Neural Computing and Applications
https://doi.org/10.1007/s00521-023-09000-2

ORIGINAL ARTICLE

Q-learning whale optimization algorithm for test suite generation with constraints support

Ali Abdullah Hassan[1] · Salwani Abdullah[1] · Kamal Z. Zamli[2,3] · Rozilawati Razali[1]

Received: 24 November 2021 / Accepted: 22 August 2023

Abstract
This paper introduces a new variant of a metaheuristic algorithm based on the whale optimization algorithm (WOA), the *Q*-learning algorithm and the Exponential Monte Carlo Acceptance Probability called (QWOA-EMC). Unlike WOA, QWOA-EMC permits just-in-time adaptive selection of its operators (i.e., between shrinking mechanism, spiral shape mechanism, and random generation) based on their historical performances as well as exploits the Monte Carlo Acceptance probability to further strengthen its exploration capabilities by allowing a poor performing operator to be reselected with probability in the early part of the iteration. Experimental results for constraints combinatorial test generation demonstrate that the proposed QWOA-EMC outperforms WOA and performs competitively against other metaheuristic algorithms.

Keywords Combinatorial testing · Constrained software testing · Meta-heuristic · Test case generation · Whale optimization algorithm · Reinforcement learning · *Q*-learning algorithm

1 Introduction

As part of the quality assurance exercise, testing is an important activity within the software development life-cycle. Although desirable to ensure conformance to specification, exhaustive testing is practically impossible to resource and timing constraints. Many sampling-based approaches have been proposed in the literature to minimize tests (e.g., equivalences partitioning and boundary values analysis). Although useful, much of the existing sampling approaches do not address fault due to interaction (termed *t*-way testing, whereby t indicates the interaction strength).

To date, considering *t*-way testing as an optimization problem, many metaheuristic-based strategies have usefully been developed in the literature. The performance of each metaheuristic-based *t*-way strategy often depends on its backbone algorithm. Owing to its robust global search capability, the Whale Optimization Algorithm (WOA) is a suitable choice as the backbone algorithm for our new *t*-way strategy. However, a closer look reveals the following limitations: (i) the three whale operators' selection is solely based on a fixed probability and amplitude change as a function of iteration [1]. Using probability and amplitude change as selection control can be problematic when they do not portray the search process's current need, (ii) no information can be inferred or manipulated on the historical performance of each WOA operator, and (iii) WOA also does not provide an effective way to go out of local optima [1].

Addressing these issues, this paper proposes an ensemble of the *Q*-learning algorithm with the WOA. More precisely, the *Q*-learning algorithm replaces the controlling parameters of WOA in order to balance between exploration and exploitation search. Additionally, the acceptance probability of the Exponential Monte Carlo algorithm (EMC) [2, 3] is embedded to reconsider the worst-performing operator in the early iteration to explore more search regions.

✉ Ali Abdullah Hassan
p94321@siswa.ukm.edu.my; ali87hassan@gmail.com

[1] Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

[2] Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia

[3] Faculty of Science and Technology, Universitas Airlangga, C Campus JI. Dr. H. Soekamo, Mulyorejo, Surabaya 60115, Indonesia

Published online: 18 September 2023

Thus, the contribution of our work is as follows: (i) A new QWOA-EMC variant algorithm that allows just-in-time adaptive selection of the best-performing operators (between shrinking mechanism, spiral shape mechanism, and random generation) at any particular instance of iteration (ii) An integrated acceptance probability Exponential Monte Carlo as the mechanism to escape from local optima, and (iii) a benchmark results of the combinatorial test suite with constraints support.

The remainder of the article is organized as follows: Sect. 2 illustrates the *t*-way testing background, and Sect. 3 presents the related work. Section 4 discusses the original WOA. Section 5 introduces the proposed method, including the *Q*-learning algorithm and the acceptance probability of the exponential Monte Carlo algorithm. Section 6 benchmarks the proposed method with the existing state-of-the-art strategies. Finally, Sect. 7 presents the conclusion.

2 Overview of *T*-way testing

The testers are always required to operate on a tight timetable. As a result, in order to obtain optimum test coverage and a fast fault detection rate, the testing techniques must be highly successful. The above features are delivered by *t*-way testing [1]. *T*-way testing is a method for examining all the possible discrete combinations of the software system parameters [4]. *T*-way testing uses specific Design of Experiments (DoE) techniques, such as the Covering Array (CA).

CAs generalize OAs by requesting that the *t*-tuples be covered at least once instead of a certain number of times to achieve balanced *t*-tuples [5]. A tuple is a finite and ordered list of objects in mathematics (i.e., the parameter interactions' values). A *t*-tuple is a series of *t* elements (or arranged list). CA is a combinatorial object denoted as CA (N; t, v^p), where p is the parameter (i.e., system configurations, input data, or both), N is the number of test cases generated, v is the parameter value, and t is the interaction strength.

The primary aim of *t*-way testing is to minimize the test suite size and yet not sacrificing its faults detection capability. A test suite is a series of test cases designed to evaluate a software system and show a specific set of behaviors. *T*-way testing is achieved by constructing a test case out of a software system input or a system configuration. Following that, a test suite for the software system is built from a series of test cases.

Consider the example of basic pizza ordering System as shown in Fig. 1. The basic pizza ordering System has five primary configurations/ parameters, i.e., pizza types, crust, toppings, size, and order. Let us say that all the parameters take two values (i.e., Pizza Types = {vegetarian, meat lover}, Crust = {thin crust, extra thick}, Toppings = {pepperoni, mushroom}, Size = {small, big}, Order = {order-in, take away}). For the given basic pizza ordering system, exhaustive testing of all potential interactions generally requires $2 \times 2 \times 2 \times 2 \times 2 = 32$ test cases to cover all basic pizza ordering system configurations. While, when meta-heuristic strategy (i.e., QWOA-EMC) was used in 2-way testing, only 6 test cases were generated to cover all the configurations of the example mentioned above as shown in Table 1.

It may be deduced from Table 1 that each 2-way interaction between parameters is only covered once (suggesting that the presented result is the best). However, there are certain constraints. Because pairings of Pizza Type (Vegetarian) and Toppings (Pepperoni) are untenable, they must be prohibited. Similarly, interactions between Pizza Type (Meat Lover) and Toppings (Mushroom) are not permitted. Thus, these combinations should be omitted when producing the final test suite. Such combination types are identified as forbidden combinations or constraint combinations. Forbidden combinations or constraints are types of test cases that may result in unsatisfactory performance or output. Thus, they should not be generated in the final test suite.

CA with constraints can be denoted as CCA $(N; t, v^p, F)$, where F is the forbidden combination. F can also be denoted as $F = \{(C_{p_{\text{no}},v_{\text{no}}}, C_{p_{\text{no}},v_{\text{no}}})_1, (C_{p_{\text{no}},v_{\text{no}}}, C_{p_{\text{no}},v_{\text{no}}})_2, \ldots, (C_{p_{\text{no}},v_{\text{no}}}, C_{p_{\text{no}},v_{\text{no}}})_n\}$, where C is the constraint, p_{no} represents the parameter number in the *t*-tuple table, and v_{no} represents the value number of the parameter in the *t*-tuple table.

In the case of the basic pizza ordering system, the configurations may be represented as CCA $(N; 2, 2^5, F)$ where $F = \{(C_{p_1,v_1}, C_{p_3,v_1}), (C_{p_1,v_2}, C_{p_3,v_2})\}$. Table 2 shows the proper representation of CCA when these constraints are taken into account. Table 2 confirms that the provided constraints are effectively forbidden. As a result, Vegetarian 2-way interactions will only cover Mushroom, whereas Meat Lover 2-way interactions can only take Pepperoni.

3 Related work

Several nature-inspired meta-heuristic algorithms emerged in recent years, attracting the attention of many researchers due to nature's role as a source of inspiration. However, the literature indicates that meta-heuristic algorithms are widely employed in various fields such as *t*-way testing but are underestimated [6, 7]. Not all meta-heuristics are efficient in *t*-way testing, even though some have proven to be

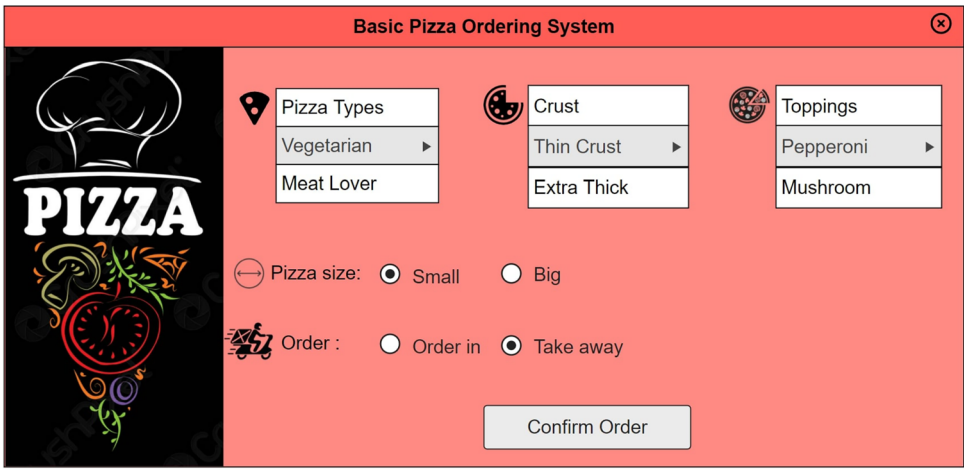

Fig. 1 Basic pizza ordering system

Table 1 2-way test suite CA$(6; 2, 2^5)$ for basic pizza ordering system example

No	Pizza types	Curst	Toppings	Size	Order
1	Vegetarian	Thin crust	Mushroom	Small	Order in
2	Vegetarian	Extra thick	Pepperoni	Big	Take away
3	Meat lover	Thin crust	Pepperoni	Big	Order in
4	Meat lover	Extra thick	Mushroom	Small	Take away
5	Vegetarian	Thin crust	Mushroom	Big	Take away
6	Meat lover	Extra thick	Pepperoni	Small	Order in

Table 2 2-way test suite with constraints CCA$(6; 2, 2^5, F)$, where $F = \{(C_{p_1,v_1}, C_{p_3,v_1}), (C_{p_1,v_2}, C_{p_3,v_2})\}$ for basic pizza ordering system example

No	Pizza types	Curst	Toppings	Size	Order
1	Meat lover	Extra thick	Pepperoni	Big	Take away
2	Vegetarian	Extra thick	Mushroom	Small	Order in
3	Meat lover	Thin crust	Pepperoni	Small	Order in
4	Vegetarian	Thin crust	Mushroom	Big	Take away
5	Meat lover	Thin crust	Pepperoni	Big	Order in
6	Vegetarian	Thin crust	Mushroom	Small	Take away

quite powerful and have thus been adopted for optimization, while others have not [8].

The most popular AI-based algorithm is the Genetic Algorithm (GA). This algorithm mimics natural selection, in which the fittest individuals are chosen for reproduction to produce offspring of the next generation. Esfandyari and Rafe adopted GA for the t-way testing strategy called Genetic Strategy (GS) [9]. GS supports uniform and variable CAs and higher interaction strengths of up to $t = 20$, through the modification of the crossover and mutation operators. However, it does not support constraints [9]. Moreover, the GS parameters like population size, crossover, and mutation rates were tuned properly to improve the efficiency and performance [9]. Similarly, a number of other approaches, strategies and tools that are based on GA have been created [10, 11].

Discrete Particle Swarm Optimization (DPSO) [12] is variant of PSO that utilizes separated particle swarm as its foundation and has a schema based on separated search space. DPSO exhibits improved performance owing to the introduction of two additional procedures: (1) particles are reinitialized, and (2) gbest (i.e., best test case obtained) is additionally evaluated. DPSO also provides instructions for parameter tuning. DPSO has exhibited better results compared to the standard PSO, and the other existing evolutionary algorithms [12]. Unfortunately, most of the variants of PSO suffer from premature convergence [4].

Cuckoo Search (CS) is an AI-based strategy applied to construct CA with interaction strength of up to six [13]. The idea behind implementing CS to t-way testing was to decrease the search space for test cases. Furthermore, inspired by how the musicians compose their best song, Harmony Search, an AI-based strategy was applied to t-way testing. The harmony search implementation was referred to as Harmony Search Strategy (HSS) [14]. The method followed in HSS was the addition of a test case to the CA (test suite) at each iteration until all the t-interactions were covered and the CA was completed. HSS could support higher interaction strength of up to fifteen and addresses constraints. Since t-way testing still has room for improvement, researchers are inclined to apply newly developed meta-heuristic algorithms (i.e., the AI-based ones) in the hope of producing better results.

