A novel Fruit Fly Optimization Algorithm with quasi-affine transformation evolutionary for numerical optimization and application

Ru-Yu Wang¹, Pei Hu¹,², Chia-Cheng Hu³ and Jeng-Shyang Pan¹,⁴

Abstract
The Fruit Fly Optimization Algorithm is a swarm intelligence algorithm with strong versatility and high computational efficiency. However, when faced with complex multi-peak problems, Fruit Fly Optimization Algorithm tends to converge prematurely. In response to this situation, this article proposes a new optimized structure—Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm. The new algorithm uses the evolution matrix in QUasi-Affine TRansformation Evolution algorithm to update the position coordinates of particles. This strategy makes the movement of particles more scientific and the search space broader. In order to prove its effectiveness, we compare Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm with five other mature intelligent algorithms, and test them on 22 different types of benchmark functions. In order to observe the multi-faceted performance of Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm more intuitively, we also conduct experiments on algorithm convergence analysis, the Friedman test, the Wilcoxon signed-rank test, and running time comparison. Through the above several comparative experiments, Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm has indeed demonstrated its strong competitiveness. Finally, we apply it to Capacitated Vehicle Routing Problem. Through comparing with the contrast algorithms, it is confirmed that Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm can achieve better vehicle routes planning.

Keywords
Intelligence algorithm, Fruit Fly Optimization Algorithm, evolution matrix, Quasi-affine Transformation Evolution Algorithm, Capacitated Vehicle Routing Problem

Date received: 19 October 2021; accepted: 17 December 2021

Handling Editor: Yanjiao Chen

Introduction
Generally, the optimization problem is to select some parameters, so that the design index can reach the optimal value under a series of related constraints. It can usually be expressed as a problem in the form of mathematical programming.¹,² Metaheuristic algorithm is an efficient optimization method based on computational intelligence. It has strong versatility and can solve various continuous problems as well as discrete problems.³⁻⁵ Up to now, there have been many mature
metaheuristic algorithms with good performance, such as Sparrow Search Algorithm (SSA), Genetic Algorithm (GA), Butterfly Optimization Algorithm (BOA), Differential Evolution (DE), Cuckoo Search (CS), Harris Hawk Optimization (HHO), Gray Wolf Optimization (GWO), Fish Migration Optimization (FMO), Particle Swarm Optimization (PSO), Phasmatodea Population Evolution algorithm (PPE), Cat Swarm Optimization (CSO), and Ant Colony Optimization (ACO).

The group fly to this place quickly. This can easily induce the algorithm to fall into a local extremum. The optimization process of fruit flies. It was proposed by Pan in 2012. However, the optimization process of fruit fly relies heavily on its sense of smell. It first uses the sense of smell to identify the direction in which the food is most likely to exist. Then, many other individuals in the group fly to this place quickly. This can easily induce the algorithm to fall into a local extremum. Therefore, for this shortcoming, many scholars had made efforts. In 2015, Wang et al. proposed an elite selection mechanism and applied the algorithm to joint supply problem. Next, in 2016, Zheng and Wang created a dual-strategy mechanism to allow FOA to carry out more detailed local exploitation. Then, in 2018, Kanarachos et al. and his team adopted a dynamic multi-decision method, so that the algorithm had the ability to self-adjust and improved the diversity of solutions in the iterative process.

QUasi-Affine TRansformation Evolution (QUATRE) is an excellent optimization evolution structure proposed by Pan et al. in 2016. The position update of its search agent uses an affine transformation formula. Then, Meng et al. compared the performance of QUATRE with various variants of PSO and DE, and demonstrated the powerful global optimization capabilities of it. In view of the good performance of QUATRE, many scholars had proposed a lot of algorithms based on it. In 2020, Du et al. introduced the timestamp mutation strategy to QUATRE, so the performance of the algorithm was further promoted. In 2018, Meng and Pan and his partners proposed a multi-group multi-selection communication strategy, which greatly improved the ability of the original QUATRE to perform global search, and applied it to the received signal strength indication (RSSI)-based wireless sensor network (WSN) node positioning problem.

In this article, combining FOA and QUATRE, a new intelligent algorithm is proposed—Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm (QTFOA). At first, QTFOA randomly allocates the location coordinates of search agents based on the mechanism of FOA, and determines the global optimal individual. In this way, randomly initializing the positions of the particles can greatly enhance the distribution range of the population, which is conducive to the subsequent global optimization. After entering the iteration, the algorithm uses evolution matrix in QUATRE to update the positions of particles. QUATRE’s unique affine transformation matrix allows particles to affine from one position to another, so that the jump-type position transformation can make the movement of the particles more scientific and have a wider range of optimization. In this way, the evolutionary method in FOA can perform local exploitation in detail without any worries. Because it largely does not have to worry about falling into a local optimum. This combination of FOA and QUATRE not only provides a guarantee for the global exploration but also lays the foundation for the local exploitation.

Capacitated Vehicle Routing Problem (CVRP) is a typical problem in transportation. Its foundation is Vehicle Routing Problem (VRP), which is a common problem in life. CVRP is a further expansion of VRP, which adds capacity restriction condition on the basis of original problem. There are already many algorithms for solving CVRP. Among them, the metaheuristic algorithm is particularly prominent. Metaheuristic algorithm can use search agents to quickly find the optimal path. Examples of good results include Sine Cosine Algorithm (SCA), Bee Algorithm (BA), Tabu Search algorithm (TS), Simulated Annealing algorithm (SA), and so on. Here, we applied QTFOA to this practical scenario and compared it with several other mature algorithms. It was finally confirmed that QTFOA could always find the shortest path.

There are five remaining sections in this article. Section “Related works” is an introduction of the theoretical basis of FOA and QUATRE. Section “The proposed QTFOA” details the operating mechanism of the newly proposed QTFOA. Section “Simulation experiment and result analysis” presents the experimental results of a series of performance tests. Section “Application of QTFOA in CVRP” verifies the excellent effect of QTFOA on CVRP. Section “Conclusion” summarizes the whole article and indicates the potential shortcomings of QTFOA.

