methodology, Project Administration, Data Pre-processing, WritingReviewing and Editing.

Huilai Li: Conceptualization, Methodology, Experiment, Software, Visualization, WritingOriginal draft preparation.

Ying Xing: Investigation, Supervision, Methodology, Validation, Resources, Writing- Reviewing and Editing.

Fuping Zeng: Supervision, Resources, Writing- Reviewing.

Chengdong Qian, Youzhi Shen, Jiongbo Wang: Investigation, Supervision, Dataset provision.

Conflict of interest

The authors declare no conflict of interest.

References

[1] Minimol, A.J. (2021). Automating and optimizing software testing using artificial intelligence techniques, In *International Journal of Advanced Computer Science and Applications*, DOI: http://dx.doi.org/10.14569/IJACSA.2021.0120571, 12(05), 2021.

[2] Chen, Y. (2022). Analysis and application of software automated testing method, In *Modern Industrial Economy and Informatization*, 12(01), 167–168+171, 2022.

[3] Mahdieh, M.; Mirian, H.S.H.; Mahdieh, M. (2022). Test case prioritization using test case diversification and fault-proneness estimations, In *Automated Software Engineering*, 29(02), 50, 2022.

[4] Rongcun, W.; Zhengmin, L.; Shujuan, J.; *et al* (2020). Regression test case prioritization based on fixed size candidate set ART algorithm, In *International Journal of Software Engineering and Knowledge Engineering*, 30(03), 291–320, 2020.

[5] Lee, Y.S. (2022). A study on intermediate code generation for security weakness analysis of smart contract chaincode, In *Journal of Logistics, Informatics and Service Science*, 9(1), 53–67, 2022.

[6] Kim, J.A. (2022). A case study of domain engineering in software product line engineering, In *Journal of Logistics, Informatics and Service Science*, 9(1), 97–115, 2022.

[7] Wong, W.; Horgan, J.; London, S.; *et al* (1997). A study of effective regression testing in practice, In *The Eighth International Symposium On Software Reliability Engineering*, 264–274, 1997.

[8] Ani, R.; Sabrina, A.; Intan, E.A.J.; *et al* (2021). A systematic literature review on regression test case prioritization, In *International Journal of Advanced Computer Science and Applications*, 12(9), 253–267, 2021.

[9] Geetha, U; Sankar, S.; Sandhya, M. (2021). Acceptance testing based test case prioritization, In *Cogent Engineering*, 8(1), 1–22, 2021.

[10] Elbaum, S.; Malishevsky, A. G.; Rothermel, G. (2002). Test case prioritization: A family of empirical studies, In *Transactions on software engineering*, 28(2), 159-182, 2002.

[11] Haiyan, Z.; Hui, F.; Qingsong, X.; *et al* (2008). Research on test case prioritization, In *Computer Engineering and Science*, 1(01), 79–81, 2008.

[12] Li, Z.; Harman, M.; Hierons, R. (2007). Search algorithms for regression test case prioritization, In *Transactions on Software Engineering*, 33(4), 225-237, 2007.

[13] Dharmveer, K.Y.; Sandip, D. (2021). Test case prioritization based on early fault detection technique, In *Recent Advances in Computer Science and Communications*, 14(1), 302–316, 2021.

[14] Stanujkic, D.; Karabasevic, D.; Popovic, G.; *et al* (2021). Multiple-criteria decision-making based on the use of single-valued neutrosophic sets and similarity measures, In *Economic Computation And Economic Cybernetics Studies And Research*, 55(2), 5–22, 2021.

[15] Weixiang, Z.; Bo, W.; Huisen D. (2015). Test case prioritization method based on genetic algorithm, In *Minicomputer System*, 36(09), 1998–2002, 2015.

[16] Hongwei, X.; Pengcheng, L.; Zhongxiao C.; *et al* (2021). Test case prioritization based on artificial immune algorithm, In *Tehnički vjesnik*, 28(6) , 1871–1876, 2021.

[17] Xue L.; Yunna, T.; Yuan, T. (2021). A survey of swarm intelligence methods, In *Information and Computer (theoretical version)*, 33(24), 63–69, 2021.

[18] Tole, K.; Milani, M.; Mwakondo, F. (2021). Particle swarm algorithm for improved handling of the mirrored traveling tournament problem, In *Tehnički vjesnik*, 28(5), 1647–1653, 2021.

[19] Liu, L.; Chen, T.; Gao, S.; *et al* (2021). Optimization of agricultural machinery allocation in heilongjiang reclamation area based on particle swarm optimization algorithm, In *Tehnički vjesnik*, 28(6), 1885–1893, 2021.

[20] Sabonchi, A.K.S.; Akay, B. (2021). Cryptanalysis of polyalphabetic cipher using differential evolution algorithm, In *Tehnicki vjesnik-Technical Gazette*, 27(4), 1101–1107, 2021.

[21] Vescan, A.; Pintea, C.M.; Pop, P.C. (2022). Test case prioritization-ANT algorithm with faults severity, In *Logic Journal of the IGPL*, 30(2), 277–288, 2022.

[22] Ying, X.; Xingde, W.; Shen, Q. (2021). Test case prioritization based on artificial fish school algorithm, In *Computer Communications*, 180, 295–302, 2021.