The hybrid meta-heuristics (AI-based strategies) have attracted attention in the field of optimization. The best

performance and results obtained for various real-world and classical optimization issues have been achieved using hybrid algorithms [15]. It has been almost 2 decades since the first hybrid meta-heuristic algorithm was applied to combinatorial testing. The rest of this section reports the review of the recent hybridizations that have exhibited better performance and promising results in CA construction. One example of such hybridization is the Fuzzy Self Adaptive PSO (FSAPSO) [16]. The methodology of FSAPSO involves using the Mamdani-type fuzzy inference system (FIS) to tune the parameters of PSO. FSAPSO was reported to outperform DPSO and CS in most of the cases. In 2018, Zamli et al. proposed the same hybridization with a different process and functionality, which was named FPSO [17]. The methodology of FPSO involved allowing fuzzy adaptive selection of PSO's global and local search process. FPSO exhibited efficiency, and in certain cases, outperformed DPSO in the average result; both hybridizations generated CAs with interaction strength up to four (i.e., $t \leq 4$). Another strategy used FIS to balance the exploration and exploitation of the teaching learning-based optimization algorithm (TLBO) and was named ATLBO [18]. This strategy's goal was to adaptively choose local and global search processes for generating both uniform and mixed-strength t-way test suites and support interaction strength of up to four. This strategy has exhibited competitive results when compared against the other AI-based strategies.

Zamli et al. introduced a hybridization of the sine-cosine algorithm (SCA), and the Q-learning algorithm referred to as QLSCA that generated a test suite of interaction strength of up to four [15]. QLSCA outperformed several existing strategies by producing the best average results for most cases. Several researchers have proposed hybridization for strengthening strategies by enhancing the quality function, improving the solutions' diversification, and overcoming the common meta-heuristic (AI-based) drawbacks. In t-way testing, the objective of hybridization is to minimize the test suite. These hybridization methods have been reported in some earlier studies [19, 20].

All the previous works on t-way testing have usefully contributed to improving the current state of the art on interaction t-way test case generation and potentially broadens its applications in line with the emergence of the IR4.0 agenda. Nevertheless, in the context of AI-based approaches related to meta-heuristics, there are still three main issues that can be investigated further. Firstly, many newer meta-heuristics have been developed in the literature in recent years (i.e., including that of Whale Optimization Algorithm, Grey Wolf, and the like). Capitalizing on their reported performance would be deemed a profitable endeavor to improve the generation process's efficiency. Secondly, as the No Free Lunch Theorem [21] postulates, no single meta-heuristic can singly outperform others—as some algorithms perform better than others on certain types of optimization problems. Lastly, the WOA limitations are addressed, and the limitations are discussed in the following section. Our work investigates how well QWOA-EMC performs for constrained test suite generation and compares it with the state of the art.

4 Whale optimization algorithm

WOA is a modern meta-heuristic algorithm inspired by nature; it was introduced in 2016 [22]. WOA is a swarm-based approach that imitates the hunting behavior of humpback whales. Humpback whales are intelligent creatures that have a sophisticated way of working collectively. They use a unique tracking technique, referred to as the bubble-net feeding method, illustrated in Fig. 2. This technique is performed by producing peculiar bubbles along a circle or '9'-shaped path. The whales hunt near the surface and trap the victim in a net of bubbles.

WOA comprises two phases: the exploitation phase and the exploration phase. Encircling-a-prey method and the spiral bubble-net attacking method are used in the exploitation phase, while the exploration phase involves searching randomly for a victim.

The first method of the exploitation phase is the encircling-a-prey, where humpback whales identify the victim's location, and then, they surround it. In WOA, the target victim is presumed to be the current best candidate solution. Next, the best search agent is located, the other search agents attempt to move their locations toward it. In other words, it updates the movement (location) of the whale

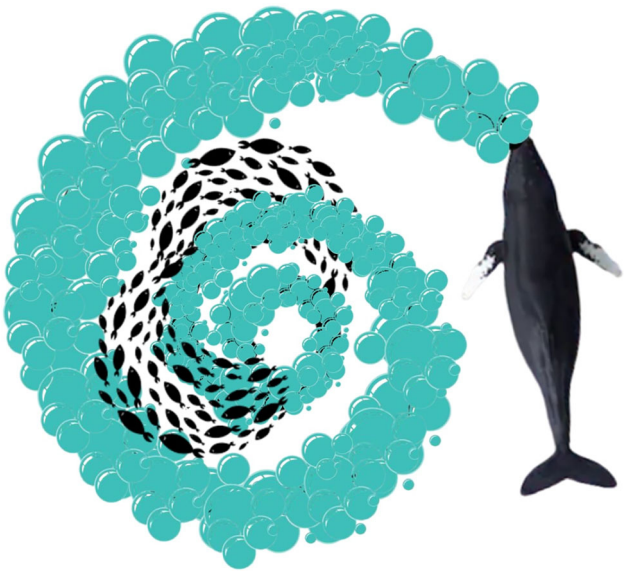

Fig. 2 Bubble-net hunting behavior of humpback whales Source: [1]

around the victim, which can be mathematically modeled as follows:

$$D = |CX^* - X(t)| \tag{1}$$

$$X(t+1) = X^*(t) - A.D \tag{2}$$

where t is the current iteration, X^* is the best solution acquired so far, X is the current solution. A and C are coefficients computed as following:

$$A = 2a.r - a \tag{3}$$

$$C = 2.r \tag{4}$$

where a is linearly reduced from 2 to 0 over the trajectory of iterations as showing in Eq. (5) and r is a random number between 0 and 1.

$$a = 2 - t\frac{2}{\text{MaxIter}} \tag{5}$$

The second method is the Bubble-net attacking method which involves two mechanisms: the shrinking encircling mechanism and the spiral updating position mechanism. Shrinking encircling mechanism is carried out by the reduction in the value of a in Eq. (3). Thus, the new location of a search agent is located between the agent's actual location and the location of the existing best agent. The spiral updating position mechanism involves computing the distance between the current solution (whale) and the best solution (victim) by using the spiral equation as following:

$$X(t+1) = D^{'}.e^{\text{bl}}.\cos(2\Pi l) + X^*(t) \tag{6}$$

where $D^{'}$ is the distance between the whale and the victim, b is a constant for defining the shape of the logarithmic spiral, and l is a random number between -1 and 1. Humpback whales use both mechanisms simultaneously. To model this behavior, a probability of 50% is introduced to select one of the mechanisms to update the whales' location during the search. The mathematical model is as follows:

$$X(t+1) = \begin{cases} \text{Shrinking/encircling equation}(2), & \text{if } p < 0.5 \\ \text{Spiral-shaped path equation } (6), & \text{if } p \geq 0.5 \end{cases} \tag{7}$$

where p is a random number in [0,1].

While in the exploration phase, WOA involves a global search. The whales search randomly based on the location of each other. Therefore, a search agent's location is updated randomly instead of depending on the best search agent identified so far. This technique is used when the random values of A are greater than one to cause the search agent to move away from a reference whale (best solution). This mechanism emphasizes global search and allows the WOA algorithm to perform exploration as the Algorithm 1 illustrated in Fig. 3. The mathematical model for the exploration phase is as follows:

$$D = |C.X_{\text{rand}} - X| \tag{8}$$

$$X(t+1) = X_{\text{rand}} - A.D \tag{9}$$

5 Proposed method

The proposed QWOA-EMC integrates the Q-learning algorithm with the Whale optimization algorithm and acceptance probability of the Exponential Monte Carlo algorithm. The controlling parameters of WOA depend on random distribution, causing an imbalance between exploitation and exploration [4]. Thus, the controlling parameters are to be replaced by the Q-learning algorithm to balance exploitation and exploration. While the acceptance probability of the Exponential Monte Carlo algorithm is employed to escape local optima by allowing the worst solution to be accepted at a certain degree.

5.1 *Q*-learning algorithm

Q-learning Algorithm [23] is labeled to be one of the most effective and efficient methods for reinforcement learning [15]. Q-learning operates upon the reward or punishment concept. The learner acquires a reward or punishment as input from the environment once the state changes. When an action is executed, the given state moves to the next one. In a state-action table, also known as Q-table, the learner adjusts the reward and punishment. In the future, the adjusted Q-table is used as a reference when encountering comparable situations, which means the learner does not need a structural environmental model. Thus, the Q-learning algorithm is considered model-free.

Q-table utilizes a state-action pair to index a Q value as a cumulative reward and is denoted as $Q(s, a)$ where s is the state, and a is the action. The Q-table is dynamically updated depending on a given state-action pair's reward/punishment.

$$Q_{(t+1)}(s_t, a_t) = Q(s_t, a_t) + \alpha_t(r_t + \gamma \max(Q_t(s_{t+1}, a_{t+1})) - Q(s_t, a_t)) \tag{10}$$

where γ is the discount factor within [0,1], r is reward/punishment and α is the learning rate within [0,1] and calculated as follows

Fig. 3 Pseudo-code of the WOA algorithm

Algorithm 1: Pseudo-code of the WOA algorithm

```
Initialize the whales population X_i (i = 1, 2, ..., n)
Calculate the fitness of each solution
X* = the best search agent.
while i < maximum number of iteration do
    for every solution do
        Update a, A, C, l, and p
        if ( p < 0.5) then
            if ( |A| < 1 ) then
                Update the position of the current solution using Eq (2)
            else if ( |A| > 1) then
                Random solution is generated
                Update the position of the current solution using Eq (9)
        else if ( p >= 0.5 ) then
            Update the position of the current solution using Eq (6)
    end
    Check whether any solution exceeds the search space and adjust it
    Compute the fitness of every solution
    UpdateX* if there is a better solution
    t = t + 1
end
return X*
```

$$\alpha = 1 - 0.9 * \frac{t}{\text{MaxIteration}} \tag{11}$$

The value of the parameter α is an indication to perform exploration or exploitation. If the value is close to 1, the recently acquired data are given a greater priority, meaning exploration is performed for all defined states. Whereas, if the value is close to 0, the current data are given greater priority to be exploited. The value of the parameter γ is an indication of whether to take the current reword/punishment or the previous one, and it was set to 0.8 as suggested by [24]. The value of parameter r is set based on Eq. 12.

The pseudo-code of the Q-learning algorithm is stated in Fig. 4.

$$\begin{cases} r_t = 1, & \textit{if the current action improves the solution} \\ r_t = -1, & \textit{otherwise} \end{cases} \tag{12}$$

5.2 The acceptance probability of exponential Monte Carlo algorithm

The acceptance probability of the EMC [2] is comparable to the one of the Simulated Annealing (SA) algorithm;

Fig. 4 Pseudo-code of the Q-learning algorithm

Algorithm 2: Pseudo-code of the Q-learning algorithm

```
for each state S = [s_1, s_2, ..., s_n] and action A = [a_1, a_2, ..., a_n] do
    set Q (s_t, a_t) = 0
end
Randomly select an initial state, s_t
while stopping criteria are not met do
    Select the best action a_t, for the current state s_t from Q-table
    Execute action a_t and get immediate reward/punishment r_t using Eq (12)
    Get the maximum Q value for the next state s_{t+1}.
    Update α using Eq (11).
    Update Q-table entry using Eq (10).
    Update the current state s_t = s_{t+1}.
end
return the updated Q (s, a) table
```

however, the cooling schedule is not required. The acceptance probability depends on three factors for the worse solution to probable be accepted. The factors are the solution's quality, the number of iterations, and the number of successive non-improvement iterations. The last factor can be considered as a timeframe for the search. If the search spent a particular time with no improvement, it means that it got stuck in local optima. The acceptance probability is calculated as follows:

$$e^{-\Theta/\lambda} \tag{13}$$

where, $\Theta = \delta * t$, δ is the difference between the current and the best solution, t is the iteration counter (i.e., the current iteration) and λ is a successive non-improvement counter.

One may note that, as the number of iterations t increments, the probability of choosing a worse solution reduces. But, if for specific successive iterations, no improvement occurs, the likelihood of tolerating a worse solution would increment depending on the current solution's objective function and the number of iterations. In other words, if δ is small or λ is high, the selection of a worse solution seems probable. Figure 6 demonstrates the pseudo-code of QWOA-EMC and illustrates how the acceptance probability is employed.

