2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems

On Test Case Prioritization Using Ant Colony Optimization Algorithm

Weixiang ZHANG[1], Yuhua QI[1], Xuebo ZHANG[2], Bo WEI[1], Min ZHANG[1] and Zhaohui DOU[1]
1) Beijing Institute of Tracking and Telecommunications Technology, Beijing, 100094, China
2) Space Engineering University, Beijing, 101416, China
wxchung@msn.com

Abstract—**Test case prioritization technology improves the efficiency of software testing by optimizing the execution order of test cases, which is an important research topic of software regression testing. In order to solve the problem of requirement-based test case prioritization, this paper proposed a solution based on ant colony optimization algorithm and gave its two different implementation methods: distance-based and index-based implementation. Firstly, a general indicator based on requirements was designed to evaluate the test cases. Secondly, the concept of test case attractivity was proposed, and the definition of the distance between test cases was given based on it. Finally, the main design strategies such as the pheromone update strategy, the optimal solution update strategy, and the local optimal mutation strategy were given. The experimental results show that the method has good global optimization ability, and its overall effect is better than particle swarm optimization algorithm, genetic algorithm and random testing.**

Keywords- Software Testing; Test Case Prioritization; Ant Colony Algorithm; Black Box Testing; Regression Testing

I. INTRODUCTION

Effective maintenance of the original test case set (including expansion, reduction and sorting, optimization, etc.) is an important part of the software regression testing. One of the simplest maintenance strategies is to re-execute all existing test cases, but this often leads to problems such as project budget or project schedule not allowed, some test cases cannot be executed, and existing test cases cannot cover new test requirements. Therefore, a series of test case maintenance techniques have been proposed, including test case set expansion, test case set reduction, test case selection, failure test case identification and repair, test case prioritization technology and so on [1].

Test case prioritization (TCP) reorders test cases under the guidance of specific sorting criteria according to pre-selected test objectives, and improves test efficiency by optimizing the execution order of test cases. It is a hot research direction [3]. In large-scale software testing, it often takes weeks or even months for a test case to run completely. In this case, the tester wants to sort the test cases so that the higher priority can be executed as early as possible. Wong et al. [4] conducted related research earlier, combined with test case history coverage information and code modification information to identify redundant test cases, and ranked non-redundant test cases according to their ability to cover code. Elbaum et al. [5] gave a general description of the TCP problem in 2000. Rothermel et al. [6] have conducted a series of targeted experimental studies to demonstrate the effectiveness of TCP in improving error detection rates. Tonella et al. [7] proposed a machine learning method that uses the tester's experience to evaluate and rank the importance of test cases. Yoo et al. [8] introduced the Pareto efficiency method and combined multi-objective optimization to set test case priority. Chen Xiang et al [3] divided the TCP technology into three categories: code-based, model-based and requirement-based TCP. At present, the research of code-based TCP technology is quite sufficient, and the research results of TCP technology based on requirement are still rare.

The Ant Colony Algorithm (ACA) is proposed by Dorigo et al. [9] and is a bionic optimization algorithm that simulates the foraging behavior of ant colonies. ACA introduces a positive feedback parallel mechanism, which has many advantages: strong robustness, excellent distributed computer system, and easy integration with other methods [10]. ACA has successfully solved complex optimization problems in many fields and has become a research hotspot in the field of intelligent optimization. Dorigo et al. [11] proposed an Ant Colony System (ACS) based on the basic ant colony algorithm, which allows the path with the largest amount of information to be selected with a large probability, which can fully enhance the feedback ability of optimal information. Stützle et al. [12] proposed the MAX-MIN Ant Colony (MMAS), which limits the amount of information on each path to an interval to avoid premature convergence of the algorithm to the non-global optimal solution. Ant colony algorithm was first applied to the famous traveling salesman problem, and then widely used in many fields such as network routing, task scheduling, system identification and image processing [13][14][15][16].

Since the TCP problem is essentially a discrete optimization problem that finds the optimal order of test cases, it can be solved using ACA. Gu et al. [17] proposed a multi-objective optimization method based on ant colony algorithm. For the optimization goal of average sentence coverage and effective execution time, the optimal ordering of test case sets was realized and the evaluation of multi-objective solution set was improved. method. Xing et al. [18] proposed a pheromone update strategy based on the upper gene segment to improve the problem that the ant colony algorithm has a slow convergence rate and is easy to fall into the local optimum when solving the multi-objective test case prioritization.