Related works

FOA

Since modern times, fruit fly has become a model organism that is very popular among researchers. It has outstanding sense of smell and vision. After the fruit flies perceive the concentration of the food odor, they will send information to their companions. Similarly, each search agent also receives the odor concentration
information transmitted by the companions. In this way, fruit flies can compare the odor concentration of various parties, and then use their excellent vision to fly to the location with highest concentration. Repeat this process continuously, and eventually they have a good chance of finding food. To unify the standard, this article only studies the minimization problem. The following describes the specific steps of FOA:

1. Position initialization: there are \( ps \) search agents. The moving direction and distance of them are initialized around the best one in population casually. Equation (1) is used to solve the location of each search agent

\[
\begin{align*}
X_i &= X_{axis} + \text{RandomValue} \\
Y_i &= Y_{axis} + \text{RandomValue}
\end{align*}
\]

The position of the \( i \)th search agent is represented by \( X_i \) and \( Y_i \) is optimal position of the population. \( \text{RandomValue} \) is a random search distance.

2. Calculate taste concentration judgment value: initially, the location of food is not known, so we reckon the distance \( \text{Dist}_i \) between each search agent and origin. Later, we count the taste concentration judgment value \( S_i \) at each particle’s location. The calculation formula is shown in equations (2) and (3)

\[
\text{Dist}_i = \sqrt{X_i^2 + Y_i^2} \quad (2)
\]

\[
S_i = \frac{1}{\text{Dist}_i} \quad (3)
\]

3. Calculate taste concentration value: we calculate the taste concentration \( \text{Smelli} \) of each one by equation (4)

\[
\text{Smelli} = \text{Function}(S_i) \quad (4)
\]

4. Choose the best individual: we choose the best search agent which has the global optimal concentration value \( \text{bestSmell} \) in the population

\[
[\text{bestSmell}, \text{bestIndex}] = \min(\text{Smelli}) \quad (5)
\]

5. Fly to food: other individuals in the group will fly to the best individual.

6. Iterative optimization: repeat Steps 1–4 and continuously update the global optimal individual.

Other fruit flies always fly to the best individual in each generation.

Through understanding the optimization process of fruit flies, combined with the existing references, the following three shortcomings of FOA can be summarized:

1. The most significant shortcoming of FOA is that the optimization process of it is blind and random, which makes the algorithm converge too fast and the population diversity is insufficient in the later stage.

2. According to equations (2) and (3), we see FOA implicitly stipulates optimal solution can only be a positive number. This largely limits the scope of the algorithm.

3. There is no certain mathematical formula to express the position relationship between the current generation and previous generation. So, the search agent’s location cannot be accurately described.

**QUATRE**

The position update process in QUATRE is similar to affine transformation in geometric transformation, which uses mathematical equations to linearly transform coordinates from one affine space to another. This process is shown in equation (6)

\[
X_{G+1} = X_G + \tilde{M} \otimes B \quad (6)
\]

where \( X \) is the coordinate matrix of population, \( X = [x_1, x_2, ..., x_{ps}]^T \), and \( G \) represents the current iteration. For each particle, \( x_i = [x_{i1}, x_{i2}, ..., x_{iD}] \). \( D \) is the dimension. \( \otimes \) is a symbol for component multiplication.

\( B \) can be regarded as a mutation matrix, and there are six representation methods, as shown in Table 1. \( F \) is an important scale factor and in this article, it is set to 0.7. \( X_{r1,G}, X_{r2,G}, X_{r3,G}, X_{r4,G}, X_{r5,G} \) are matrices obtained by randomly transforming row vector of \( X \). \( X_{gbest,G} \) is composed of coordinate vector of the optimal particle position in current generation, as shown in equation (7)

\[
X_{gbest,G} = \left[ \begin{array}{c} x_{gbest,G} \\ x_{gbest,G} \\ \vdots \\ x_{gbest,G} \end{array} \right] \quad (7)
\]

\( M \) is a matrix of size \( ps \times D \), \( \tilde{M} \) is its binary inverse matrix, as shown in equation (8). \( M \) is derived from the lower triangular matrix \( M_{imp} \). First, for each row of \( M_{imp} \), \( D \) column elements are randomly scrambled, and then this matrix arbitrarily transforms the row vector.
According to the value of population size. Suppose \( ps \) is shown in equation (1). At this time, two population random directions and distances. The position solution other individuals fly around the best search agent with 

\[ M_{imp} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \sim \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = M \]  

(9)

In this way, \( M \) can be obtained, and the transformation process from \( M_{imp} \) to \( M \) is shown in equation (9)

We use equations (2) and (3) to calculate the taste concentration judgment value, and we use equation (4) to calculate the taste concentration value. \( new_X \) and \( new_Y \) are used to retain the position information of the current global optimal individual

\[ \begin{aligned} new_X &= X(\text{bestIndex}, :) \\ new_Y &= Y(\text{bestIndex}, :) \end{aligned} \]  

(13)

**Position change**

Next, the algorithm enters the core. We change the particle's position with the affine transformation evolution matrix in QUATRE, and the specific process is as follows:

1. Generate matrices \( M \) and \( \tilde{M} \) according to the principles of equations (10) and (8).
2. In order to strengthen the randomness of the particles, it is preferred to generate mutation matrices \( B_t \) and \( B_f \) according to the second mutation strategy in Table 1. Next, we randomly transform the row vectors of \( X \) and \( Y \) to generate six random matrices. Among them, \( posrx, posr1 \), and \( posr2 \) are obtained corresponding to \( X \). \( posy, posr3 \), and \( posr4 \) are obtained corresponding to \( Y \). The two mutation matrices \( B_t \) and \( B_f \) serve for the transformation of \( X \) and \( Y \), respectively. The generation process is shown in equations (14) and (15)

\[ B_t = posrx + F^*(posr1 - posr2) \]  

(14)

\[ B_f = posy + F^*(posr3 - posr4) \]  

(15)

**The proposed QTFOA**

**Initialization**

Like the original FOA, the position is initialized, and other individuals fly around the best search agent with random directions and distances. The position solution is shown in equation (1). At this time, two population position matrices of size \( ps \times D \) are generated, which are defined as \( X \) and \( Y \) shown in equations (11) and (12)

\[ X = [x_1, x_2, \ldots, x_{ps}]^T \]  

(11)

\[ Y = [y_1, y_2, \ldots, y_{ps}]^T \]  

(12)
Algorithm 1. The pseudo code of QTFOA.