5.3 The QWOA-EMC strategy for *t*-way testing

The WOA and Q-learning algorithms are combined as QWOA-EMC, with the Q-learning algorithm substituting the controlling parameters of WOA (i.e., p and A) with a state-action pair to index Q value to make the selection. WOA has three search operators (shrinking, spiral-shaped, and random). The WOA parameters determine which one to use. The Q-table of QWOA-EMC strategy is constructed as a 3*3 matrix in which the rows represent the state (s_t). The columns represent the action (a_t) for each state as shown in Fig. 5.

The Q-learner's actions and the states to be learned are chosen at random at the start of the search (i.e., for the first episode). Then, depending on the Q-table, actions are determined, and states are updated, meaning that WOA search operators are selected based on the past performance. A Q-table snapshot for the QWOA-EMC is shown in Fig. 5, along with a numerical illustration. let us say that $r = -1$ (i.e., no improvement), $\alpha_t = 0.69$, $\gamma = 0.80$, and the current state-action pair is s_t = *Shrinking Operator* and has a_t = *Random Operator*. Then, the $Q(s_t, a_t) = 0.95$ as highlighted in Fig. 5. Then, the new value for $Q_{t+1}(s_t, a_t)$ in the Q-table is updated based on Eq. 10 as follows:

$$Q_{(t+1)}(s_t, a_t) = 0.95 + 0.69[-1 + 0.80 * \max(1.29, 0.00, -0.58) - 0.95] = 0.32$$

After that, the state is changed from *Shrinking Operator* to *Random Operator*. The following action is then selected based on the best action (i.e., max value) in the *Random Operator* state, a_{t+1} = *Shrinking Operator* as highlighted in Fig. 5. This is unlike the original WOA algorithm in that the shrinking, spiral-shaped, or random operator is selected based on the controlling parameters, p and A. Figure 6 demonstrates the pseudo-code of QWOA-EMC.

The QWOA-EMC strategy is employed to automatically generate a test suite and decrease the number of test cases. Figure 7 presents an overview of the QWOA-EMC implementation in t-way testing. To generate the t-tuple table, assume the number of parameters is four (a, b, c, d), each one of them has two values (0,1), and the interaction strength is $t = 2$ (i.e., $\mathrm{CA}(N; 2, 2^4)$). Then, the 2-way combinations are (*ab*, *ac*, *ad*, *bc*, *bd*, and *cd*). Thus, the t-tuple table will be presented by the table in Fig. 7, where the x would be replaced randomly with one of the parameter values (0,1) during the search.

Meanwhile, in the presence of constraints, each time the QWOA-EMC updates its solution (i.e., generate a new solution), the new solution will be checked whether or not it is one of the forbidden combinations. This step is to ensure that the solutions will not converge to one of the forbidden combinations.

Actions / States	Shrinking Operator	Spiral-shaped Operator	Random Operator
Shrinking Operator	-0.12	-1.02	0.95
Spiral-shaped Operator	0.00	1.65	0.78
Random Operator	1.29	0.00	-0.58

Current Q-table

Actions / States	Shrinking Operator	Spiral-shaped Operator	Random Operator
Shrinking Operator	-0.12	-1.02	0.32
Spiral-shaped Operator	0.00	1.65	0.78
Random Operator	1.29	0.00	-0.58

Updated Q-table

Fig. 5 The Q-table update example

Fig. 6 Pseudo-code of the QWOA-EMC algorithm

Algorithm 3: Pseudo-code of the QWOA-EMC algorithm

```
Initialize the whales population X_i (i = 1, 2, ..., n)
Calculate the fitness of each search agent.
X* = the best search agent.
X_current = the current search agent.
episode = 0.
for each state S = [s_1, s_2, ..., s_n] and actions A = [a_1, a_2, ..., a_n] do
|  set Q(s_t, a_t) = 0
end
while i < maximum number of iteration do
    for every search agent do
        if (episode < 1) then
            Select action and state randomly
        else
            From the current state s_t, selecte the best action a_t from Q-table
        if ( action == shrinking mechnism ) then
            Update the current search agent using Eq (2)
        else if ( action == spiral-shaped mechnism ) then
            Update the current search agent using Eq (6)
        else if ( action == random generation ) then
            Select a random search agent ()
            Update the current search agent using Eq (9)
        Check if search agent goes beyond the search space
        Calculate the fitness of current search agent
        if ( X_current better than X* ) then
            X* = X_current
            λ = 1
            non-improvement = 0
        else
            δ = X_current - X*
            Generate a random number randNum in [0 1]
            if ( randNum ≤ e^(-δ*t / λ) ) then
                X* = X_current
                λ = 1
                non-improvement = 0
            else
                non-improvement ++
                if ( non-improvement == Max-non-improvement) then
                    λ ++
                    non-improvement ++
        Get immediate r using Eq (12)
        Get the maximum Q value for the next state s_t+1
        Update α using Eq (11)
        Update Q-table entry using Eq (10)
    t = t + 1
    episode ++
return X*
```

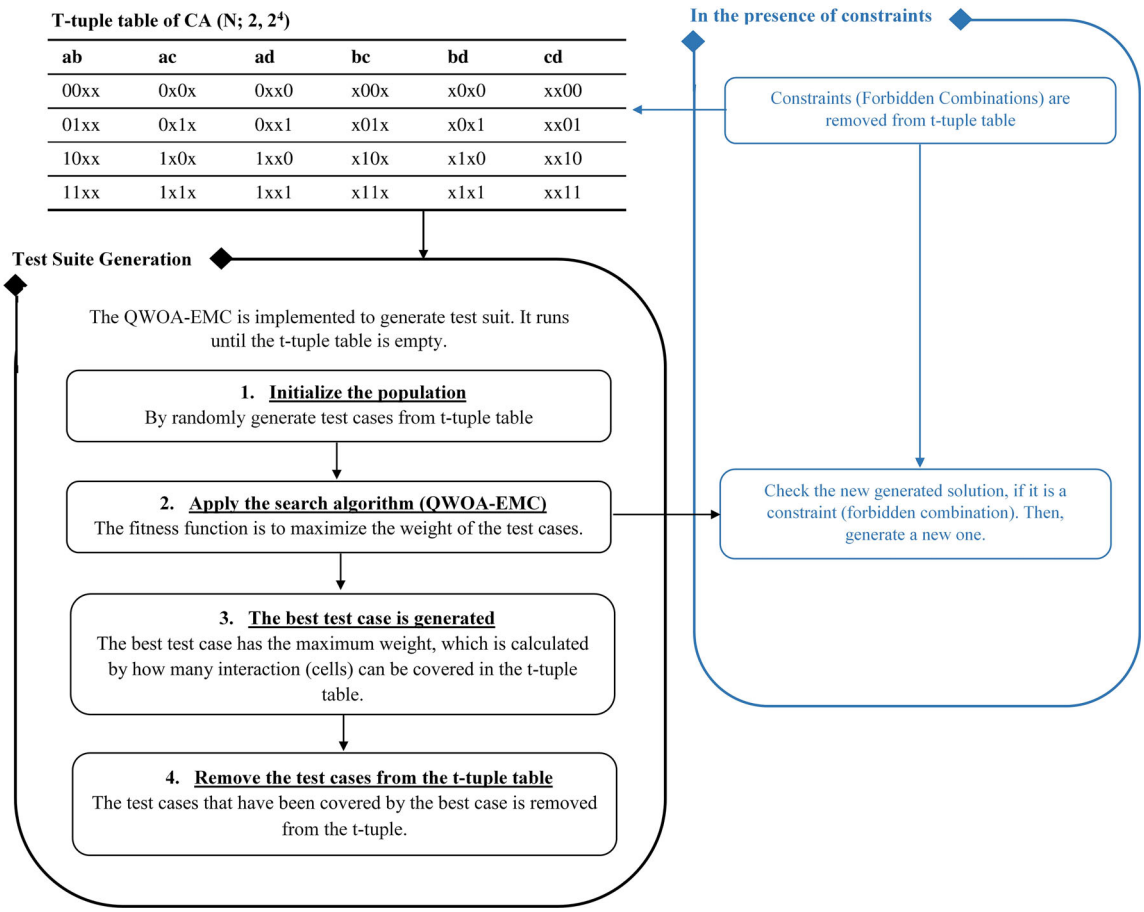

Fig. 7 The overview of *t*-way testing implementation

6 Experimental results and discussion

This section reports assessing the efficiency of QWOA-EMC versus other existing, well-known, population-based meta-heuristic algorithms and pure computational strategies. The efficiency was measured in terms of the CA's size (i.e., test suite size). Two more measurements have been provided: (1) the convergence behavior of the proposed QWOA-EMC strategy and (2) the applicability of the proposed strategy for combinatorial test data generation.

We divide our experiments into four parts. First, we systematically tuned the *max-non-improvement* parameter of the acceptance probability of the EMC algorithm. Second, we evaluated and compared the QWOA-EMC strategy with existing population-based meta-heuristic algorithms. Third, we benchmarked our method with existing constraint-supporting strategies. Lastly, we evaluated the applicability and the effectiveness of the QWOA-EMC strategy for combinatorial test data generation. Another measurement was also applied based on Wilcoxon's signed-rank test for all reported results.

6.1 Parameter tuning

The population size and the maximum number of iterations are still required to be tuned for most meta-heuristic algorithms. Thus, we have tuned the population size, and the maximum number of iterations in our previous publication [1]; where the population size was set to 180 and the maximum number of iterations to 100.

Another parameter that we need to consider is the *max-non-improvement* parameter in the EMC algorithm's acceptance probability. This parameter counts the number of iterations where the solutions were not improved. Small *max-non-improvement* could prevent the best candidate solution from being reached, while a large one could lead the candidate solution to be trapped in local optima. Therefore, it is necessary to coordinate the selection of *max-non-improvement* carefully.

To tune the *max-non-improvement* parameter, the covering arrays CA $(N; 2, 5^7)$ and CA$(N; 2, 4^6)$ were used as a case study. The rationale for accepting these covering arrays is that many AI-based approaches are tuned with the same covering arrays [1, 4].

Table 3 Best and average test suite size and execution time acquired with varied *max-non-improvement* parameter for CA($N; 2, 4^6$) and CA($N; 2, 5^7$)

Max number of no improvement	CA($N; 2, 4^6$)				CA($N; 2, 5^7$)			
	Best		Average		Best		Average	
	Size	Time	Size	Time	Size	Time	Size	Time
5	**23**	35.45	24.8	37.84	40	112.45	42	121.6
10	**23**	36.82	24.8	39.89	38	110.2	40.75	118.51
20	**23**	35.65	**24.75**	37.4	**36**	112.72	39.3	119.28
30	**23**	35.48	**24.75**	37.88	**36**	110.75	**38.9**	117.77
40	24	37.44	24.9	40.97	37	108.88	40.5	116.29
50	24	37.28	25.05	40.38	38	116.87	40.4	120.72
60	24	38.21	24.95	40.43	40	110.03	42.25	119.72

The QWOA-EMC strategy for CA($N; 2, 5^7$) and CA($N; 2, 4^6$) was executed repeatedly 30 times with different *max-non-improvement* values tested by having various values (i.e., 5, 10, 20, 30, 40, 50, 60). The best test suite size and the average test suite size are shown in Table 3, where the darkened cells indicate the most optimal size. The execution time is reported in seconds. The best execution time and the average execution time are also shown in Table 3.

In Table 3, for CA($N; 2, 4^6$), the best results achieved when *max-non-improvement* was 20 and 30. While for CA($N; 2, 5^7$), best results were achieved when *max-non-improvement* was 30. When considering the time, the optimal *max-non-improvement* was selected, which is 30.

6.2 Benchmarking QWOA-EMC strategy against existing strategies

To assess the performance of QWOA-EMC, we benchmarked it against other existing strategies in terms of the test suit size. Two well-known datasets were employed in the experiments.

1. Comparing the QWOA-EMC strategy with currently available strategies using CA(t, v^7), where the number of parameters remains constant while their values are varied. Besides, the interaction strength t is varied from 2 to 6.
2. Comparing the QWOA-EMC strategy with existing strategies using CA($t, 3^P$), where the number of parameters is varied, and their values are kept constant. Besides, the interaction strength t is varied from 2 to 6.

The experimental environment is a laptop operating on Windows 10, with a 64-bit, 2.71 GHz, an Intel Core i5 CPU, and 8 GB of RAM. The proposed strategy was coded and implemented in Java. To put our work into perspective, we provided the parameter settings for each meta-heuristic algorithm used for the comparison and are displayed in Table 4.