The main contributions of this article are:

- A method for solving requirements-oriented TCP problem based on ACA (ACA for TCP, TCP-ACA) is proposed, and two different implementation methods are given, which can be directly applied to black box testing.
- The concept of test case attraction is proposed, and the distance between test cases is defined based on it,

978-1-7281-2058-4/19/$31.00 ©2019 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2019.00388

which is used as external heuristic information to guide the optimization of ant colony.

- A general evaluation index based on requirements is proposed to quantify the pros and cons of the test case sequence.
- An improved pheromone update strategy is introduced and a local optimal solution mutation strategy is introduced to improve the global optimal solution convergence speed and search ability.

Section 2 of this paper is a description of the TCP problem based on requirements, and gives a general evaluation of the sequence of test cases based on requirements. Section 3 introduces the basic ant colony algorithm, proposes the definition of the test case distance based on requirements, and gives the design and implementation of TCP-ACA. Section 4 is an experimental verification of TCP-ACA and an analysis of the verification results. Section 5 summarizes and looks ahead to the next step.

II. STATEMENT OF PROBLEM

A. Requirements-based Test Prioritization

The general description of the TCP problem is [5]: For a given set of test cases $\boldsymbol{T}$, its full-aligned set $\boldsymbol{PT}$ and sort objective function $f:\boldsymbol{PT}\to\boldsymbol{R}$, look for $\boldsymbol{T}'\in\boldsymbol{PT}$ such that for $\forall\boldsymbol{T}''\in\boldsymbol{PT}\left(\boldsymbol{T}''\neq\boldsymbol{T}'\right)$, there is $f\left(\boldsymbol{T}'\right)\geq f\left(\boldsymbol{T}''\right)$. As can be seen from the problem description, the input of the objective function f is the execution order of the test case, and the output is a value. The larger the value, the better the sorting effect.

According to the different sorting basis, TCP technology can be divided into three categories: code-based, model-based and requirement-based TCP technology [3]. The current research mainly focuses on code-based TCP technology, and has made many research results using techniques such as greed, machine learning, and fusion of expert knowledge.

The research results of requirement-based TCP technology are still relatively few. Qu et al. [19] proposed a set of demand-based test case priority dynamic adjustment algorithm based on the design information of test cases. Zhang et al. [20] proposed a TCP technology based on the Total strategy and the Additional strategy to consider the test demand priority and test case execution overhead. Krishnamoorthi et al. [21] based on the software requirements specification, considered more influencing factors when sorting test cases, including demand priority, demand change information, requirements implementation complexity, requirement integrity, demand traceability and defect impact.

Zhang et al [22] pointed out that the various factors affecting requirement-based TCP can be divided into two categories: test cost factors (Cost-Keys) and test income factors (Win-Keys). Some of these factors are positively correlated, some are negatively related, some have large influences, some have small influences, but they can all be normalized by weighting.

Without loss of generality, suppose the complete set of Cost-Keys factors is $\boldsymbol{C}=\{c_1,c_2,\mathrm{L}\ ,c_n\}$, and the complete set of Win-Keys factors is $\boldsymbol{W}=\{w_1,w_2,\mathrm{L}\ ,w_m\}$. For any test case $t_i\in\boldsymbol{T}$:

$$W_i=h(\boldsymbol{W}),\ C_i=g(\boldsymbol{C}) \tag{1}$$

In above, W_i is called the comprehensive benefit of test case t_i, which indicates the comprehensive impact of all test income factors on test case t_i. C_i is called the comprehensive cost of test case t_i, indicating the impact of all test cost factors on t_i. $h:\boldsymbol{W}\to\boldsymbol{R}$ and $g:\boldsymbol{C}\to\boldsymbol{R}$ are the weight functions of the test income factor and the test cost factor. Note that we extend the h h and g in Zhang et al. [22] so that h,g here can be not only linear but also nonlinear functions.

B. Test Case Sequence Evaluation Indicators

In order to demonstrate the effectiveness of TCP technology, it is necessary to evaluate its ranking results. The advantage of TCP over random-sequence testing is the ability to check for errors faster. To this end, Elbaum et al. [23] used the relationship between the number of test cases and the number of detection errors to quantify the pros and cons of the test case sequence, and gave the APFD (average percentage of fault detection) evaluation index. Since the defect detection information of the test case cannot be known before the test case is fully executed, the APFD has obvious disadvantages. Li et al. [24] then proposed the APBC (average percentage of block coverage), APDC (average percentage of decision coverage) and APSC (average percentage of statement coverage) based on the coverage rate of blocks, branches and sentences. But obviously these indicators are more suitable for code-based structural testing.