Initialize the population size \(ps\), the maximum number of iterations \(nfe_{\text{max}}\), the objective function \(f(5)\) and the current iteration number \(G = 1\). Randomly initialize the position coordinates of population \((X_{\text{axis}}, Y_{\text{axis}})\) and the position matrix \(X_G, Y_G\) using equation (1).

Calculate the taste concentration value using equations (2)–(4).

\[ X_{\text{gbest},G} = X_G, Y_{\text{gbest},G} = Y_G \]

While \(G < nfe_{\text{max}}\) do

Generate matrix \(M\) and calculate the corresponding matrix \(\bar{M}\) using equations (10) and (8).

Generate mutation matrices \(B_x\) and \(B_y\) using equations (14) and (15).

Update two location matrices of search agents:

\[ X_G + 1 = M \otimes X_G + \bar{M} \otimes B_x \]
\[ Y_G + 1 = M \otimes Y_G + \bar{M} \otimes B_y \]

Calculate the taste concentration value \(\text{Smell}_{G + 1}\) using equations (2)–(4).

Update individual optimal and global optimal through the principles of equations (18) and (19).

\[ X_{G + 1} = X_{\text{gbest},G + 1}, Y_{G + 1} = Y_{\text{gbest},G + 1} \]
\[ X_{\text{gbest},G + 1} = \text{opt}(X_{\text{gbest},G + 1}), Y_{\text{gbest},G + 1} = \text{opt}(Y_{\text{gbest},G + 1}) \]
\[ G = G + 1 \]

end While

\[ s_{\text{best}} = \left( s_{\text{gbest},G}^2 + s_{\text{opt}}^2 \right)^{1/2} \]

Output the global optimal solution \(X_{\text{gbest},G}, Y_{\text{gbest},G}\) and the global optimal value \(f(s_{\text{best}})\).

3. \(X\) and \(Y\) are updated according to the principle of equation (6), and the update process is shown in equations (16) and (17)

\[ X = M \otimes X + \bar{M} \otimes B_x \quad (16) \]
\[ Y = M \otimes Y + \bar{M} \otimes B_y \quad (17) \]

Update the optimal value

1. We combine the updated \(X\) and \(Y\) with equations (3) and (4) to calculate the taste concentration value of the location of each search agent.

2. For each search agent, we compare the current taste concentration value with its previous taste concentration value. And then keep the minimum value \(\text{bestmore}_i\). The operation process is shown in equation (18)

\[ \text{bestmore}_{i,G} = \min(\text{Smell}_{i,G}, \text{Smell}_{i,G-1}), \quad i = 1, 2, \ldots, ps \quad (18) \]

3. As shown in equation (19), the minimum value among \(\text{bestmore}_G\) is the contemporary global optimal value \(g_{\text{bestval},G}\)

\[ g_{\text{bestval},G} = \min \left( \text{bestmore}_{1,G}, \text{bestmore}_{2,G}, \ldots, \text{bestmore}_{ps,G} \right) \quad (19) \]

The above is the main process of QTFOA. On one hand, QTFOA retains the inherent exploration and exploitation strategy of FOA. On the other hand, it adds the affine transformation in QUATRE, which helps the particles to change the position coordinates in a large range and effectively reduces the probability of falling into local extremes. The details of this process are described by the pseudo code of Algorithm 1.

Simulation experiment and result analysis

Benchmark functions

We select 22 benchmark functions to conduct experiment to test the optimization effect of QTFOA. Tables 2–5 show the specific details. Among them, \(f_1-f_8\) are the universal unimodal functions, \(f_9-f_{10}\) are the low-dimensional unimodal functions, \(f_{11}-f_{17}\) are the universal multimodal functions, and \(f_{18}-f_{22}\) are the low-dimensional multimodal functions.

Unimodal function has only one peak. So, it can explore whether the algorithm can find the global optimal value under limited optimization conditions (such as fewer particles). Multimodal function has multiple peaks, which can explore the ability of one algorithm to escape the trap of falling into local optima. This article adds low-dimensional test functions. In low-dimensional situations, algorithm is more likely to converge prematurely, so this can better detect the performance of QTFOA.
### Table 2. Universal unimodal functions.

| Name                   | Function                                                                 | Search Space      | D  | $f_{\text{min}}$ |
|------------------------|--------------------------------------------------------------------------|-------------------|----|-----------------|
| Sphere                 | $f_1(x) = \sum_{i=1}^{n} x_i^2$                                         | $[-100,100]$      | 30 | 0               |
| Schwefel’s function 2.1| $f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$                        | $[-10,10]$        | 30 | 0               |
| Schwefel’s function 1.2| $f_3(x) = \sum_{i=1}^{n} (\sum_{i=1}^{n} x_i)^2$                         | $[-100,100]$      | 30 | 0               |
| Schwefel’s function 2.22| $f_4(x) = \max_{1 \leq i \leq n} |x_i|$                               | $[-100,100]$      | 30 | 0               |
| Ellipsoid              | $f_5(x) = \sum_{i=0}^{d} x_i^4$                                         | $[-1,1]$          | 30 | 0               |
| Sum of different powers| $f_6(x) = \sum_{i=1}^{d} |x_i|^{i+1}$                               | $[-1.28,1.28]$    | 30 | 0               |
| Dejong’s noisy         | $f_7(x) = \sum_{i=1}^{d} x_i^4 + \text{rand}(0, 1)$                      | $[-1.28,1.28]$    | 30 | 0               |
| Zakharov               | $f_8(x) = \sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5(x_i)^2) + (\sum_{i=1}^{d} 0.5(x_i)^4)$ | $[-5.10]$         | 30 | 0               |

### Table 3. Low-dimensional unimodal functions.

| Name  | Function                                                                 | Search Space      | D  | $f_{\text{min}}$ |
|-------|--------------------------------------------------------------------------|-------------------|----|-----------------|
| Easom | $f_9(x) = -\cos(x_1)\cos(x_2)\exp(-((x_1-\pi)^2+(x_2-\pi)^2))$       | $[-100,100]$      | 2  | -1              |
| Matyas| $f_{10}(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$                          | $[-10,10]$        | 2  | 0               |