In Table 5, the configurations of CA($t, 3^P$) were adopted, where t varied as $2 \leq t \leq 6$, p varied as $3 \leq p \leq 12$ and v remained constant at $v = 3$; the results were reported in terms of the best and average among the 31 runs. The results revealed that QWOA-EMC outperformed all the pure computational strategies because they construct tests in a locally optimized way that does not always result in a globally optimized test set. Also, QWOA-EMC outperformed most AI-based strategies, including the standard WOA, GBGA, GS, APSO, CS, and ABSVS. Moreover, QWOA-EMC produced competitive results compared to QLSCA and DPSO strategies.

In order to ensure that QWOA-EMC exhibited superiority over the existing strategies, a statistical analysis was conducted, particularly the Wilcoxon signed-rank test, which is a nonparametric test for matched or coupled data concentrated at differential ratings. However, it also considers the extent of the observed differences in response to evaluating the signs of the differences. The reason for utilizing the Wilcoxon signed-rank test was that it could inform if there exists a significant difference between the two results.

Wilcoxon's signed-rank test produced two factors. The first one is Asymp. Sig. (2-tailed) and Z, which are statistical tests that indicate the difference between the two groups. If the value of Asymp. Sig. (2-tailed) was smaller than 0.05, it implied that a significant difference between the two groups. Although the value of Z is not relevant and beyond the applicability of this study, the value has nonetheless been provided in the report. The second factor is the ranking, which ranks which values are greater than, equal to, or less than the comparable values.

Thus, Wilcoxon signed-rank test was applied to the results reported in Table 5. Table 6 presents the outcomes of the Wilcoxon signed-rank test. It was confirmed that QWOA-EMC produced results that were significantly different from those produced by the standard WQA, GBGA, GS, APSO, CS, and ABCVS strategies, with the exception of the QLSCA and DPSO strategies. However, one may

Table 4 Parameter values used for the existing meta-heuristic strategies

Algorithm	Parameters	Values
DPSO [12]	Population size	80
	Acceleration coefficients	1.3
	Inertia weight	0.5
	Iterations	250
	Probability parameter 1	0.5
	Probability parameter 2	0.3
	Probability parameter 3	0.7
CS [13]	Population size	100
	Probability Pa	0.25
	Iterations	100
GS [9]	Population size	250
	Mutation rate	0.4
	Crossover rate	0.4
ABCVS [25]	Population size	50
	Number of cycles	80
QLSCA [15]	Population size	40
	Iterations	100
	Constant M	3
APSO [16]	Population size	80
	Dynamic acceleration coefficients (c_1 and c_2)	$1 \leq c \leq 2$
	Dynamic Inertia weight (w)	$1 \leq w \leq 2$
	Iterations	100

argue that there were insufficient output samples for the QLSCA and DPSO strategies to provide a comprehensive picture. This was because both the strategies' interaction strength was up to four (i.e., $t \leq 4$).

Table 7 which was for CA(t, v^7) configurations where t varied as $2 \leq t \leq 6$, v varied as $2 \leq v \leq 7$ and p remained constant at $p = 7$. The results showed that QWOA-EMC outperformed the pure computational strategies and the AI-based strategies. Additionally, the Wilcoxon test was implemented on the results reported in Table 7 and the outcomes of the Wilcoxon test illustrated in Table 8.

Furthermore, the Wilcoxon test results presented in Tables 6 and 8 revealed that the results of QWOA-EMC were not significantly different from those of QLSCA and DSPO. This occurred because there were insufficient samples to compare and test as the interaction strength of both these strategies was only up to four (i.e., $t \leq 4$). Moreover, it has been demonstrated previously that the higher the interaction strength, the higher is the efficiency of the strategy (algorithm) in detecting faults [4]. Therefore, QWOA-EMC produces a test suite with an interaction strength of up to 20.

Additionally, the convergence behavior of the proposed QWOA-EMC was analyzed. Two CAs were selected for the analysis: CA$(N; 2, 3^{13})$ and CA$(N; 4, 3^5)$. Figure 8 depicts the convergence behavior of the standard WOA and QWOA-EMC, where *Y-Axis* represents the number of uncovered test cases in the t-tuple table, and *X-Axis* represents the iterations. It may be noticed that QWOA-EMC managed to converge faster than the standard WOA. In Fig. 8a, QWOA-EMC covered the t-tuple table of CA$(N; 2, 3^{13})$ on only 19 iterations, unlike WOA, which took 22 iterations to cover the t-tuple of CA$(N; 2, 3^{13})$. While in Fig. 8b, QWOA-EMC covered the t-tuple table of CA$(N; 4, 3^5)$ on only 91 iterations, unlike WOA, which took 105 iterations to cover the t-tuple of CA$(N; 4, 3^5)$. Thus, Fig. 8 and the results in Tables 5 and 7 indicated that QWOA-EMC could effectively escape the local optima.

Complementing the convergence behavior analysis, the box plot analysis comparison between WOA and QWOA-EMC in Fig. 9 reveals several salient features of QWOA-EMC. Considering the CA$(N; 2, 3^{13})$, it can be observed that QWOA-EMC has a lower quartile bias range, lower whisker width, and a lower median than the WOA, indicating consistently better results for the 31 runs. For the CA$(N; 2, 10^5)$, both box plots are symmetric. However, the QWOA-EMC's quartile range is smaller, and its median is much lower than that of the WOA. In the CA$(N; 3, 4^6)$, The box plot for the WOA is asymmetric. The WOA also has a larger first quartile range and a higher mean than the QWOA-EMC. Furthermore, WOA 's best result is an outlier which is in the QWOA-EMC quartile range. The

Table 5 Test suite size performance for CA(t, 3^P) where P varied from 3 to 12 and t varied from 2 to 6

CA(t, 3^P)		Pure computation strategies					AI-based strategies										
t	p	Jenny [26] Best	TConfig [27] Best	PICT [28] Best	IPOG-D [29] Best	IPOG [30] Best	GBGA [31] Best	DPSO [12] Best	GS [9] Best	CS [13] Best	QLSCA [15] Best	APSO [16] Best	ABCVS [25] Best	WOA [1]		QWOA-EMC	
														Best	Mean	Best	Mean
2	3	**9**	10	10	15	**9**	**9**	NS	**9**	**9**	**9**	**9**	**9**	**9**	9.8	**9**	***9.63***
	4	13	10	13	15	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	10.3	**9**	***9.93***
	5	14	14	13	15	15	12	**11**	**11**	**11**	**11**	**11**	**11**	**11**	12.86	**11**	***12.69***
	6	15	15	14	15	15	14	14	13	13	14	**12**	13	14	14.56	**12**	***14.4***
	7	16	15	16	15	15	15	15	**14**	**14**	**14**	15	15	**14**	15.26	**14**	***15.04***
	8	17	17	16	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	15.93	**15**	***15.58***
	9	18	17	17	**15**	**15**	16	**15**	**15**	**15**	**15**	16	16	16	16.7	**15**	***16.53***
	10	19	17	18	21	**15**	16	16	16	17	**15**	17	17	16	17.73	16	***17.39***
	11	17	20	18	21	17	17	17	**16**	18	**16**	NS	17	17	***17.93***	**16**	17.95
	12	19	20	19	21	21	18	**16**	**16**	18	**16**	NS	18	17	18.76	17	***18.41***
3	4	34	32	34	**27**	32	29	NS	**27**	28	**27**	**27**	**27**	**27**	31.53	**27**	***31.16***
	5	40	40	43	45	41	39	41	38	38	39	41	38	38	40.33	**37**	***39.92***
	6	51	48	48	45	46	45	**33**	43	43	**33**	45	44	42	45.9	39	***44.97***
	7	51	55	51	50	55	49	**48**	49	**48**	49	**48**	49	**48**	51.6	**48**	***50.56***
	8	58	58	59	**50**	56	54	52	54	53	52	**50**	54	53	55.96	51	***54.71***
	9	62	64	63	71	63	58	**56**	58	58	**56**	59	58	57	59.7	**56**	***58.59***
4	5	109	97	100	162	97	87	NS	90	94	**81**	94	98	90	100.03	90	***99.05***
	6	140	141	142	162	141	133	131	129	132	129	129	135	129	***134.65***	**128**	134.66
	7	169	166	168	226	167	156	**150**	153	154	**150**	154	157	154	159.46	151	***157.31***
5	6	348	305	310	386	305	**273**	NS	301	304	NS	NS	**273**	304	312.22	298	***312.09***
	7	458	477	452	678	466	433	NS	432	434	NS	NS	433	432	441.26	**430**	***438.81***
6	7	1089	921	1015	1201	**921**	982	NS	963	973	NS	NS	982	959	976.73	945	***965.76***
	8	1466	1515	1455	1763	1493	NS	NS	1399	1401	NS	NS	NS	1394	1402.24	**1389**	***1402.16***

Table 6 Wilcoxon test for the results reported in Table 5

	Ranks			Test statistics	
	QWOA-EMC >	QWOA-EMC <	QWOA-EMC =	*Z*	Asymp. Sig. (2-tailed)
WOA	0	13	10	− 3.204	0.001
ABCVS	1	16	5	− 2.899	0.004
APSO	1	9	7	− 2.534	0.011
QLSCA	5	5	9	− 0.209	0.835
CS	0	16	7	− 3.536	0.000
GS	1	12	10	− 2.993	0.003
DPSO	3	6	7	− 0.905	0.366
GBGA	2	16	4	− 2.475	0.013
IPOG	2	17	4	− 3.067	0.002
IPOG-D	1	19	3	− 3.866	0.000
PICT	0	23	0	− 4.211	0.000
TConfig	1	22	0	− 3.565	0.000
Jenny	0	22	1	− 4.118	0.000

Table 7 Test suite size performance for CA(t, v^7) where v varied from 2 to 7 and t varied from 2 to 6

CA(t, v^7)		Pure computation strategies					AI-based strategies								
t	v	Jenny [26] Best	TConfig [27] Best	PICT [28] Best	IPOG-D [29] Best	IPOG [30] Best	QLSCA [15] Best	GS [9] Best	DPSO [12] Best	APSO [16] Best	CS [13] Best	WOA [1] Best	WOA [1] Mean	QWOA-EMC Best	QWOA-EMC Mean
2	2	8	7	7	8	7	7	**6**	7	**6**	**6**	**6**	7.5	**6**	***7.25***
	3	16	15	16	15	15	15	**14**	**14**	15	15	**14**	15.5	**14**	***14.77***
	4	28	28	27	32	29	**23**	24	24	25	25	25	25.76	24	***25.25***
	5	37	40	40	45	45	**34**	36	**34**	35	37	36	38.56	36	***37.7***
	6	55	57	56	72	55	48	52	**47**	NS	NS	52	54.33	51	***53.16***
	7	74	76	74	91	**49**	64	68	64	NS	NS	70	72.53	70	***72.12***
3	2	14	16	15	14	16	15	**12**	15	15	**12**	**12**	***14.2***	**12**	14.29
4	3	51	55	51	50	55	49	49	49	**48**	49	49	51.46	**48**	***50.06***
	4	124	**112**	124	114	**112**	**112**	116	**112**	118	117	116	119.36	113	***116.9***
	5	236	239	241	252	237	**215**	221	216	239	223	223	230.07	221	***224.8***
	6	400	423	413	470	420	**364**	374	365	NS	NS	382	***386.25***	382	***386.25***
	2	31	36	32	40	35	31	27	34	30	27	27	30.86	**25**	***30.7***
	3	169	166	168	226	167	149	153	**150**	153	155	152	158.46	151	***156.48***
	4	517	568	529	704	614	477	486	**472**	**472**	487	484	***488.58***	484	***488.58***
5	2	57	56	57	80	60	NS	**51**	NS	NS	53	52	55.73	53	***55.41***
	3	458	477	452	678	466	NS	432	NS	NS	439	432	441.58	**430**	***436.41***
	4	1938	**1792**	1933	2816	1792	NS	1821	NS	NS	1845	1815	1823.1	1813	***1822.77***
6	2	87	**64**	72	96	**64**	NS	65	NS	NS	66	**64**	75.3	**64**	***72***
	3	1087	**921**	1015	1201	921	NS	963	NS	NS	973	945	969.61	945	***968.61***
	4	6127	NS	5847	5120	**4096**	NS	5608	NS	NS	5610	5567	***5567***	5567	5591.12

box plot chart led to the conclusion that QWOA-EMC has superior average solutions than WOA, and that its best solutions are not generated by chance as they were for WOA when its best solution was an outlier.