In requirement-based functional testing, testers are based on software specifications. Generally, the tester first converts the software requirements into test requirements, then decomposes the test requirements into test points, and finally designs the test cases for the test points to form a test case set. To this end, Zhang Weixiang et al [25] proposed an average percentage of test-point coverage (APTC). For the test case set $\Phi=\{T_1,T_2,\mathrm{L}\ ,T_m\}$, the APTC calculation formula is defined as [25]:

$$APTC=1-\frac{TT_1+TT_2+\mathrm{L}\ +TT_m}{nm}+\frac{1}{2n} \tag{2}$$

Where n is the number of test cases, m is the number of test points, and TT_i is the order in which the first test case that can cover the i-th test point is in the use case sequence. The value range of APTC is between 0 and 100%. The higher the value, the faster the coverage rate of the test point.

Similar to [22], the evaluation target is adjusted to the comprehensive income obtained by the unit comprehensive cost, and the APTC is transformed by the comprehensive income and the comprehensive cost in the formula (1), so we propose the evaluation index eAPWC (enhanced average

percentage of win- Cost coverage). eAPWC is formally consistent with APWC [22] and is formulated as:

$$eAPWC = \frac{\sum_{i=1}^{m}\left(W_i \times \left(\sum_{j=TT_i}^{n} C_j - \frac{1}{2}C_{TT_i}\right)\right)}{\sum_{j=1}^{n} C_j \times \sum_{i=1}^{m} W_i} \tag{3}$$

Among them, the meaning of each variable is the same as the corresponding variable in formula (1) and formula (2). Formula (3) is formula (2) when the combined returns of all test cases are the same and the combined costs are equal. That is, the APTC indicator is a special case of the eAPWC indicator.

III. SOLUTION BASED ON ANT COLONY ALGORITHM

This section presents a method for solving TCP problems based on ant colony algorithm (ACA for TCP, TCP-ACA). First introduce the basic ant colony algorithm, then give the design and implementation of TCP-ACA and its basic flow.

A. Basic Ant Colony Algorithm

The basic ant colony algorithm is a method that uses the artificial ant's walking route to represent a feasible solution to the problem to be solved. The whole ant colony is concentrated on the route representing the optimal solution under the action of positive feedback, and the optimal solution is found [26].

The basic ant colony algorithm is introduced in the context of Traveling Salesman Problem (TSP). Given n cities $\boldsymbol{C} = \{c_1, c_2, \mathrm{L}\ , c_n\}$ and the length $d_{ij}\ (1 \le i \le n, 1 \le j \le n, i \ne j)$ of the travel path between them. The TSP problem is to find the shortest loop that passes through each city once and returns to the starting point. Let the probability that ant k located in city i at time t choose city j as the next target is $p_{ij}^k(t)$, there is:

$$p_{ij}^k(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{l \in N_i^k(t)}\left[\tau_{il}(t)\right]^{\alpha}\left[\eta_{il}(t)\right]^{\beta}}, & j \in N_i^k(t); \\ 0, & j \notin N_i^k(t). \end{cases} \tag{4}$$

Where $\tau_{ij}(t)$ is the heuristic information (pheromone concentration) provided by the ant colony; $\eta_{ij}(t)$ is the heuristic information (external heuristic or a priori information) provided by the problem itself, usually $\frac{1}{d_{ij}}$, indicating that the path with a short local distance is superior; α, β are parameters (generally constant), respectively reflecting the degree of influence of pheromone and foreign heuristic information; $N_i^k(t)$ represents the set of the next city where ant k in city i is allowed to access at time t.

In order to avoid flooding the heuristic information due to excessive residual information, the pheromone $\tau_{ij}(t)$ is updated after each ant has completed one step or completed traversal of all cities:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{5}$$

Where, $\Delta\tau_{ij}(t) = \sum_{k=1}^{m}\Delta\tau_{ij}^k(t)$, $\tau_{ij}^k(t)$ represents the amount of pheromone released by ant k on path $(c_j - c_i)$ at time t, m is the number of ants in the ant colony; ρ is the pheromone volatilization coefficient, $1 \le \rho \le 1$. $\Delta\tau_{ij}^k(t)$ is defined in the Ant-Cycle Model as follows:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k}, & \text{If ant k passes the path } (c_j - c_i); \\ 0, & \text{else.} \end{cases} \tag{6}$$

Where L^k is the sum of the path lengths of ants k traversing all cities; Q is a constant.