### Table 4. Universal multimodal functions.

| Name                   | Function                                                                 | Search Space      | D  | $f_{\text{min}}$ |
|------------------------|--------------------------------------------------------------------------|-------------------|----|-----------------|
| Schwefel               | $f_{11}(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|})$                    | $[-500,500]$      | 30 | -12.569         |
| Rastrigin              | $f_{12}(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$            | $[-5.125,12]$     | 30 | 0               |
| Ackley                 | $f_{13}(x) = -20 \exp(\frac{-0.2(0.1)^n}{\prod_{i=1}^{n} x_i^2}) - \prod_{i=1}^{n} \cos(2\pi x_i) + 20 + e$ | $[-32.32]$        | 30 | 0               |
| Griewank               | $f_{14}(x) = (1/4000) \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$ | $[-600,600]$      | 30 | 0               |
| Levy                   | $f_{15}(x) = \sin^2(\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + 10\sin^2(\pi x_i + 1)]$ | $[-10,10]$        | 30 | 0               |
| Powell                 | $f_{16}(x) = \sum_{i=1}^{d} ((x_{i+3} - 10x_{i+1})^2 + 5(x_{i+1} + x_{i+4})^2 + (x_{i+2} - 2x_{i+4})^4 + 10(x_{i+3} - x_{i+4})^4)$ | $[-4.5]$          | 30 | 0               |
| Styblinski-Tang        | $f_{17}(x) = (1/2) \sum_{i=1}^{d} (x_i^4 - 16x_i^2 + 5x_i)$               | $[-5.5]$          | 30 | -39.16599       |

### Table 5. Low-dimensional multimodal functions.

| Name                   | Function                                                                 | Search Space      | D  | $f_{\text{min}}$ |
|------------------------|--------------------------------------------------------------------------|-------------------|----|-----------------|
| Cross-in-tray          | $f_{18}(x) = -0.0001(|\sin(x_1)|+|\sin(x_2)|)$                          | $[-10,10]$        | 2  | -2.06261        |
| Drop-wave              | $f_{19}(x) = -1 + \cos(12(\sqrt{x_1^2 + x_2^2})/0.5(x_1^2 + x_2^2) + 2)$ | $[-5.125,12]$     | 2  | 1               |
| Schaffer function N. 2 | $f_{20}(x) = 0.5 + (\sin^2(x_1^2 - x_2) - 0.5)/[1 + 0.01(\sin(x_1^2 + x_2^2)^2)]$ | $[-100,100]$      | 2  | 0               |
| Hartman1               | $f_{21}(x) = -\sum_{i=1}^{6} \alpha_i \exp(-\sum_{i=1}^{6} \alpha_i (x_i - p_i)^2)$ | $[1.3]$           | 3  | -3.86           |
| Hartman2               | $f_{22}(x) = -\sum_{i=1}^{d} \alpha_i \exp(-\sum_{i=1}^{d} \alpha_i (x_i - p_i)^2)$ | $[0.1]$           | 6  | -3.32           |
**Compare with mature algorithms**

In this part, we compare QTFOA with FOA, QUATRE, DE, PSO, and GWO. Specifically, we use 30 particles to run these six algorithms on 22 benchmark functions 30 times, and each single function performs 3000 iterations. Finally, we use error average (AVG) to compare optimization results and we use error standard deviation (STD) to evaluate the stability of solution. The win, lose, and draw are used to represent the number of functions that QTFOA is better than, worse than, and equal to corresponding algorithm’s optimized performance, respectively. The running results are shown in Table 6, and in order to highlight the optimal values, we bold them.

From Table 6, for eight universal unimodal functions $f_1$–$f_8$, the performance advantage of QTFOA is absolute. QTFOA randomly generates the solution in the early stage, and then performs the affine transformation of the particle position coordinates, which greatly improves the position diversity and helps the particles to quickly find the optimal value. But for the two low-dimensional unimodal functions $f_9$ and $f_{10}$, neither of them can beat all other algorithms. This shows the performance of QTFOA is not very good on low-dimensional unimodal functions.

For the seven universal multimodal functions $f_{11}$–$f_{17}$, QTFOA ranks first in six functions of $f_{11}$, $f_{12}$, $f_{13}$, $f_{15}$, $f_{16}$, and $f_{17}$. And the data appear that the result of QTFOA is not only superior than other algorithms but also helps most universal multimodal functions find the theoretical global optimum. What multimodal functions urgently need to solve is the inability to escape the local extremum problem. In response to this, QTFOA introduces the affine transformation mechanism, which helps particles seek the optimal solution toward higher expectations and greater optimization potential.

For the five low-dimensional multimodal functions $f_{18}$–$f_{22}$, QTFOA ranks first in three functions of $f_{18}$, $f_{19}$, and $f_{20}$. It can be seen that QTFOA is also excellent in low-dimensional multimodal functions. In short, through the above analysis and the values of win, lose, and draw in Table 6, the overall performance of QTFOA is higher than the other five algorithms.

**Analysis of algorithm convergence**

Convergence is another indicator for judging the quality of an algorithm. Table 7 shows the number of iterations of these six algorithms to get the optimal value on 22 functions. We set the maximum number of iterations to 500. At the same time, to evaluate experimental results faster, we assume that when the optimal value is less than $10^{-4}$, it is roughly considered that the algorithm has converged to 0. If the algorithm does not converge after 500 iterations, 500 is regarded as the number of iterations here. For each function, we bold the data that iterates fastest to the optimal value. That is, the smaller the value, the faster the algorithm iterates on this function.

Figure 1 reveals the convergence curves of the six algorithms on functions $f_1$, $f_9$, $f_{12}$, and $f_{19}$. Four selected examples correspond to the four types of functions proposed, which can help us better analyze the algorithm. Please note that in Figure 1(a), the values of FOA and QTFOA in the whole iteration process are quite different from those of other algorithms. Therefore, Figure 1(b) adds the details of numerical changes of these two algorithms.

From Table 7 and Figure 1, we can see that for most functions, QTFOA can efficiently converge in the fastest time. Whether the speed of convergence or the quality of the solution, QTFOA has demonstrated strong competitiveness. However, the performance of QTFOA on low-dimensional unimodal functions is not good. For example, it can be seen from the iterative curve of $f_9$ that QTFOA tends to fall into local optimal values.

**Significant difference analysis**

To detect the difference of experimental results samples, we compare QTFOA with other five algorithms for significant differences test by the Friedman test and the famous Wilcoxon signed-rank test. Tables 8 and 9 respectively show the experimental results. Among them, when the $p$-value is greater than 0.05, it means that this sample result is not statistically significant.