6.3 Benchmarking QWOA-EMC strategy in the presence of constraints against existing strategies

In this section, we compare QWOA-EMC with other existing strategies that support constraints. Two datasets

Table 8 Wilcoxon test for the results reported in Table 7

	Ranks			Test statistics	
	QWOA-EMC >	QWOA-EMC <	QWOA-EMC =	Z	Asymp. Sig. (2-tailed)
WOA	1	9	10	− 2.547	0.011
CS	0	14	3	− 3.308	0.001
APSO	2	7	2	− 1.487	0.137
DPSO	8	4	2	− 1.417	0.156
GS	3	11	6	− 1.775	0.076
QLSCA	9	5	0	− 1.516	0.129
IPOG	5	14	1	− 1.289	0.197
IPOG− D	1	19	0	− 3.212	0.001
PICT	0	20	0	− 3.923	0.000
TConfig	3	15	1	− 2.376	0.018
Jenny	0	20	0	− 3.925	0.000

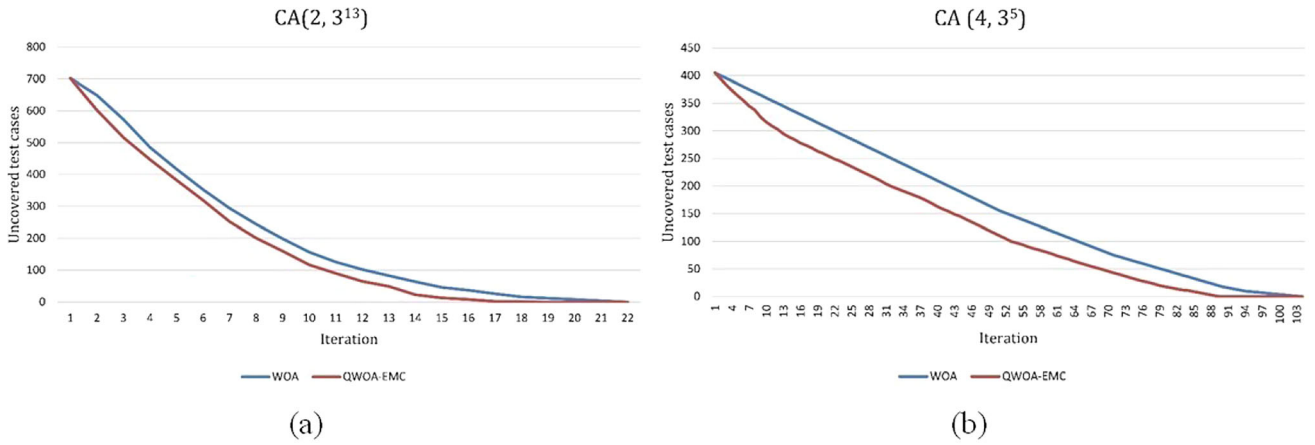

Fig. 8 The convergence behavior of QWOA-EMC for CA($N; 2, 3^{13}$) and CA($N; 4, 3^5$)

were subjected to experimentation. The first was based on the findings of our previous study [1] in which we developed and constructed the constraints. Three dataset groups were used in the experiments. The datasets' details are as follows:

1. The QWOA-EMC method was compared to six various algorithms, including the standard WOA algorithm using CCA($2, 3^P, F$), where the number of parameters was varied and their values ($v = 3$) and interaction strength ($t = 2$) were kept constant. In addition, the number of constraints (i.e., forbidden combinations) were varied between 3 and 5 pairs of constraints, as shown in Tables 9 and 10.
2. The QWOA-EMC method was compared to six various algorithms, including the standard WOA algorithm using CCA($2, v^7, F$), where the number of parameters ($p = 7$) and the interaction strength ($t = 2$) were kept constant and their values were varied. In addition, the number of constraints (i.e., forbidden combinations) was varied between 3 and 5 pairs of constraints, as shown in Tables 9 and 10.
3. The QWOA-EMC method was compared to six various algorithms, including the standard WOA algorithm using CCA($t, 2^{10}, F$), where the number of parameters and their values were kept constant ($p = 2$ and $v = 10$). While the interaction strength t was varied from 2 to 6. In addition, the number of constraints (i.e, forbidden combinations) were varied between 1 and 3 pairs of constraints, as shown in Tables 9 and 10.

While the second one was based on the results published in the works by Alsewari and Zamli [14]. PICT, TestCover, and mATEG_SAT are pure computational-based strategies, while the rest of the strategies are AI-based. The experimental results are presented in Table 11.

We compared QWOA-EMC to six other t-way approaches deployed to assess the QWOA-EMC strategy's performance. The size was one of the performance evaluation factors (i.e., optimal test suite size). Tables 9 and 10 exhibit the experimental outcomes; Table 9 displays the minimum

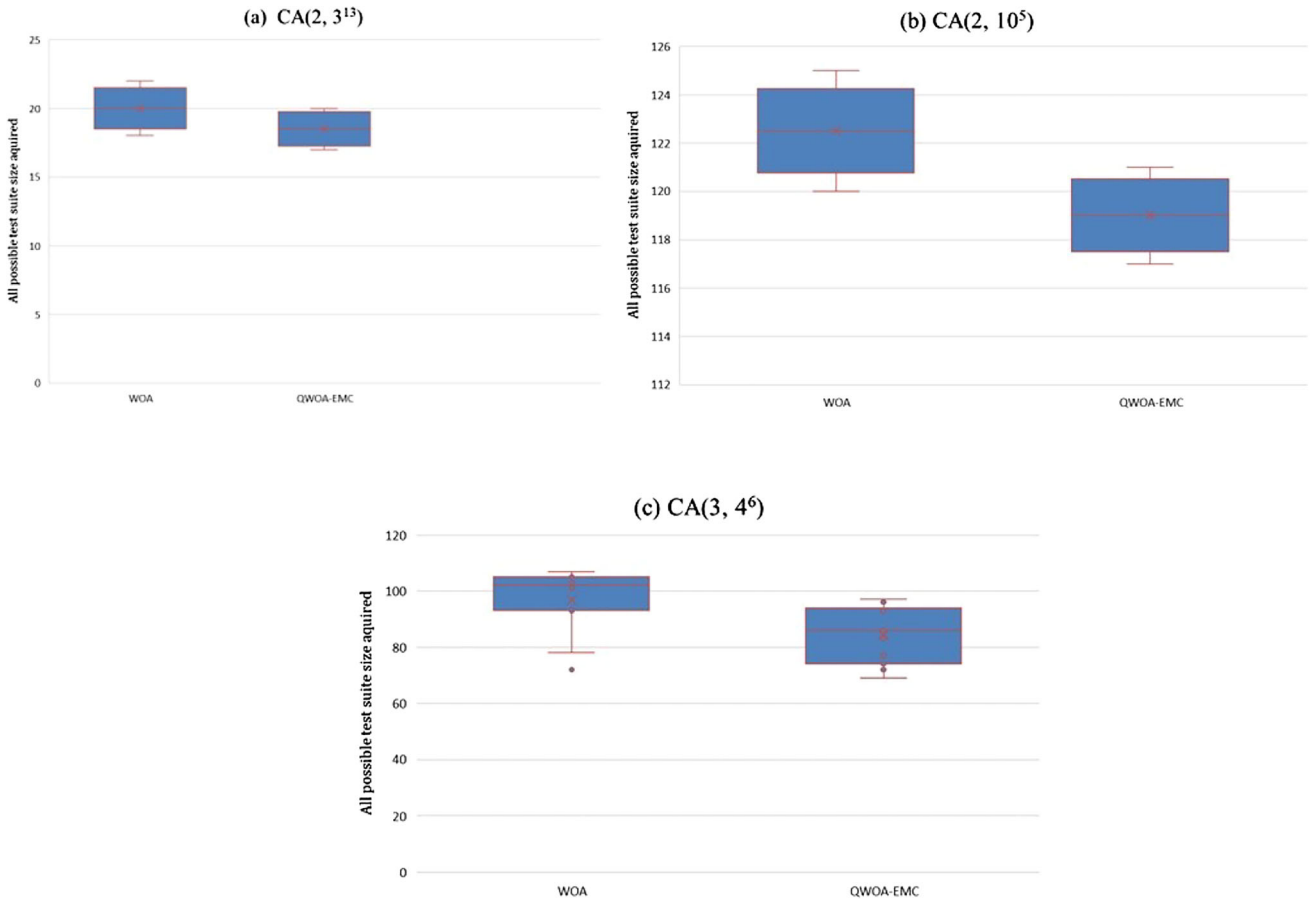

Fig. 9 The box plots analyses for CA($N; 2, 3^{13}$), CA($N; 2, 10^5$), and CA($N; 3, 4^6$)

(or best) test suite size, while Table 10 gives the average suite size for each competing strategy. Each strategy's best results are shown in bold and bolditalic.

The results of Tables 9 and 10 reveal that QWOA-EMC outperformed the six algorithms, including the standard WOA, for both the best and average test suites. It is because of QWOA-EMC's capacity to balance between global and local search, as determined by the Q-learning algorithm and based on historical performance.

In addition, we examine the QWOA-EMC strategy's performance by comparing it to the state-of-the-art. The performance evaluation factor was the size. Table 11 shows that QWOA-EMC performed better than pure computational-based strategies, HSS, LAHC, SA_SAT, and BTS. Comparing QWOA-EMC with PhABC, QWOA-EMC produced competitive results. However, PhABC supports constraints for pairwise testing (i.e., $t \leq 2$).

Furthermore, the convergence pattern analysis for the best solutions in Fig. 10 describes the convergence behavior of the QWOA-EMC and the WOA. The convergence of the test cases indicates that the QWOA-EMC converges faster than the WOA. Therefore, it has the potential to yield a smaller test suite size and avoid premature convergence.

The validity of our strategy can be argued further based on the scientific contributions mentioned earlier. Notably, the QWOA-EMC strategy has two characteristics. The first characteristic is the ability to escape local optima, as seen in Figs. 8 and 10. The QWOA-EMC strategy acquired this characteristic because it uses the EMC algorithm's acceptance probability, allowing the QWOA-EMC strategy to accept the worst solutions.

The second characteristic is the adaptive selecting factor, which is dependent on previous performance and is based on a cooperative penalized and rewarded model. Therefore, the three whale operators were selected based on an informed decision that led to having a balance between exploration and exploitation. Thus, the adaptive selecting factor has significantly improved the performance of the QWOA-EMC.

Table 9 Comparison of the implemented algorithms using best results of the three datasets