For solving the TCP problem, the basic ant colony algorithm has natural adaptability, but it needs to be modified to overcome the shortcomings such as being sensitive to parameters and easy to fall into local optimum.

B. Requirements-based Test Case Distance

Definition 1 (test case attraction): For any two test cases c_i, c_j, the set of test points covered is $\boldsymbol{TP}_i, \boldsymbol{TP}_j$ respectively. Define the attractiveness $aspiration_{ij}$ of c_i and c_j as:

$$aspiration_{ij} = \lambda \times \frac{\sum_{p \in \boldsymbol{TP}_i \cup \boldsymbol{TP}_j} v_p - \mu \times \sum_{q \in \boldsymbol{TP}_i \cap \boldsymbol{TP}_j} v_q}{C_i + C_j} \tag{7}$$

Where v_p, v_q represents the value of the test point (importance), C_i, C_j represent the combined cost of the test case (as defined by equation (1)), and λ, μ are the adjustment factor.

By definition, the degree of attraction of the two use cases is positively correlated with the total value of the test points they collectively cover, and negatively correlated with the overall cost of the test case. In particular, the higher the similarity of the test points covered by the two use cases, the lower the attractiveness.

Definition 2 (test case distance): For any two test cases c_i, c_j, define the distance between them d_{ij} (where ε is a small constant, such as 10^{-3}) as:

$$d_{ij} = \begin{cases} \dfrac{1}{aspiration_{ij}}, & aspiration_{ij} > 0; \\ \varepsilon, & aspiration_{ij} = 0. \end{cases} \tag{8}$$

It can be seen that the more test points that are covered, the higher the value and the lower the similarity, the smaller the distance between the two test cases; otherwise, the greater the distance. This is consistent with the goal of solving the TCP problem.

C. TCP-ACA Design and Implementation

According to different evaluation methods, TCP-ACA has two implementation methods:

Mode 1 (TCP-ACA-1): Define the distance between use cases (using equation (8)), and let the length of the test case sequence be the sum of the distances between the use cases on its path. According to the idea of solving the TSP problem, evaluate the test case sequence based on its length, and iteratively update.

Mode 2 (TCP-ACA-2): Do not define the distance between use cases (or assume that the distance between each use case is equal), and let the ant colony search for the path randomly. Use formula (3) as the index to evaluate the test case sequence, and add extra pheromone to the path corresponding to the local optimal solution, and then iteratively update.

Regardless of which mode is adopted, TCP-ACA improves the deconstruction strategy, solution evaluation strategy, and pheromone update strategy of the basic ant colony algorithm, and introduces a local optimal bounce strategy to ensure its applicability to TCP problems and improve its ability to search for global optimal solutions.

1) Solution Construction

For each ant in the ant colony, the solution is constructed separately. Unlike TSP, the first and last test cases in the sequence are not connected.

When solving, the ant first randomly selects a test case as the initial position, and then accesses the next untested test case one by one according to the transition probability. Until all the test cases are accessed, a feasible solution is obtained.

The transition probability is calculated according to formula (4). For ant k, test case i and j, the strategy of the largest and minimum ant system MMAS [12] can be used. $\tau_{ij}(t)$ indicates the pheromone concentration of the path between test case i and test case j, and is updated after each ant completes the traversal. The update strategy will be described in detail later. The amount of pheromone on each path is limited to interval $[\tau_{\min}, \tau_{\max}]$ and is initially set to the upper limit of the interval, $[\tau_{\min}, \tau_{\max}]$ is calculated by formula (9), where $L(s^{gb})$ is the length of the global optimal solution path (an empirical value before the first generation), σ is the number of elite ants (0 before the first generation).

$$\begin{aligned} \tau_{\max}(t) &= \frac{1}{2(1-\rho)} \cdot \frac{1}{L(s^{gb})} + \frac{\sigma}{L(s^{gb})} \\ \tau_{\min}(t) &= \frac{\tau_{\max}(t)}{20} \end{aligned} \tag{9}$$

$\eta_{ij}(t)$ in formula (4) is the outer heuristic information. For TCP-ACA-1, $\eta_{ij}(t) = \frac{1}{d_{ij}}$, d_{ij} is defined by formula (8); for TCP-ACA-2, $\eta_{ij}(t) = const$, $const$ is a constant (e.g. 1).