We bold the data which are greater than 0.05. From the experimental results, we can see only a small amount of data has been annotated. This shows the optimization result of QTFOA is indeed very different from other algorithms significantly.

**Comparison of the running time of six algorithms**

Time complexity is also one of the indicators for judging the capability of an algorithm, which can roughly characterize its computational efficiency. For each function, we run it 500 times with six algorithms, respectively, and then take the average of the corresponding time as the algorithm’s optimization time for this function. The unit of time is seconds. The experiment result is shown in Table 10, and the shortest time is bolded.

From Table 10, it can be seen that QUATRE and PSO have higher operating efficiency, but the QTFOA has a long running time. In QTFOA, the random optimization process and matrix operations increase the
Table 6. Performance comparison of FOA, QUATRE, QTFOA, DE, PSO, and GWO algorithms on 22 functions.

| Function | FOA | QUATRE | QTFOA |
|----------|-----|--------|-------|
|          | AVG | STD    | AVG   | AVG    | STD    | AVG    | STD    | AVG    | STD    | AVG    | STD    |
| f₁       | 7.015e -07 | 5.566e -09 | 9.428e -71 | 2.314e -70 | 0.000e + 00 | 0.000e + 00 |
| f₂       | 4.635e -03 | 2.227e -05 | 6.451e -03 | 3.533e -02 | 0.000e + 00 | 0.000e + 00 |
| f₃       | 2.034e -04 | 2.739e -06 | 1.363e -06 | 2.550e -06 | 0.000e + 00 | 0.000e + 00 |
| f₄       | 1.025e -03 | 8.935e -05 | 2.499e + 00 | 1.691e + 00 | 0.000e + 00 | 0.000e + 00 |
| f₅       | 9.836e -06 | 9.404e -08 | 5.901e -70 | 1.281e -69 | 0.000e + 00 | 0.000e + 00 |
| f₆       | 9.593e -10 | 7.225e -12 | 1.062e -129 | 3.636e -129 | 0.000e + 00 | 0.000e + 00 |
| f₇       | 6.528e -04 | 2.298e -04 | 8.182e -03 | 4.778e -03 | 1.397e -05 | 1.473e -05 |
| f₈       | 1.132e -03 | 1.003e -05 | 1.263e -151 | 4.087e -151 | 0.000e + 00 | 0.000e + 00 |
| f₉       | 9.715e -01 | 6.952e -02 | 4.067e -05 | 2.201e -04 | 1.000e + 00 | 2.915e -08 |
| f₁₀      | 6.841e -11 | 1.094e -12 | 2.003e -59 | 1.089e -58 | 2.323e -231 | 0.000e + 00 |
| f₁₁      | 2.198e +00 | 1.609e +01 | -1.048e +04 | 3.473e +02 | 1.251e +01 | 3.557e -02 |
| f₁₂      | 1.425e -04 | 1.440e -06 | 6.734e +01 | 1.853e +01 | 0.000e + 00 | 0.000e + 00 |
| f₁₃      | 6.192e -04 | 2.851e -06 | 8.937e -01 | 1.397e +00 | 8.882e -16 | 0.000e + 00 |
| f₁₄      | 3.215e -08 | 3.662e -10 | 9.351e -03 | 1.102e -02 | 0.000e + 00 | 0.000e + 00 |
| f₁₅      | 2.481e +00 | 4.172e -02 | 3.725e +00 | 3.472e +00 | 2.726e +00 | 1.042e-01 |
| f₁₆      | 2.174e -05 | 4.392e -06 | 2.426e -06 | 1.718e -06 | 0.000e + 00 | 0.000e + 00 |
| f₁₇      | -2.130e +01 | 8.992e +00 | -1.018e +03 | 3.839e +01 | -1.078e +01 | 0.000e + 00 |
| f₁₈      | -2.053e +00 | 5.654e +00 | -2.063e +00 | 9.034e +16 | -1.089e +00 | 9.031e -02 |
| f₁₉      | 6.375e -02 | 2.012e -07 | 3.429e -02 | 3.205e -02 | 0.000e + 00 | 0.000e + 00 |
| f₂₀      | 3.372e -12 | 3.376e -14 | 2.515e -03 | 7.052e -03 | 0.000e + 00 | 0.000e + 00 |
| f₂₁      | 9.327e -02 | 7.244e -02 | -2.782e -03 | 8.108e -17 | 3.574e -01 | 2.719e -01 |
| f₂₂      | 6.843e -01 | 1.872e -01 | 2.178e -02 | 4.837e -02 | 9.860e -01 | 4.386e -01 |

| Win       | 17  | 19  | 11  |
| Lose      | 5   | 3   |   |
| Draw      | 0   | 0   |   |

Function | DE | PSO | GWO |
|----------|----|-----|-----|
| AVG      |    |    |     |
| STD      |    |    |     |
| AVG      |    |    |     |
| STD      |    |    |     |
| AVG      |    |    |     |
| STD      |    |    |     |
| AVG      |    |    |     |
| STD      |    |    |     |
| AVG      |    |    |     |
| STD      |    |    |     |

FOA: Fruit Fly Optimization Algorithm; QUATRE: QJassi-Affine TRansformation Evolution; QTFOA: Quasi-affine Transformation evolutionary for the fruit fly Optimization Algorithm; DE: Differential Evolution; PSO: Particle Swarm Optimization; GWO: Gray Wolf Optimization.
Each shipping point has a different demand and the demand for each shipping point is fulfilled if the demand of delivery place $i$ is fulfilled by vehicle $k$, otherwise 0.

$$y_{ki} = \begin{cases} 
1 & \text{if the demand of delivery place } i \text{ is fulfilled by vehicle } k \\
0 & \text{else}
\end{cases} \tag{20}$$

$C_{ij}$ is the distance between shipping point $i$ and $j$. $Z$ is the sum of all vehicle paths and $Q$ represents the maximum capacity of transport vehicles. CVRP is to find the minimum of $Z$ and the related formula is shown below.

$$\min Z = \sum_{i=0}^{L} \sum_{j=0}^{L} \sum_{k=1}^{K} C_{ij} x_{ijk} \tag{22}$$

$$\sum_{j=0}^{L} x_{ijk} = y_{jk}, \quad j = 1, 2, ..., L; k = 1, 2, ..., K \tag{23}$$

$$\sum_{i=0}^{L} x_{ijk} = y_{ik}, \quad i = 1, 2, ..., L; k = 1, 2, ..., K \tag{24}$$

$$\sum_{k=1}^{K} y_{ik} = 1, \quad i = 1, 2, ..., L \tag{25}$$

$$\sum_{i=0}^{L} g_{ij} y_{ik} \leq Q, \quad k = 1, 2, ..., K \tag{26}$$

Equations (23) and (24) can ensure that the requirements of each shipping point are met. Equation (25) can ensure that the needs of each shipping point can only be met by one car. For each vehicle, equation (26) can ensure the total cargo volume of whole transportation path shall not exceed its own maximum capacity.