	System configurations	Forbidden combinations (F)	Implemented AI-based algorithms						
			LAHC	SCA	Jaya	FPA	CS	WOA	QWOA-EMC
Dataset one	CCA$(2, 3^3, F)$	$F = \{(C_{p_2,v_3}, C_{p_3,v_1}), (C_{p_2,v_2}, C_{p_3,v_1}), (C_{p_1,v_1}, C_{p_3,v_2})\}$	11	11	11	11	11	11	**10**
	CCA$(2, 3^4, F)$	$F = \{(C_{p_2,v_2}, C_{p_3,v_1}), (C_{p_2,v_3}, C_{p_4,v_2}), (C_{p_1,v_3}, C_{p_3,v_2}), (C_{p_1,v_2}, C_{p_2,v_1}, C_{p_4,v_3})\}$	**10**	**10**	**10**	**10**	**10**	**10**	**10**
	CCA$(2, 3^5, F)$	$F = \{(C_{p_2,v_1}, C_{p_3,v_1}), (C_{p_1,v_3}, C_{p_3,v_1}, C_{p_5,v_3}), (C_{p_2,v_2}, C_{p_4,v_2}, C_{p_5,v_1}), (C_{p_2,v_3}, C_{p_3,v_3})\}$	**11**	**11**	**11**	**11**	**11**	**11**	**11**
	CCA$(2, 3^6, F)$	$F = \{(C_{p_3,v_2}, C_{p_6,v_2}), (C_{p_5,v_3}, C_{p_6,v_1}), (C_{p_4,v_1}, C_{p_6,v_3}), (C_{p_2,v_1}, C_{p_4,v_3}, C_{p_5,v_1}, C_{p_6,v_2})\}$	**12**	**12**	13	**12**	13	**12**	**12**
	CCA$(2, 3^7, F)$	$F = \{(C_{p_3,v_2}, C_{p_4,v_2}), (C_{p_5,v_1}, C_{p_6,v_1}), (C_{p_6,v_3}, C_{p_7,v_3}), (C_{p_1,v_1}, C_{p_7,v_2}), (C_{p_2,v_1}, C_{p_7,v_1}), (C_{p_1,v_2}, C_{p_6,v_2})\}$	14	14	14	14	**13**	14	**13**
	CCA$(2, 3^8, F)$	$F = \{(C_{p_3,v_1}, C_{p_4,v_2}, C_{p_5,v_3}), (C_{p_6,v_1}, C_{p_7,v_1}, C_{p_8,v_1}), (C_{p_5,v_2}, C_{p_7,v_3}, C_{p_8,v_2}), (C_{p_1,v_3}, C_{p_7,v_3}, C_{p_8,v_2})\}$	15	**14**	15	**14**	**14**	**14**	**14**
	CCA$(2, 3^9, F)$	$F = \{(C_{p_2,v_2}, C_{p_3,v_2}, C_{p_6,v_2}, C_{p_7,v_2}, C_{p_9,v_1}), (C_{p_4,v_2}, C_{p_6,v_1}, C_{p_8,v_2}), (C_{p_1,v_1}, C_{p_2,v_1}, C_{p_7,v_3}, C_{p_8,v_3}, C_{p_9,v_3}), (C_{p_2,v_2}, C_{p_4,v_1}, C_{p_8,v_2})\}$	15	15	15	15	**14**	**14**	**14**
	CCA$(2, 3^{10}, F)$	$F = \{(C_{p_8,v_1}, C_{p_9,v_1}, C_{p_{10},v_1}), (C_{p_2,v_2}, C_{p_6,v_1}, C_{p_8,v_3}), (C_{p_4,v_3}, C_{p_6,v_3}, C_{p_8,v_3}, C_{p_{10},v_3}), (C_{p_1,v_2}, C_{p_5,v_1}, C_{p_6,v_2}, C_{p_6,v_3}, C_{p_7,v_3}, C_{p_9,v_2}, C_{p_{10},v_1})\}$	17	**16**	**16**	**16**	**16**	**16**	**16**
Dataset two	CCA$(2, 2^7, F)$	$F = \{(C_{p_1,v_1}, C_{p_2,v_1}), (C_{p_4,v_1}, C_{p_5,v_2})\}$	**7**	**7**	**7**	**7**	**7**	**7**	**7**
	CCA$(2, 3^7, F)$	$F = \{(C_{p_3,v_2}, C_{p_4,v_2}), (C_{p_5,v_1}, C_{p_6,v_1}), (C_{p_6,v_3}, C_{p_7,v_3}), (C_{p_2,v_1}, C_{p_3,v_3}, C_{p_6,v_2})\}$	14	14	**13**	14	14	14	**13**
	CCA$(2, 4^7, F)$	$F = \{(C_{p_3,v_2}, C_{p_4,v_4}), (C_{p_5,v_4}, C_{p_4,v_2}, C_{p_7,v_3}), (C_{p_7,v_1}, C_{p_2,v_1}, C_{p_1,v_2}), (C_{p_1,v_1}, C_{p_2,v_1})\}$	25	24	23	24	24	23	**22**
	CCA$(2, 5^7, F)$	$F = \{(C_{p_1,v_5}, C_{p_2,v_5}), (C_{p_3,v_4}, C_{p_4,v_3}), (C_{p_5,v_3}, C_{p_6,v_5}), (C_{p_6,v_1}, C_{p_7,v_1}), (C_{p_5,v_1}, C_{p_3,v_3}, C_{p1,v2}, C_{p7,v4})\}$	39	35	**33**	36	36	34	**33**
	CCA$(2, 6^7, F)$	$F = \{(C_{p_1,v_6}, C_{p_3,v_5}, C_{p_4,v_4}), (C_{p_6,v_6}, C_{p_7,v_6}), (C_{p_2,v_4}, C_{p_5,v_3}), (C_{p_1,v_2}, C_{p_4,v_1}, C_{p_6,v_2}, C_{p_5,v_2})\}$	54	49	49	49	51	49	**47**
	CCA$(2, 7^7, F)$	$F = \{(C_{p_1,v_2}, C_{p_2,v_2}, C_{p_3,v_1}, C_{p_4,v_1}), (C_{p_7,v_7}, C_{p_6,v_7}, C_{p_5,v_6}, C_{p_4,v_6}), (C_{p_4,v_3}, C_{p_5,v_4}, C_{p_6,v_5}), (C_{p_5,v_1}, C_{p_7,v_5})\}$	75	**65**	66	67	71	66	**65**
	CCA$(2, 8^7, F)$	$F = \{(C_{p_1,v_4}, C_{p_2,v_4}, C_{p_5,v_6}, C_{p_6,v_6}, C_{p_7,v_8}), (C_{p_2,v_3}, C_{p_3,v_3}, C_{p_4,v_2}, C_{p_5,v_2}, C_{p_6,v_1}, C_{p_7,v_1}), (C_{p_3,v_5}, C_{p_4,v_5}), (C_{p_6,v_7}, C_{p_5,v_7})\}$	98	**85**	86	86	93	**85**	**85**
Dataset three	CCA$(2, 2^{10}, F)$	$F = \{(C_{p_2,v_2}, C_{p_3,v_2}), (C_{p_5,v_1}, C_{p_6,v_1}), (C_{p_5,v_2}, C_{p_7,v_1}, C_{p_8,v_2})\}$	**8**	**8**	**8**	**8**	**8**	**8**	**8**
	CCA$(3, 2^{10}, F)$	$F = \{(C_{p_1,v_1}, C_{p_2,v_2}, C_{p_4,v_1}), (C_{p_5,v_2}, C_{p_7,v_1}, C_{p_8,v_2})\}$	17	**16**	**16**	17	17	17	**16**
	CCA$(4, 2^{10}, F)$	$F = \{(C_{p_1,v_2}, C_{p_2,v_2}, C_{p_3,v_2}, C_{p_4,v_1}), (C_{p_6,v_1}, C_{p_7,v_1}, C_{p_9,v_2}, C_{p_{10},v_2})\}$	**32**	34	33	34	34	**32**	**32**
	CCA$(5, 2^{10}, F)$	$F = \{(C_{p_8,v_1}, C_{p_8,v_2}, C_{p_{10},v_1}), (C_{p_4,v_2}, C_{p_6,v_2}, C_{p_7,v_2}), (C_{p_2,v_1}, C_{p_3,v_1})\}$	81	77	74	77	79	74	**73**
	CCA$(6, 2^{10}, F)$	$F = \{(C_{p_1,v_2}, C_{p_2,v_1}, C_{p_3,v_1}, C_{p_8,v_1}, C_{p_9,v_1}, C_{p_{10},v_1})\}$	157	158	**154**	157	**154**	**154**	**154**

Table 10 Comparison of the implemented algorithms using average results of the three datasets

	System configurations	Forbidden combinations (F)	Implemented AI-based algorithms						
			LAHC	SCA	Jaya	FPA	CS	WOA	QWOA-EMC
Dataset one	$CCA(2,3^3,F)$	$F=\{(C_{p_2,v_3},C_{p_3,v_1}),(C_{p_2,v_2},C_{p_3,v_1}),(C_{p_1,v_1},C_{p_3,v_2})\}$	11.13	11	11	11	11	11	**10**
	$CCA(2,3^4,F)$	$F=\{(C_{p_2,v_2},C_{p_3,v_1}),(C_{p_2,v_3},C_{p_4,v_2}),(C_{p_1,v_3},C_{p_3,v_2}),(C_{p_1,v_2},C_{p_2,v_1},C_{p_4,v_3})\}$	11.23	10.86	10.96	10.93	10.97	10.93	**10**
	$CCA(2,3^5,F)$	$F=\{(C_{p_2,v_1},C_{p_3,v_1}),(C_{p_1,v_3},C_{p_3,v_1},C_{p_5,v_3}),(C_{p_2,v_2},C_{p_4,v_2},C_{p_5,v_1}),(C_{p_2,v_3},\ C_{p_3,v_3})\}$	12.43	12.06	**12**	12.26	12.03	**12**	**12**
	$CCA(2,3^6,F)$	$F=\{(C_{p_3,v_2},C_{p_6,v_2}),(C_{p_5,v_3},C_{p_6,v_1}),(C_{p_4,v_1},C_{p_6,v_3}),(C_{p_2,v_1},C_{p_4,v_3},C_{p_5,v_1},\ C_{p_6,v_2})\}$	13.76	13.7	13.63	13.73	13.53	13.63	**13.49**
	$CCA(2,3^7,F)$	$F=\{(C_{p_3,v_2},C_{p_4,v_2}),(C_{p_5,v_1},C_{p_6,v_1}),(C_{p_6,v_3},C_{p_7,v_3}),(C_{p_1,v_1},C_{p_7,v_2}),(C_{p_2,v_1},\ C_{p_7,v_1}),(C_{p_1,v_2},C_{p_6,v_2})\}$	15.66	15.16	15.03	15.2	15.1	15.13	**14.89**
	$CCA(2,3^8,F)$	$F=\{(C_{p_3,v_1},C_{p_4,v_2},C_{p_5,v_3}),(C_{p_6,v_1},C_{p_7,v_1},C_{p_8,v_1}),(C_{p_5,v_2},C_{p_7,v_3},C_{p_8,v_2}),\ (C_{p_1,v_3},C_{p_7,v_3},C_{p_8,v_2})\}$	15.83	15.2	15.33	15.33	15.13	15.2	**15.09**
	$CCA(2,3^9,F)$	$F=\{(C_{p_2,v_2},C_{p_3,v_2},C_{p_6,v_2},C_{p_7,v_2},C_{p_9,v_1}),(C_{p_4,v_2},C_{p_6,v_1},C_{p_8,v_2}),(C_{p_1,v_1},C_{p_2,v_1},\ C_{p_7,v_3},C_{p_8,v_3},C_{p_9,v_3}),(C_{p_2,v_2},C_{p_4,v_1},C_{p_8,v_2})\}$	16.66	15.93	15.66	15.86	15.76	15.63	**15.51**
	$CCA(2,3^{10},F)$	$F=\{(C_{p_8,v_1},C_{p_9,v_1},C_{p_{10},v_1}),(C_{p_2,v_2},C_{p_6,v_1},C_{p_8,v_3}),(C_{p_4,v_3},C_{p_6,v_3},C_{p_8,v_3},\ C_{p_{10},v_3}),(C_{p_1,v_2},C_{p_5,v_1},C_{p_6,v_2},C_{p_6,v_3},C_{p_7,v_3},C_{p_9,v_2},C_{p_{10},v_1})\}$	17.4	16.7	16.36	16.8	16.76	16.46	.851, .851, .851**16.33**
Dataset two	$CCA(2,2^7,F)$	$F=\{(C_{p_1,v_1},C_{p_2,v_1}),(C_{p_4,v_1},C_{p_5,v_2})\}$	7.33	7.3	7.26	7.33	7.2	7.16	**7**
	$CCA(2,3^7,F)$	$F=\{(C_{p_3,v_2},C_{p_4,v_2}),(C_{p_5,v_1},C_{p_6,v_1}),(C_{p_6,v_3},C_{p_7,v_3}),(C_{p_2,v_1},C_{p_3,v_3},C_{p_6,v_2})\}$	15.3	**14.76**	14.93	15.03	**14.76**	14.9	**14.76**
	$CCA(2,4^7,F)$	$F=\{(C_{p_3,v_2},C_{p_4,v_4}),(C_{p_5,v_4},C_{p_4,v_2},C_{p_7,v_3}),(C_{p_7,v_1},C_{p_2,v_1},C_{p_1,v_2}),(C_{p_1,v_1},\ C_{p_2,v_1})\}$	26.2	24.75	24.93	25.1	25	24.93	**24.63**
	$CCA(2,5^7,F)$	$F=\{(C_{p_1,v_5},C_{p_2,v_5}),(C_{p_3,v_4},C_{p_4,v_3}),(C_{p_5,v_3},C_{p_6,v_5}),(C_{p_6,v_1},C_{p_7,v_1}),(C_{p_5,v_1},\ C_{p_3,v_3},C_{p1,v2},C_{p7,v4})\}$	40.56	36.5	36.24	37.16	37.43	36.83	**36.18**
	$CCA(2,6^7,F)$	$F=\{(C_{p_1,v_6},C_{p_3,v_5},C_{p_4,v_4}),(C_{p_6,v_6},C_{p_7,v_6}),(C_{p_2,v_4},C_{p_5,v_3}),(C_{p_1,v_2},C_{p_4,v_1},\ C_{p_6,v_2},C_{p_5,v_2})\}$	57.66	50.53	50.23	51.13	53.53	50.63	**50.09**
	$CCA(2,7^7,F)$	$F=\{(C_{p_1,v_2},C_{p_2,v_2},C_{p_3,v_1},C_{p_4,v_1}),(C_{p_7,v_7},C_{p_6,v_7},C_{p_5,v_6},C_{p_4,v_6}),(C_{p_4,v_3},\ C_{p_5,v_4},C_{p_6,v_5}),(C_{p_5,v_1},C_{p_7,v_5})\}$	77.56	67.8	67.56	68.7	72.43	67.43	**6.38**
	$CCA(2,8^7,F)$	$F=\{(C_{p_1,v_4},C_{p_2,v_4},C_{p_5,v_6},C_{p_6,v_6},C_{p_7,v_8}),(C_{p_2,v_3},C_{p_3,v_3},C_{p_4,v_2},C_{p_5,v_2},C_{p_6,v_1},\ C_{p_7,v_1}),(C_{p_3,v_5},C_{p_4,v_5}),(C_{p_6,v_7},C_{p_5,v_7})\}$	100.26	87.56	87.1	88.4	94.5	87.26	**87.07**
Dataset three	$CCA(2,2^{10},F)$	$F=\{(C_{p_2,v_2},C_{p_3,v_2}),(C_{p_5,v_1},C_{p_6,v_1}),(C_{p_5,v_2},C_{p_7,v_1},C_{p_8,v_2})\}$	8.33	8.29	8.23	8.17	**8.13**	**8.13**	**8.13**
	$CCA(3,2^{10},F)$	$F=\{(C_{p_1,v_1},C_{p_2,v_2},C_{p_4,v_1}),(C_{p_5,v_2},C_{p_7,v_1},C_{p_8,v_2})\}$	18.26	18.36	**17.66**	17.99	17.9	17.96	**17.66**
	$CCA(4,2^{10},F)$	$F=\{(C_{p_1,v_2},C_{p_2,v_2},C_{p_3,v_2},C_{p_4,v_1}),(C_{p_6,v_1},C_{p_7,v_1},C_{p_9,v_2},C_{p_{10},v_2})\}$	40	39.46	39.13	39.72	39.5	39.3	**39.04**
	$CCA(5,2^{10},F)$	$F=\{(C_{p_8,v_1},C_{p_8,v_2},C_{p_{10},v_1}),(C_{p_4,v_2},C_{p_6,v_2},C_{p_7,v_2}),(C_{p_2,v_1},C_{p_3,v_1})\}$	84.53	84.5	80.3	83	80.15	79.78	**79.73**
	$CCA(6,2^{10},F)$	$F=\{(C_{p_1,v_2},C_{p_2,v_1},C_{p_3,v_1},C_{p_8,v_1},C_{p_9,v_1},C_{p_{10},v_1})\}$	161.56	161.6	157.3	161.29	158.46	158.79	**158.26**