After calculating the transition probability, the next test case is selected according to the roulette strategy. After all the ants have completed a traversal, we get a set of solutions. The number of solutions in the set is equal to the number of ants.

2) Solution Evaluation

Each solution in the collection needs to be evaluated. The evaluation strategies of the two modes of TCP-ACA are different.

For TCP-ACA-1, first calculate the length of the solution. The length of the solution is the sum of the distances between adjacent use cases in the test case sequence. Then, the quality of the solution is evaluated based on the length of the solution. The smaller the length, the higher the quality. The smallest is the optimal solution.

For TCP-ACA-2, first calculate the fitness of the solution. The fitness of the solution is the comprehensive income obtained by the unit comprehensive cost of the test case sequence, which can be obtained by using the eAPWC indicator defined by formula (3). Then, the quality of the solution is evaluated based on the fitness of the solution. The higher the fitness, the higher the quality. The largest is the optimal solution.

It should be noted that eAPWC is a general indicator, which can be simplified by setting qualification conditions or selecting specific parameters. For example, if all test cases have the same comprehensive income and equal comprehensive cost, eAPWC is equivalent to APTC [25].

3) Optimal Solution Set Update

The role of the optimal solution set is to collect the current global optimal solution generated throughout the search process, which is updated once after each iteration.

In MMAS, ants that find one of the following two optimal paths are allowed to release pheromones (other ants are not allowed): the optimal path in this iteration, or the optimal path in all iterations that have occurred so far.

After each iteration, first, the merits of each solution are evaluated, the optimal path in this iteration is marked, and then the optimal path of this iteration is compared with the optimal path in the optimal solution set. Finally, update the optimal solution set.

4) Pheromone Update

The purpose of the pheromone update is to increase the concentration of pheromone on the better path, reduce the concentration of pheromone on the worse path, and guide the ant to search in the right direction. The update strategy for the two modes of TCP-ACA is different.

For TCP-ACA-1, a secondary update strategy is adopted: firstly, the MMAS pheromone update strategy is used for the first update, and then some long-distance sub-paths on the global optimal path are updated for the second time. specifically:

(1) First update. The formula (5) is modified using MMAS's pheromone update strategy, which has:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{b}(t) \tag{10}$$

Where A is as defined in formula (6). b represents an ant that finds one of two optimal paths: the optimal path in this iteration, or the optimal path in all iterations that have occurred so far. The ants corresponding to the two optimal paths should

be used in turn, and the rotation frequency can be changed with the number of iterations.

(2) Second update. Considering that the sub-paths in some optimal paths have a small probability of being selected due to its long distance, which may result in being missed and cannot constitute a potential global optimal solution. Therefore, after each iteration, the path contribution of each sub-path is calculated, and then a second update is made to the sub-path whose path contribution is larger than the given threshold. For sub-path $\left(c_j - c_i\right)$, define its path contribution [27] as $\frac{d_{ij}}{L\left(s^{gb}\right)}$. For a given threshold q_0, calculate the pheromone rate of change as follows:

$$\Delta\tau_{ij}^{'}(t)=\begin{cases}\dfrac{Q^{'}}{d_{ij}}, & \dfrac{d_{ij}}{L\left(s^{gb}\right)}>q_0;\\ 0, & \text{其它.}\end{cases} \tag{11}$$

For TCP-ACA-2, only the first update is performed, and no secondary update is performed.

5) Local Optimal Solution Mutation

The positive feedback mechanism of the ant colony algorithm is beneficial to the fast convergence of the algorithm, but it can also lead to the accumulation of pheromones on the better path, which makes the search fall into local optimum and stagnate. In order to improve the ability of the algorithm to jump out of local optimum, TCP-ACA introduces a mutation strategy, replacing the local optimal solution that tends to stagnate with the new solution generated by the mutation.