### Application of QTFOA in CVRP

#### Principle of CVRP

The problem can be described as: there is a central warehouse (numbered 0) transporting goods to $L$ shipping points (numbered 1, 2, ..., $L$) through $K$ vehicles. Each shipping point has a different demand $g_i$ ($i = 1, 2, ..., L$) for goods and the demand for each shipping point can only be met by one car. The model requires that every shipping point must receive the delivery service of a vehicle. The capacity of each vehicle is $q_k$ ($k = 1, 2, ..., K$). For each vehicle, the total cargo volume of its transportation path shall not exceed its own maximum capacity. Now, it is required to find the shortest vehicle travel path.

First, two variables $y_{ki}$ and $x_{ijk}$ are defined by equations (20) and (21).

$$x_{ijk} = \begin{cases} 
1 & \text{if vehicle } k \text{ travels from position } i \text{ to } j \\
0 & \text{else}
\end{cases} \tag{21}$$

$C_{ij}$ is the distance between shipping point $i$ and $j$. $Z$ is the sum of all vehicle paths and $Q$ represents the maximum capacity of transport vehicles. CVRP is to find the minimum of $Z$ and the related formula is shown below.

$$\min Z = \sum_{i=0}^{L} \sum_{j=0}^{L} \sum_{k=1}^{K} C_{ij} x_{ijk} \tag{22}$$

$$\sum_{j=0}^{L} x_{ijk} = y_{jk}, \quad j = 1, 2, ..., L; k = 1, 2, ..., K \tag{23}$$

$$\sum_{i=0}^{L} x_{ijk} = y_{ik}, \quad i = 1, 2, ..., L; k = 1, 2, ..., K \tag{24}$$

$$\sum_{k=1}^{K} y_{ik} = 1, \quad i = 1, 2, ..., L \tag{25}$$

$$\sum_{i=0}^{L} g_{ij} y_{ik} \leq Q, \quad k = 1, 2, ..., K \tag{26}$$

Equations (23) and (24) can ensure that the requirements of each shipping point are met. Equation (25) can ensure that the needs of each shipping point can only be met by one car. For each vehicle, equation (26) can ensure the total cargo volume of whole transportation path shall not exceed its own maximum capacity.

#### Simulation experiment and result analysis

First, to study whether QTFOA is suitable for solving CVRP, we use a simple example in Table 11 for testing. The serial number 0 represents the central warehouse, and the serial numbers 1–7 represent seven shipping points. Set the number of transport vehicles to 3, and stipulate the capacity of each vehicle cannot exceed 100. Location information and cargo requirements are given in Table 11. The theoretical optimal path value is known to be 217.8. The optimization result of QTFOA is shown in Figure 2.

In the above test case, QTFOA can optimize the path to 221.4 quickly, which is very close to 217.8. This shows that the QTFOA is suitable for CVRP and can achieve good optimization results.

Later, we compare QTFOA with QUATRE, FOA, and SA. Four algorithms are tested on 10 sets of data from the CVRP international standard example VRPLIB. The numerical results of this comparison can be observed in Table 12. Here, in order to ensure fairness, for each algorithm, we run 3000 generations with 150 particles.
Figure 1. The convergence curve of the six algorithms on $f_1$, $f_9$, $f_{12}$, and $f_{19}$. (a) Convergence curve of $f_{1}$-1. (b) Convergence curve of $f_{1}$-2. (c) Convergence curve of $f_{9}$. (d) Convergence curve of $f_{12}$. (e) Convergence curve of $f_{19}$. 
Table 8. Friedman’s test of six comparison algorithms.

| Function | Sum of squares | Degree of freedom | Mean squares | p-value |
|----------|----------------|------------------|--------------|---------|
| $f_1$    | 2100           | 5                | 420          | 3.52222e-33 |
| $f_2$    | 2092.7         | 5                | 418.453      | 4.72139e-33 |
| $f_3$    | 2100           | 5                | 420          | 3.52222e-33 |
| $f_4$    | 2066.67        | 5                | 413.333      | 1.24527e-32 |
| $f_5$    | 2100           | 5                | 420          | 3.52222e-33 |
| $f_6$    | 2032.27        | 5                | 406.453      | 3.72309e-33 |
| $f_7$    | 1932.27        | 5                | 386.453      | 2.60534e-30 |
| $f_8$    | 1860.8         | 5                | 327.16       | 4.07451e-33 |
| $f_9$    | 1922.47        | 5                | 384.493      | 3.57677e-31 |
| $f_{10}$ | 1840.73        | 5                | 368.147      | 1.51437e-29 |
| $f_{11}$ | 2006.67        | 5                | 401.333      | 1.57545e-31 |
| $f_{12}$ | 1982.4         | 5                | 396.48       | 2.42936e-32 |
| $f_{13}$ | 1762.87        | 5                | 352.573      | 1.27161e-29 |
| $f_{14}$ | 1401.42        | 5                | 280.283      | 1.80424e-26 |
| $f_{15}$ | 1376.4         | 5                | 275.28       | 2.60282e-21 |
| $f_{16}$ | 2075.27        | 5                | 415.053      | 8.99041e-33 |
| $f_{17}$ | 1959.47        | 5                | 391.893      | 8.87273e-31 |
| $f_{18}$ | 1993.07        | 5                | 396.613      | 1.72536e-32 |
| $f_{19}$ | 1447.47        | 5                | 289.493      | 1.30706e-26 |
| $f_{20}$ | 1223.6         | 5                | 244.72       | 2.20846e-25 |
| $f_{21}$ | 1934.2         | 5                | 386.84       | 1.72389e-31 |
| $f_{22}$ | 1616.62        | 5                | 323.323      | 1.49473e-25 |