Table 11 Benchmarking QWOA-EMC Constraints with Other Strategies

Systems configurations	Forbidden combinations (F)	HSS [14]	LAHC [32]	SA_SAT [33]	mATEG_SAT [33]	PICT [33]	TestCover [33]	BTS [34]	PhABC [35]	QWOA-EMC	
		Best	Best	Best	Best	Best	Best	Best	Best	Best	Mean
CCA$(2, 3^3, F)$	$F = \{(C_{p_2,v_3}, C_{p_3,v_1}), (C_{p_2,v_2}, C_{p_3,v_1}), (C_{p_1,v_1}, C_{p_3,v_2}), (C_{p_1,v_3}, C_{p_2,v_3}), (C_{p_1,v_3}, C_{p_3,v_3}), (C_{p_1,v_2}, C_{p_2,v_3}, C_{p_3,v_3})\}$	10	10	10	10	10	10	10	**9**	10	10
CCA$(2, 4^3, F)$	$F = \{(C_{p_1,v_1}, C_{p_2,v_2}), (C_{p_1,v_3}, C_{p_3,v_4}), (C_{p_1,v_4}, C_{p_2,v_4}, C_{p_3,v_1}), (C_{p_1,v_3}, C_{p_2,v_2})\}$	**16**	17	17	17	19	17	**16**	**16**	**16**	16.86
CCA$(2, 5^3, F)$	$F = \{(C_{p_1,v_2}, C_{p_2,v_2}), (C_{p_1,v_5}, C_{p_3,v_3}), (C_{p_1,v_5}, C_{p_3,v_5}), (C_{p_1,v_5}, C_{p_2,v_4}, C_{p_3,v_2}), (C_{p_1,v_5}, C_{p_2,v_3}), (C_{p_1,v_2}, C_{p_2,v_4})\}$	26	**25**	26	26	27	30	**25**	**25**	**25**	26.76
CCA$(2, 6^3, F)$	$F = \{(C_{p_1,v_4}, C_{p_2,v_6}), (C_{p_2,v_4}, C_{p_3,v_5}), (C_{p_1,v_3}, C_{p_2,v_1}), (C_{p_2,v_2}, C_{p_3,v_3}), (C_{p_1,v_4}, C_{p_3,v_2}), (C_{p_2,v_4}, C_{p_3,v_2}), (C_{p_1,v_6}, C_{p_2,v_5}, C_{p_3,v_5})\}$	36	36	36	37	39	38	36	36	**34**	36.3
CCA$(2, 7^3, F)$	$F = \{(C_{p_2,v_1}, C_{p_3,v_6}), (C_{p_1,v_6}, C_{p_2,v_6}, C_{p_3,v_4}), (C_{p_1,v_5}, C_{p_3,v_1}), (C_{p_1,v_7}, C_{p_2,v_5}), (C_{p_1,v_2}, C_{p_2,v_5}), (C_{p_1,v_7}, C_{p_2,v_4})\}$	51	53	52	52	56	54	51	**50**	**50**	51.4
CCA$(3, 5^4, F)$	$F = \{(C_{p_1,v_4}, C_{p_3,v_3}, C_{p_4,v_2}), (C_{p_2,v_2}, C_{p_4,v_5}), (C_{p_1,v_3}, C_{p_2,v_4}), (C_{p_1,v_2}, C_{p_3,v_4})\}$	139	141	140	138	143	NS	NS	NS	**137**	141.2
CCA$(3, 6^4, F)$	$F = \{(C_{p_1,v_5}, C_{p_4,v_2}), (C_{p_3,v_3}, C_{p_4,v_1}), (C_{p_2,v_2}, C_{p_4,v_2}), (C_{p_2,v_1}, C_{p_3,v_2})\}$	**238**	245	251	241	250	NS	NS	NS	**238**	241.59
CCA$(3, 7^4, F)$	$F = \{(C_{p_2,v_3}, C_{p_3,v_7}), (C_{p_2,v_6}, C_{p_3,v_7}), (C_{p_2,v_5}, C_{p_3,v_3}), (C_{p_2,v_1}, C_{p_3,v_2}, C_{p_4,v_6}), (C_{p_3,v_3}, C_{p_4,v_5}), (C_{p_1,v_3}, C_{p_3,v_7})\}$	**377**	395	438	383	401	NS	NS	NS	381	385.12

6.4 Applicability and effectiveness of QWOA-EMC for combinatorial test data generation

In order to demonstrate the applicability and effectiveness of the t-way test data generation by QWOA-EMC, we have developed a variant Java program, called *college_acceptance*, from the original program as described in [36]. Highly intertwined program with many interacting input variables, the hypothetical *college_acceptance* program grants acceptance for college admissions in four major departments: the Department of Mathematics, the Department of Physics, the Department of Biology, and the Department of Chemistry. The department's acceptance criteria are based on a student's high school grades in six subjects: English, mathematics, physics, biology, chemistry, and computer science. In this hypothetical situation, a student can only be accepted into one of the departments if the following conditions are met:

- He/She passes all six subjects (i.e., a score of 50% or better must be achieved in each subject).
- He/She scores 75% or better in the subjects related to the department they are applying for.
- The acceptance will be conditional if the English subject is less than 75%.

The *college_acceptance* program comprises three classes, three methods, and 98 lines. The *main*(), *checkBoundary*(), and *testAcceptance*() are the three developed methods. The *testAcceptance*() method covers six type double parameters corresponding to each subject's scores. The admission status is produced and printed based on the subject scores and the criteria mentioned earlier. The status can be incorrect grades, not accepted in any department, conditionally accepted in a specific department, or accepted in a particular department.

Based on equivalence partitioning techniques, the grade level can be partitioned into two primary sets: the valid value set and the out-of-range value set (i.e., robustness). The valid set can further be subdivided into three intervals: [0, 50), [50, 75), and [75,100]. Similarly, the out-of-range values can be partitioned into two intervals: values less than 0 and values greater than 100. Table 12 summarizes the base values chosen for each valid and out-of-range value set. For example, the first value, 49, belongs to the valid set's first interval. The second value, 74, belongs to the second interval, and the third value, 76, belongs to the valid set's final interval. The fourth value, -1, belongs to the out-of-range set's first interval. Finally, the fifth value, 101, belongs to the out-of-range set's second interval.

Using the based data values in Table 12, we initially generated an exhaustive test suite at $t = 6$ for both valid and out-of-range sets. Then, we continued to generate test suites for $t = 5$ until $t = 2$. All the test suites were

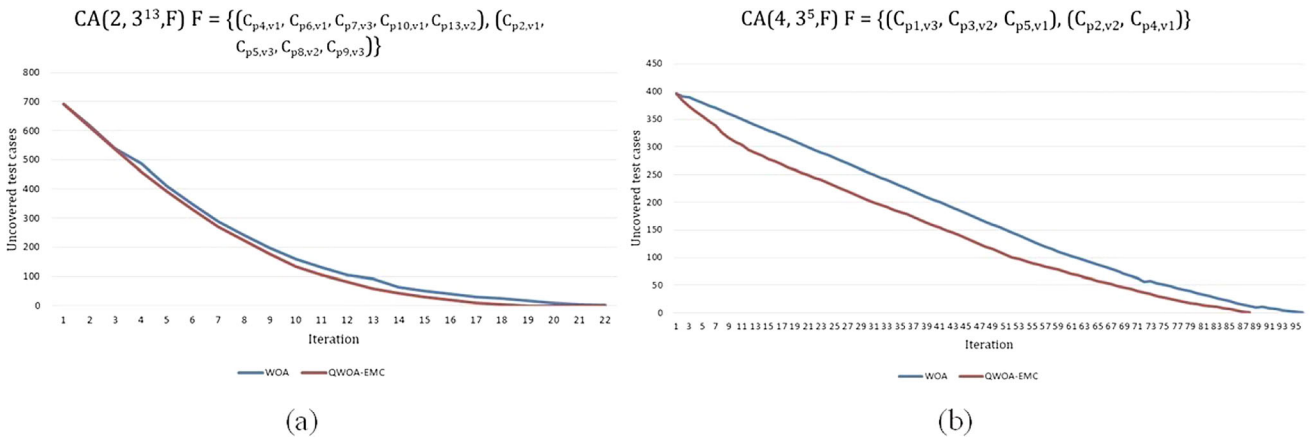

Fig. 10 The convergence behavior of QWOA-EMC for CA($N; 2, 3^{13}$) and CA($N; 4, 3^5$) with constraints

generated using QWOA-EMC. For each t, we executed the test suite against the *college_acceptance* program and measured four types of coverage, mainly class coverage (i.e., the ratio of covered classes to the total number of classes), method coverage (i.e., to the ratio of the covered methods to the total number of methods), branch coverage (i.e., the number of executed branches divided by the total number of branches), and line coverage (i.e., is the ratio of covered lines to the total number of lines). We used the open-source coverage tool EMMA [37] to help us measure coverage. The complete coverage measurement result can be seen in Table 13. A graphical depiction of Table 13 can be seen in Fig. 11.

Referring to Fig. 11, two subtle observations can be made. Firstly, the same 100 % coverages for class, method, branch, and line can be achieved at $t = 4$ without having to go to exhaustive testing (i.e., $t = 6$). More precisely, QWOA-EMC has successfully managed to minimize the number of test cases from 793 (729+64) at $t = 6$ (i.e., all possible test cases) to 161 (133+28) at $t = 4$ with a total reduction of 79.69%. Secondly, based on our test suite execution from $t = 6$ until $t = 2$, we observed no variations from the specification (i.e., there are no bugs). For these reasons, QWOA-EMC can be usefully adopted to minimize the combinatorial test data for testing purposes, as well as being able to complement existing conventional testing strategies (e.g., from equivalence partitioning, boundary values, and robustness).