The local optimal mutation strategies of the two modes of TCP-ACA are the same. Specifically, assuming that the local optimal path (i.e., the current global optimal solution) is $c_1 \to c_2 \to \mathrm{L} \to c_i \to \mathrm{L} \to c_j \to \mathrm{L} \to c_n$, at first, randomly selecting two positions i and j on the path, exchanging test cases c_i and c_j thereon to generate a mutation, and forming a mutation solution; Then, the mutation solution is evaluated, and if it is better than the original path, the local optimal solution is updated; otherwise it is not updated. This cycle is repeated several times, which can enhance the diversity of paths and improve the overall optimization ability.

D. TCP-ACA Main Process

The main processes of the TCP-ACA algorithm include:

1) Initialization. Set the maximum number of iterations, pheromone volatilization coefficient, number of ants, number of test cases, current iterations, taboo table index and other variables. The ants are randomly placed on each test case as a starting point.

2) Construct a solution space. Perform the following steps for each ant:

a) calculating the selection probability of each test case according to the state transition probability formula;

b) moving the ant to the test case with the highest probability of transition;

c) Repeat a)~b) until the ants traverse all test cases.

3) Evaluation. The fitness values of the solutions generated by this iteration are calculated one by one, and the current optimal solution is obtained.

4) Perform a local optimal solution mutation. If the optimal solution does not improve over a period of time, it means that it falls into local optimum, and a local optimal solution mutation needs to be performed; otherwise, the next step is directly performed.

5) Update the optimal solution set. The optimal solution set is updated by using the current optimal solution.

6) Update the pheromone concentration. According to the pheromone update strategy, the first update and the second update is performed.

7) Determine whether to terminate. It is judged whether the termination condition such as the maximum number of iterations is satisfied, and if so, the algorithm terminates; otherwise, it returns to step 2).

IV. EXPERIMENTAL VERIFICATION AND ANALYSIS

A. Experimental Configuration

The experiment uses a triangle classification program and compares the experimental results with the existing results of [25], [22]. The main function of the triangle classification program is to use the length of each side of the triangle to determine the type of the triangle, including the equilateral triangle, the isosceles triangle, the equilateral triangle, and the non-triangle [25]. The triangle classification program consists of 7 test points and 6 test cases. The correspondence between test cases and test points is shown in Table 1.

TABLE I. CORRESPONDENCE BETWEEN TEST POINTS AND TEST CASES OF THE TRIANGLE CLASSIFICATION PROGRAM [25]

Test case			Test point ID / value						
ID	**name**	**cost**	**1/1**	**2/1**	**3/1**	**4/2**	**5/2**	**6/2**	**7/3**
A	NumErrorTest	1	X						
B	TriangleErrorTest	1		X					
C	ScalTriangleTest	2			X	X			
D	RightTriangleTest	2			X		X		
E	IsosTriangleTest	2			X			X	
F	EquiTriangleTest	3						X	X

Ant colony algorithm, particle swarm algorithm and genetic algorithm are all heuristic methods, and it is not guaranteed that the optimal solution can be obtained. In order to test the effect of the algorithm, this paper compares and analyzes the quality and success rate of the optimal solution.

For comparison purposes, two simplified variants of eAPWC, eAPWC-1 and eAPWC-2, are used as evaluation indicators for TCP-ACA-2. The simplification of eAPWC-1 is based on the assumption that the comprehensive benefits of all test cases are the same and the combined costs are equal. It is equivalent to APTC of [25] and APWC-1 of [22].

Under the situation that the importance of each test point is different and the cost of each test case is not equal, the simplification of eAPWC-2 is to evaluate the importance of test points covered by unit test cost as the evaluation aim, which is equivalent to APTC_C of [25] and APWC-2 of [22].

Referring to the method of [10], the main parameters of TCP-ACA are set after multiple adjustments as follows:

TCP-ACA-1: ant colony size m=10, information heuristic factor $\alpha = 1$, expected heuristic factor $\beta = 5$, pheromone volatilization coefficient $\rho = 0.1$, pheromone intensity $Q = 0.3$, maximum iteration number $N_{\max} = 100$.

TCP-ACA-2: m=10, $\alpha = 1$, $\beta = 5$, $\rho = 0.1$, $Q = 1$, $N_{\max} = 100$.

B. Analysis of Results

The experiment carried out two modes of TCP-ACA, TCP-ACA-1 and TCP-ACA-2, respectively, and TCP-ACA-2 used two evaluation indexes, eAPWC-1 and eAPWC-2, respectively. Therefore, there were three sets of experiments (eAPWC-1, eAPWC-2, and use case distance for solution evaluation), and each set of experiments was repeated 15 times.