[23] Gouda, R.; Chandraprakash, V. (2022). Multi-objective crow search and fruit fly optimization for combinatorial test case prioritization, In *International Journal of Software Innovation*, 9(4), 1–19, 2022.

[24] Anu, B.; Om, P. S. (2021). Test case prioritization using bat algorithm, In *Recent Advances in Computer Science and Communications*, 14(2), 593–598, 2021.

[25] Manar, A.H.; Abdelzahir, A.; Souad, L.M.S.; *et al* (2022). Modified Harris hawks optimization based test case prioritization for software testing, In *Computers, Materials & Continua*, 72(1), 1951–1965, 2022.

[26] Bajaj, A.; Sangwan, O.P. (2021). Discrete cuckoo search algorithms for test case prioritization, In *Applied Soft Computing Journal*, 110, 1–18, 2021.

[27] Gandomi, A.H.; Yang, X.S. (2011). Benchmark problems in structural optimization, In *Computation Optimization,Methods and Algorithms*, 356, 259–281, 2011.

[28] Singh, S.; Bansal, J.C. (2022). Mutation-driven grey wolf optimizer with modified search mechanism, In *Expert Systems With Applications*, 194, 1–24, 2022.

[29] Chunyao, L.; GuangLin, Z. (2021). A hybrid whale optimization algorithm for global optimization, In *Mathematics*, 9(13), 1477, 2021.

[30] Kezhong, L.; Zongmin, M. (2021). A modified whale optimization algorithm for parameter estimation of software reliability growth models, In *Journal of Algorithms & Computational Technology*, 15, 1–14, 2021.

[31] Jianxun, L.; Jinfei, S.; Fei, H.; *et al* (2022). A reinforced exploration mechanism whale optimization algorithm for continuous optimization problems, In *Mathematics and Computers in Simulation*, 201,23–48, 2022.

[32] Rothermel, G.; Untch, R.H.; Chu. C.; *et al* (1999). Test Case Prioritization, An Empirical Study, In *International Conference on Software Maintenance*, IEEE, 179–188, 1999.

[33] Muhammad, H.; Seung, R.J.; Muhammad, F.P.; *et al* (2021). An ontology based test case prioritization approach in regression testing, In *Computers, Materials & Continua*, 67(1), 1051–1068, 2021.

[34] MohdShafie, M.L.; WanKadir, W.M.N.; Khatibsyarbini, M.; *et al* (2020). Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion, In *PloS one*, 15(2), 1–27, 2020.

[35] Li, Z.; Harman, M.; Hierons, R.M. (2007). Search algorithms for regression test case prioritization, In *Transactions on Software Engineering*, 33(4), 225–237, 2007.

[36] Li F.; Zhou, J.; Li, Y.; Hao, d. (2021). AGA: An Accelerated Greedy Additional Algorithm for Test Case Prioritization, In *Transactions on Software Engineering*, 48(12), 5102–5119, 2021.

[37] Tizhoosh, H.R. (2005). Opposition-based learning, a new scheme for machine intelligence, In *Int. Conf. Comput. Intell. Model*, 1, 695-701, 2005.

[38] Jianxun, L.; Shi, J.; Hao, F.; *et al* (2022). A novel enhanced global exploration whale optimization algorithm based on Lévy flights and judgment mechanism for global continuous optimization problems, In *Engineering with Computers*, 2022.

[39] Huiling, C.; Chenjun, Y.; Ali, A.H.; *et al* (2020). An efficient double adaptive random spare reinforced whale optimization algorithm, In *Expert Syst*, 154, 113-118, 2020.

[40] Awad, N.H.; Ali, M.Z.; Suganthan P.N.; *et al* (2016). Problem definitions and evaluation criteria, In *The CEC 2017 Special Session and Competition on single-objective Real-Parameter Numerical Optimization*, 1–34, 2016.

[41] Software-artifact Infrastructure Repository. [Online]. Available, sir.csc.ncsu.edu/content/sir.php.

[42] Hong, G. (2003). Comparison of immune algorithm with genetic algorithm, In *Journal of Jinan University(Natural Science & Medicine Edition)*, 1(01), 22–25, 2003.

[43] Arcuri, A.; Fraser, G. (2013). Parameter tuning or default values? An empirical investigation in search-based software engineering, In *Empirical Software Engineering*, 13(8), 594–623, 2013.

[44] Sayyad, A.S.; Goseva-Popstojanova, K.; Menzies, T. (2013). On parameter tuning in search based software engineering, A replicated empirical study, In *2013 3rd International Workshop on Replication in Empirical Software Engineering Research*, 84–90, 2013.

[45] Dario, D.N.; Annibale, P.; Andy, Z.; Andrea, D.L. (2020). A Test Case Prioritization Genetic Algorithm Guided by the Hypervolume Indicator, In *Transactions on Software Engineering*, 46(6), 674–696, 2020.

Cite this paper as:

Bin Y., Huilai L., Ying X., Fuping Z., Chengdong Q., Youzhi S., Jiongbo W.. (2023). Directed Search Based on Improved Whale Optimization Algorithm for Test Case Prioritization, *International Journal of Computers Communications & Control*, 18(2), 5049, 2023.
https://doi.org/10.15837/ijccc.2023.2.5049