Table 9. Wilcoxon’s signed-rank test of six comparison algorithms.

| Function | QTFOA       | FOA         | QUATRE       | DE          | PSO         | GWO         |
|----------|-------------|-------------|--------------|-------------|-------------|-------------|
| $f_1$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_2$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_3$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_4$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_5$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_6$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_7$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 1.2118e-12  | 1.2118e-12  |
| $f_8$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0180e-11  | 3.1602e-12  | 3.1602e-12  |
| $f_9$    | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0180e-11  | 3.0180e-11  | 3.0180e-11  |
| $f_{10}$ | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 8.8910e-10  | 8.8910e-10  |
| $f_{11}$ | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.1602e-12  | 3.1602e-12  |
| $f_{12}$ | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 1.2118e-12  | 1.2118e-12  |
| $f_{13}$ | N/A         | 0.1849      | 1.2118e-12   | 1.2118e-12  | 2.3638e-12  | 2.3638e-12  |
| $f_{14}$ | N/A         | 0.3771      | 1.2118e-12   | 1.2377e-09  | 3.0199e-11  | 2.4785e-08  |
| $f_{15}$ | N/A         | 0.0773      | 8.1527e-11   | 1.2118e-12  | 0.6627      | 3.0199e-11  |
| $f_{16}$ | N/A         | 3.0199e-11  | 1.2118e-12   | 3.0199e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_{17}$ | N/A         | 2.9710e-11  | 5.2650e-05   | 2.8163e-11  | 3.0199e-11  | 3.0199e-11  |
| $f_{18}$ | N/A         | 1.7203e-12  | 3.0199e-11   | 1.2118e-12  | 3.0199e-11  | 3.0199e-11  |
| $f_{19}$ | N/A         | 7.1519e-07  | 1.2118e-12   | 2.3638e-12  | 3.0199e-11  | 1.2118e-12  |
| $f_{20}$ | N/A         | 0.3736      | 1.2118e-12   | 1.2118e-12  | 0.0020      | 1.2118e-12  |
| $f_{21}$ | N/A         | 1.2118e-12  | 1.2474e-04   | 1.2118e-12  | 4.9752e-11  | 4.0772e-11  |
| $f_{22}$ | N/A         | 1.5466e-11  | 0.1907       | 1.4634e-11  | 6.6953e-11  | 3.0199e-11  |

Table 12 shows that QTFOA can obtain the shortest path on each set of test cases. At the same time, in general, as the number of shipping points and transportation vehicles increases, the advantages of QTFOA become more obvious. This shows that when faced with a complex transportation environment,
QTFOA can play a better role. In short, QTFOA has performed very well in solving CVRP.

To further explore the optimization process of QTFOA to CVRP, we have done an analysis of the convergence process of four algorithms. Figure 3

Table 10. Running time of six comparison algorithms.

| Function | FOA | QUATRE | QTFOA | DE  | PSO | GWO |
|----------|-----|--------|-------|-----|-----|-----|
| f1       | 0.5068 | 1.0067e-70 | 0.9147 | 1.5775 | 0.1645 | 0.9368 |
| f2       | 0.5116 | 1.2913e-40 | 0.9197 | 1.5949 | 0.1811 | 0.9341 |
| f3       | 1.5011 | 6.0247e-07 | 1.9067 | 2.4950 | 1.1587 | 1.8967 |
| f4       | 0.4991 | 3.0425 | 0.9081 | 1.4937 | 0.1671 | 0.9347 |
| f5       | 0.9944 | 9.8505e-123 | 1.1690 | 2.0514 | 0.5945 | 1.2882 |
| f6       | 1.0648 | 0.0078 | 1.2683 | 2.0214 | 0.7333 | 1.4733 |
| f7       | 0.5067 | 5.5970e-152 | 0.9109 | 1.4389 | 0.1715 | 0.9219 |
| f8       | 0.3583 | 0.9524 | 0.7141 | 1.2973 | 0.1287 | 0.2576 |

FOA: Fruit Fly Optimization Algorithm; QUATRE: QUasi-Affine TRansformation Evolution; QTFOA: Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm; DE: Differential Evolution; PSO: Particle Swarm Optimization; GWO: Gray Wolf Optimization.

Table 11. Coordinates and requirements of each task point.

| Sequence | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----------|---|---|---|---|---|---|---|---|
| Coordinate | (18,54) | (22,60) | (58,69) | (71,71) | (83,46) | (91,38) | (24,42) | (18,40) |
| Need | 0 | 89 | 14 | 28 | 33 | 21 | 41 | 57 |

Table 12. Comparison results of four algorithms to solve CVRP.

| Function | QTFOA | QUATRE | FOA | SA |
|----------|-------|--------|-----|----|
| A-L32-K5 | 1282 | 1681 | 1686 | 1625 |
| A-L33-K5 | 1090 | 1551 | 1476 | 1398 |
| A-L33-K6 | 1180 | 1661 | 1459 | 1481 |
| A-L34-K5 | 1379 | 1795 | 1664 | 1606 |
| A-L36-K5 | 1356 | 1879 | 1664 | 1712 |
| A-L44-K7 | 1748 | 2294 | 2235 | 2114 |
| A-L45-K7 | 1749 | 2576 | 2048 | 2317 |
| A-L46-K7 | 1934 | 2597 | 2296 | 2276 |
| A-L53-K7 | 2194 | 2963 | 2668 | 2776 |
| A-L62-K8 | 2505 | 3850 | 3287 | 3603 |

FOA: Fruit Fly Optimization Algorithm; QUATRE: QUasi-Affine TRansformation Evolution; QTFOA: Quasi-affine Transformation evolutionary for the Fruit fly Optimization Algorithm; SA: Simulated Annealing algorithm.

Figure 2. Optimization results of QTFOA on CVRP.
reveals the convergence curves of A-L32-k5 and A-L33-k5. As can be seen from Figure 3, QTFOA can get the highest quality solution, that is, the shortest path. However, from the image, we can see that QTFOA converges very slowly. When solving the situation with large quantities of data, the calculation speed and efficiency must be specially considered. So, this can be an aspect of future research.