6.5 Threats to validity

Few threats to the validity of empirical and experimental research are expected. A couple of threats can be raised in the course of our study. The first threat is the impartiality of the benchmark experiments using meta-heuristic-based techniques may be a concern. Due to the lack of source codes, many comparisons with similar work rely primarily on published results. Returning to Table 4, the only values that can be attained are max iteration and population size. These parameters' values could be enough if we were dealing with a single-based solution problem. However, t-way testing is a population-based solution problem where the outer loop is required to achieve convergence. As a result, the maximum number of fitness function evaluations ($F_{\max}$ = number of iteration for convergence X population size X max iteration) cannot be precisely estimated because it depends on each algorithm's convergence process. Thus, the maximum number of fitness function evaluations (Fmax) for all strategies is not guaranteed.

The second threat is the statistical analysis based on the best-reported values rather than the mean or average values (as the mean values are not highlighted in most published results). The primary challenge is that some of the best results (especially in the case of meta-heuristic-based techniques) may be achieved by luck, impacting the conclusion.

Table 12 Generated test-based values using Equivalence Partitioning

	Math	Physics	Biology	Chemistry	English	Computer Science
Valid values	49	49	49	49	49	49
	74	74	74	74	74	74
	76	76	76	76	76	76
Out-of-range values	− 1	− 1	− 1	− 1	− 1	− 1
	101	101	101	101	101	101

Table 13 Coverage Measurement

t	Number of test cases		Class coverage (%)	Method coverage (%)	Branch coverage (%)	Line coverage (%)
	Valid Set	Out-of-range set				
2	14	7	100	100	38.8	61.1
3	44	12	100	100	54.1	88.9
4	133	28	100	100	100	100
5	292	32	100	100	100	100
6	729	64	100	100	100	100

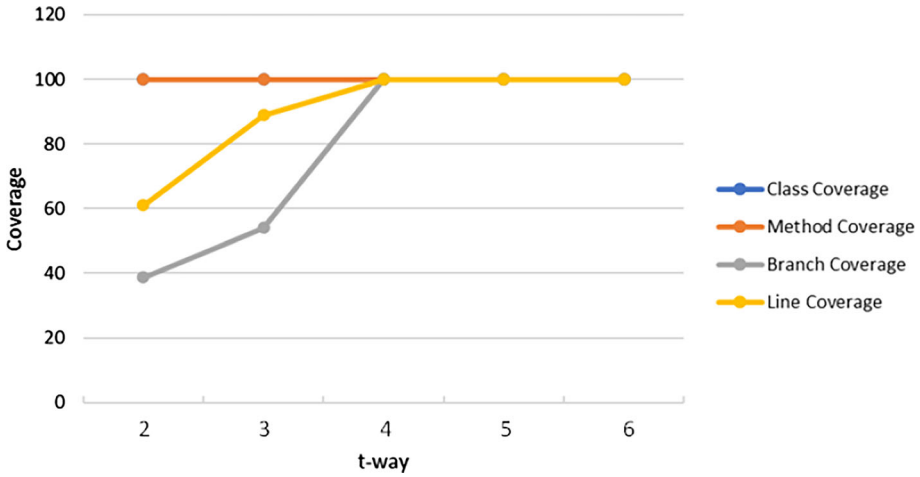

Fig. 11 Overall coverage measurement

7 Conclusion

This study proposed the QWOA-EMC strategy for test suite generation that supports constraints, unlike most AI-based approaches. The proposed QWOA-EMC strategy was developed by ensembling the *Q*-learning algorithm with Whale Optimization Algorithm. The *Q*-learning algorithm replaced the controlling parameters of WOA to make an informed decision based on past performance and thus, achieve a balance between exploration and exploitation. Additionally, the acceptance probability of the Exponential Monte Carlo algorithm was embedded to accept the worst solution with a certain degree to escape local optima.

The experiments in Sect. 6 indicated that the proposed QWOA-EMC effectively managed to overcome the issue of premature convergence and escaped the local optima. Moreover, QWOA-EMC performed better than standard WOA by statistically generating smaller test suite sizes (i.e., produce better solutions). Furthermore, the QWOA-EMC performed better than the pure computational-based strategies by statistically generating smaller test suite sizes. Further, it outperformed most existing AI-based strategies by conducting various experiments on different configurations. Additionally, the experimental results illustrated that QWOA-EMC could support constrained *t*-way testing while competing against other state-of-the-art strategies in terms of efficiency and performance.

Therefore, it can be concluded that the advantages of QWOA-EMC help improve the performance and overcome the WOA's limitations stated earlier. Such advantages include (1) the ability to learn from prior performance and (2) the ability to accept the worst solution to a certain degree to allow further exploration and escape from local optima.

In future work, given our promising results, we expect to expand the QWOA-EMC strategy as a multi-objective optimization method for combinatorial testing and to support variable strength *t*-way testing and higher interaction strengths up to 20. Additionally, we expect to expand our approach by using the iterative reproducing kernel method (IRKM), a powerful and promising technique for finding approximate solutions [38].

Acknowledgements This work was supported by the Ministry of Education, Malaysia (FRGS /1/2019/ICT02/UKM/01/1), and the Universiti Kebangsaan Malaysia (DIP-2016-024). Ali Abdullah Hassan would like to express his gratitude to Hadhramout Foundation in Yemen for their tuition fee support.

Availability of data and materials The authors confirm that the data supporting the findings of this study come from different sources. The sources are as follows:The data used in Sect. 5.2 are available within the article [9] and its supplementary materials..The data used in Sect. 5.3 are available within the article [4] and its supplementary materials.

Declaration

Conflict of interest The authors declare that they have no known competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

References

1. Hassan AA, Abdullah S, Zamli KZ, Razali R (2020) Combinatorial test suites generation strategy utilizing the whale optimization algorithm. IEEE Access 8:192288–192303
2. Abdullah S, Sabar NR, Nazri MZA, Ayob M (2014) An exponential Monte-Carlo algorithm for feature selection problems. Comput Ind Eng 67:160–167
3. Sabar, N. R., Ayob, M., & Kendall, G. (2009). Tabu exponential monte-carlo with counter heuristic for examination timetabling. In 2009 IEEE Symposium on Computational Intelligence in Scheduling (pp. 90-94). IEEE.
4. Hassan AA, Abdullah S, Zamli KZ, Razali R (2022) Whale optimization algorithm strategies for higher interaction strength *t*-way testing. Comput Mater Cont 73(1):2057–2077. https://doi.org/10.32604/cmc.2022.026310
5. Colbourn CJ (2004) Combinatorial aspects of covering arrays. Le Matematiche 59(1,2):125–172
6. Behmanesh R, Rahimi I, Gandomi AH (2021) Evolutionary many-objective algorithms for combinatorial optimization problems: a comparative study. Arch Comput Meth Eng 28(2):673–688
7. Kassaymeh S, Abdullah S, Al-Laham M, Alweshah M, Al-Betar MA, Othman Z (2021) Salp swarm optimizer for modeling software reliability prediction problems. Neural Process Lett 53(6):4451–4487
8. Muazu AA, Hashim AS, Sarlan A (2022) Review of nature inspired metaheuristic algorithm selection for combinatorial *t*-way testing. IEEE Access 10:27404–27431
9. Esfandyari S, Rafe V (2018) A tuned version of genetic algorithm for efficient test suite generation in interactive *t*-way testing strategy. Inf Softw Technol 94:165–185
10. Sabharwal S, Bansal P, Mittal N, Malik S (2016) Construction of mixed covering arrays for pair-wise testing using probabilistic approach in genetic algorithm. Arab J Sci Eng 41(8):2821–2835
11. Bansal P, Sabharwal S, Mittal N, Arora S (2015) Construction of variable strength covering array for combinatorial testing using a greedy approach to genetic algorithm, e-Inf Softw Eng J 9(1)
12. Wu H, Nie C, Kuo F-C, Leung H, Colbourn CJ (2014) A discrete particle swarm optimization for covering array generation. IEEE Trans Evolut Comput 19(4):575–591
13. Ahmed BS, Abdulsamad TS, Potrus MY (2015) Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm. Inf Softw Technol 66:13–29
14. Alsewari ARA, Zamli KZ (2012) Design and implementation of a harmony-search-based variable-strength *t*-way testing strategy with constraints support. Inf Softw Technol 54(6):553–568
15. Zamli KZ, Din F, Ahmed BS, Bures M (2018) A hybrid q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem. PloS One 13(5):e0195675
16. Mahmoud T, Ahmed BS (2015) An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use. Exp Syst Appl 42(22):8753–8765
17. Zamli KZ, Ahmed BS, Mahmoud T, Afzal W (2018) Fuzzy adaptive tuning of a particle swarm optimization algorithm for variable-strength combinatorial test suite generation, arXiv preprint arXiv:1810.05824
18. Zamli KZ, Din F, Baharom S, Ahmed BS (2017) Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength *t*-way test suites. Eng Appl Artif Intell 59:35–50
19. Nasser AB, Zamli KZ, Ahmed BS (2019) Dynamic solution probability acceptance within the flower pollination algorithm for t-way test suite generation, arXiv preprint arXiv:1902.11160
20. Htay KM, Othman RR, Amir A, Alkanaani JMH (2021) Gravitational search algorithm based strategy for combinatorial t-way test suite generation. J King Saud Univ Comput Inf Sci 34(8):4860–4873
21. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evolut Comput 1(1):67–82
22. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
23. Watkins CJ, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292
24. Samma H, Lim CP, Saleh JM (2016) A new reinforcement learning-based memetic particle swarm optimizer. Appl Soft Comput 43:276–297
25. Alazzawi AK, Rais HM, Basri S (2019) Abcvs: an artificial bee colony for generating variable *t*-way test sets. Int J Adv Comput Sci Appl
26. Jenkins B (2016) Jenny test tool. http://www.burtleburtle.net./bob/math/jenny.html
27. Williams AW (2000) Determination of test configurations for pair-wise interaction coverage. In: Testing of communicating systems, Springer, pp 59–74
28. Czerwonka J, Butt D, Gens C (2006) Pairwise testing in real word: practical extensions to test case generators. In: Proceedings of the 24th pacific northwest software quality conf, Vol. 2006
29. Lei Y, Kacker R, Kuhn DR, Okun V, Lawrence J (2008) Ipog/ipog-d: efficient test generation for multi-way combinatorial testing. Softw Test Verif Reliab 18(3):125–148
30. Lei Y, Kacker R, Kuhn DR, Okun V, Lawrence J (2007) Ipog: a general strategy for *t*-way software testing. In: 14th Annual IEEE international conference and workshops on the engineering of computer-based systems (ECBS'07), IEEE, pp 549–556
31. Torres-Jimenez J, Perez-Torres JC (2019) A greedy algorithm to construct covering arrays using a graph representation. Inf Sci 477:234–245
32. Alsewari A, Zamli KZ, Al-Kazemi B (2015) Generating *t*-way test suite in the presence of constraints. J Eng Technol (JET) 6(2):52–66
33. Cohen MB, Dwyer MB, Shi J (2007) Interaction testing of highly-configurable systems in the presence of constraints. In: Proceedings of the 2007 international symposium on Software testing and analysis, pp 129–139
34. Alsariera YA, Ahmed HAS, Alamri HS, Majid MA, Zamli KZ (2018) A bat-inspired testing strategy for generating constraints pairwise test suite. Adv Sci Lett 24(10):7245–7250

35. Alazzawi AK, Rais HM, Basri S, Alsariera YA (2019) Phabc: a hybrid artificial bee colony strategy for pairwise test suite generation with constraints support. In: 2019 IEEE student conference on research and development (SCOReD), IEEE, pp 106–111
36. Zamli KZ, Klaib MF, Younis MI, Isa NAM, Abdullah R (2011) Design and implementation of a *t*-way test data generation strategy with automated execution tool support. Inf Sci 181(9):1741–1758
37. Vlad-Roubtsov, Emma: a free java code coverage tool (2006). http://emma.sourceforge.net
38. Altawallbeh Z, Al-Smadi M, Komashynska I, Ateiwi A (2018) Numerical solutions of fractional systems of two-point bvps by using the iterative reproducing kernel algorithm. Ukrain Math J 70(5):687–701