The optimal solutions obtained by different methods such as TCP-ACA and TCP-DPSO [22], GA [25] are shown in Table 2, and the optimal solution success rate is shown in Table 3. TCP-DPSO [22] and GA [25] do not use the use case distance as an evaluation index.

TABLE II. COMPARISON OF THE OPTIMAL SOLUTION QUALITY

	TCP-ACA	TCP-DPSO[22]	GA[25]
eAPWC-1	C-F-B-A-D-E: 0.65476	C-F-B-A-D-E: 0.65476	C-F-B-A-D-E: 0.65476
eAPWC-2	F-D-C-B-A-E: 0.81061	F-D-C-B-A-E: 0.81061	F-D-C-B-A-E: 0.81061
Use case distance	B-C-F-D-A-E: 1.75	-	-

TABLE III. COMPARISON OF OPTIMAL SOLUTION SUCCESS RATES

		TCP-ACA	TCP-DPSO[22]	GA[25]
eAPWC-1	Number of times (optimal / total)	15/15	14/15	12/15
eAPWC-2	Number of times (optimal / total)	14/15	14/15	11/15
Use case distance	Number of times (optimal / total)	15/15	14/15	11/15

15 experiments were also performed using random testing, and 100 test case sequences were randomly generated for each experiment. The distribution of the solutions was as shown in Table 4, it is almost normal distribution, and the probability of the optimal solution is about 2% to 3%.

TABLE IV. DISTRIBUTION OF SOLUTIONS BY RANDOM TESTING

	eAPWC-1	eAPWC-2	Use case distance
Distribution of solutions	3%: 0.65476; 8%: [0.63095, 0.65476]; 33%: [0.58333, 0.65476]; 83%: [0.55952, 0.65476]; 98%: [0.53571, 0.65476]; 100%: [0.51190, 0.65476].	2%: 0.81061; 11%: [0.76136, 0.81061]; 33%: [0.70455, 0.81061]; 63%: [0.64394, 0.81061]; 92%: [0.57197, 0.81061]; 100%: [0.54167, 0.81061].	3%: 1.75; 15%: [1.75, 1.87202]; 34%: [1.75, 1.89286]; 64%: 1.75, 2.03869]; 90%: [1.75, 2.22890]; 100%: [1.75, 2.39556].

It can be seen from Table 2 and Table 3: from the quality of the optimal solution, TCP-ACA is much better than random testing, and is equivalent to TCP-DPSO and genetic algorithm; from the solution success rate of the optimal solution, TCP-ACA is better than the genetic algorithm and slightly better than TCP-DPSO. In general, the TCP-ACA method is feasible, has a good global optimization ability, and is superior in effect to particle swarm optimization, genetic algorithms, and random testing.

V. CONCLUSION

The test case prioritization problem is essentially a discrete combinatorial optimization problem. This paper proposes a new method, TCP-ACA, which utilizes the characteristics of robustness and strong parallelism of the ant colony algorithm to solve the requirement-based test case prioritization problem, which can be directly applied to software regression testing.

According to different solution evaluation strategies, this paper implements the proposed TCP-ACA method in two ways. TCP-ACA-1 proposes the concept of use case attraction, defines the distance between test cases, evaluates the test case sequence according to its length, and uses the optimal ant to update the pheromone to continuously iterate and optimize. TCP-ACA-2 does not define the distance between use cases, it evaluates the randomly generated test case sequence by using the eAPWC-1 or eAPWC-2 proposed in this paper, and then, by adding additional pheromone to the obtained optimal path, the purpose of iteration and optimization is achieved.

The experimental results show that the TCP-ACA method has a good global optimization ability, and the overall effect is better than particle swarm optimization algorithm, genetic algorithm and random test.

With the rapid development of artificial intelligence, the application of intelligent technology is increasingly widespread. Using intelligent optimization technology to solve hot problems in test cases such as test case prioritization, test cases and automatic generation of test data is a feasible technical direction and has important research value. Next, we will conduct further research on the application of various intelligent technologies in software testing.

REFERENCES

[1] Yoo S, Harman M. Regression testing minimization, selection and prioritization: A survey. Software Testing, Verification & Reliability, 2012, 22(2): 67-120.

[2] Chen X, Chen JH, Ju XL, Gu Q. Survey of test case prioritization techniques for regression testing. Journal of Software, 2013, 24(8): 1695-1712. (in Chinese)

[3] Wong W, Horgan J, London S, Agrawal H. A study of effective regression testing in practice. In: Proc. of the Int. Symp. on Software Reliability Engineering. Albuquerque: IEEE Press, 1997. 264-274.

[4] Elbaum S, Malishevsky AG, Rothermel G. Prioritizing test cases for regression testing. In: Proc. of the Int. Symp. on Software Testing and Analysis. ACM Press, 2000. 102-112.

[5] Rothermel G, Unteh RH, Chu C, Harrold MJ. Prioritizing test cases for regression testing. IEEE Trans. on Software Engineering, 2001, 27(10): 929-948.

[6] Tonella P, Avesani P, Susi A. Using the case-based ranking

methodology for test case prioritization. In: Mancoridis S, ed. Proc. Of the 22nd IEEE Int'l Conf. on Software Maintenance. Los Alamitos: IEEE Press, 2006. 123-133.
[7] Yoo S, Harman M. Pareto efficient multi-objective test case selection. In: Rosenblum DS, ed. Proc. of the 2007 Int'l Symp. On Software Testing and Analysis. New York: ACM Press, 2007. 140-150.
[8] Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperative Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 1996, 26(1): 1-13.
[9] Duan Haibin. Principle of ant colony algorithm and its application [M]. Beijing: Science Press, 2005: 24-29. (in Chinese)
[10] Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutional Computation, 1997, 1(1): 53-66.
[11] Stützle T, Hoos H, MAX-MIN Ant System. Future Generation Computer Systems. 2000, 16(9): 889-914.
[12] Schoonderwoerd R, Holland O, Bruten J. Ant-based Load Balancing in Telecommunications Networks. Adaptive Behavior, 1996, 5(2):169-207.
[13] C.J. Liao, H.C. Juan. An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. Computers and Operations Research, 2007, 34(7):1899-1909.
[14] Wang Lei, Wu Qidi. Application of Ant Colony Algorithm in System Identification. Chinese Journal of Automation, 2003, 29(1): 102-109. (in Chinese)
[15] Han YF, Shi PF. An improved ant colony algorithm for fuzzy clustering in image segmentation. Neurocomputing, 2007, 70(4-6): 665-671.
[16] GU Conghui, LI Zheng, ZHAO Ruilian. ACO based test case prioritization and impact analysis of parameters. Journal of Frontiers of Computer Science and Technology, 2014, 8(12): 1463-1473. (in Chinese)
[17] XING Xing, SHANG Ying, ZHAO Ruilian, LI Zheng. Pheromone updating strategy of ant colony algorithm for multi-objective test case prioritization. Journal of Computer Applications, 2016, 36(9): 2497-2502. (in Chinese)
[18] Qu B, Nie CH, Xu BW. Test case prioritization based on test suite design information. Chinese Journal of Computers, 2008,31(3): 431-439. (in Chinese)
[19] Zhang XF, Nie CH, Xu BW, Qu B. Test case prioritization based on varying testing requirement priorities and test case costs. In: Proc. of the Int'l Conf. on Quality Software. IEEE Press, 2007. 15-24.
[20] Krishnamoorthi R, Sahaaya Arul Mary SA. Factor oriented requirement coverage based system test case prioritization of new and regression test cases. Information and Software Technology, 2009,51(4):799-808.
[21] Zhang Weixiang, Qi Yuhua, Li Dezhi. Test case prioritization based on discrete particle swarm optimization algorithm. Journal of Computer Applications. 2017, 37(1): 108-113. (in Chinese)
[22] Elbaum S, Malishevsky A, Rothermel G. Prioritizing test cases for regression testing. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2000. 102-112.
[23] Li Z, Harman M, Hierons RM. Search algorithms for regression test case prioritization. IEEE Trans. on Software Engineering, 2007, 33(4): 225-237.
[24] Zhang Wei-xiang, Wei Bo, Du Hui-sen. Test case prioritization method based on genetic algorithm. Journal of Chinese Computer Systems. 2015, 36(9): 1998-2002. (in Chinese)
[25] Wang Dingwei, Wang Junwei et al. Intelligent Optimization Method. Beijing: Higher Education Press, 2007.4. (in Chinese)
[26] WU H F，CHEN X Q， MAO Q H， et al． Improved ant colony algorithm based on natural selection strategy for solving TSP problem． Journal on Communications, 2013, 34(4): 165-170．(in Chinese)