Conclusion

Combining FOA and QUATRE, a new optimization algorithm QTFOA is proposed in this article. To effectively avoid sinking into the local optimum, QTFOA uses affine transformation formula in QUATRE algorithm, which helps the particles to expand the search range greatly. To explore the optimization ability of QTFOA, we select 22 functions of four types for testing, and compare the algorithm with FOA, QUATRE, PSO, DE, and GWO. It is obvious from the above experiment result that the optimization performance of QTFOA is higher than the other five algorithms as a whole. To evaluate the capabilities of QTFOA in multiple aspects more comprehensively, we also design experiments for algorithm convergence analysis, the Friedman test, the Wilcoxon signed-rank test, and running time comparison. Most experiments have proved that QTFOA is excellent in different evaluation indexes. In the end, we apply QTFOA to CVRP. Through 10 sets of cases on VRPLIB, QTFOA is compared with QUATRE, FOA, and SA and the experimental result proves that QTFOA can always find the shortest effective path. But QTFOA still has three obvious shortcomings. First, QTFOA has poor performance on low-dimensional unimodal functions. Second, it has a particularly slow running time. Finally, when solving CVRP, the convergence speed of it is very slow. How to solve these three problems of QTFOA is an improvement direction that can be considered in the future.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs

Pei Hu https://orcid.org/0000-0003-3711-4952
Jeng-Shyang Pan https://orcid.org/0000-0002-3128-9025

References

1. Kou X and Feng J. Matching ontologies through compact monarch butterfly algorithm. J Netw Intel 2020; 5: 191–197.
2. Yang J, Gui ZG, Zhang LY, et al. Aperture generation for intensity-modulated radiotherapy scheme based on gradient information. J Netw Intel 2019; 4: 58–70.
3. Geetha R and Geetha S. A multi-layered “plus-minus one” reversible data embedding scheme. J Inf Hiding Multim Signal Process 2020; 11: 58–69.
4. Sun Y, Yin S and Li H. A new wavelet threshold function based on Gaussian kernel function for image de-noising. J Inf Hiding Multim Signal Process 2019; 10: 91–101.
5. Song W and Huang C. Mining high average-utility itemsets based on particle swarm optimization. *Data Sci Pattern Recogn* 2020; 4: 19–32.

6. Yuan J, Zhao Z, Liu Y, et al. DMPPT control of photovoltaic microgrid based on improved sparrow search algorithm. *IEEE Access* 2021; 9: 16623–16629.

7. Zhang C and Ding S. A stochastic configuration network based on chaotic sparrow search algorithm. *Knowl Based Syst* 2021; 220: 106924.

8. Xue J and Shen B. A novel swarm intelligence optimization approach: sparrow search algorithm. *Syst Sci Control Eng* 2020; 8: 22–34.

9. Liang H, Zou J, Zuo K, et al. An improved genetic algorithm optimization fuzzy controller applied to the wellhead back pressure control system. *Mech Syst Signal Process* 2020; 142: 106708.

10. Guijarro F, Martínez-Go´mez M and Visbal-Cadavid D. A model for sector restructuring through genetic algorithm and inverse DEA. *Exp Syst Appl* 2020; 154: 113422.

11. Lin H, Yang C and Xu X. A new optimization model of CCHP system based on genetic algorithm. *Sustain Cit Soc* 2020; 52: 101811.

12. Arora S and Singh S. Butterfly optimization algorithm: a novel approach for global optimization. *Soft Comput* 2019; 23: 715–734.

13. Yuan Z, Wang W and Wang H. Optimal parameter estimation for PEMFC using modified monarch butterfly optimization. *Int J Energ Res* 2020; 44: 8427–8441.

14. Maheshwari P, Sharma AK and Verma K. Energy efficient cluster based routing protocol for WSN using butterfly optimization algorithm and ant colony optimization. *Ad Hoc Netw* 2021; 110: 102317.

15. Wang BC, Feng Y and Li HX. Individual-dependent feasibility rule for constrained differential evolution. *Inform Sci* 2020; 506: 174–195.

16. Krishna GJ and Ravi V. High utility itemset mining using binary differential evolution: an application to customer segmentation. *Exp Syst Appl* 2021; 181: 115122.

17. Li S, Gong W, Hu C, et al. Adaptive constraint differential evolution for optimal power flow. *Energy* 2021; 235: 121362.

18. Sun G, Li C and Deng L. An adaptive regeneration framework based on search space adjustment for differential evolution. *Neur Comput Appl* 2021; 33: 1–17.

19. Cai X, Niu Y, Geng S, et al. An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search. *Concurr Comput Pract Exp* 2020; 32: e5478.

20. Inci M and Caliskan A. Performance enhancement of energy extraction capability for fuel cell implementations with improved cuckoo search algorithm. *Int J Hydro Energy* 2020; 45: 11309–11320.

21. Gude S and Jana KC. Parameter extraction of photovoltaic cell using an improved cuckoo search optimization. *Sol Energy* 2020; 204: 280–293.

22. Too J, Abdullah AR and Mohd Saad N. A new quadratic binary Harris hawk optimization for feature selection. *Electronics* 2019; 8: 1130.

23. Li C, Li J, Chen H, et al. Enhanced Harris hawks optimization with multi-strategy for global optimization tasks. *Exp Syst Appl* 2021; 185: 115499.

24. Jiao S, Chong G, Huang C, et al. Orthogonally adapted Harris hawks optimization for parameter estimation of photovoltaic models. *Energy* 2020; 203: 117804.

25. Tripathi S, Shrivastava A and Jana KC. Self-tuning fuzzy controller for sun-tracker system using gray wolf optimization (GWO) technique. *ISA Trans* 2020; 101: 50–59.

26. Hua D, Liu X, Sun S, et al. Precise locomotion controller design for a novel magnetorheological fluid robot based on improved gray wolf optimization algorithm. *Smart Mater Struct* 2021; 30: 025038.

27. Moayedi H, Nguyen H and Kok Foong L. Nonlinear evolutionary swarm intelligence of grasshopper optimization algorithm and gray wolf optimization for weight adjustment of neural network. *Eng Comput* 2021; 37: 1265–1275.

28. Xie Q, Guo Z, Liu D, et al. Optimization of heliostat field distribution based on improved gray wolf optimization algorithm. *Renew Energ* 2021; 176: 447–458.

29. Pan JS, Tsai PW and Liao YB. Fish migration optimization based on the fishy biology. In: *Proceedings of the 4th international conference on genetic and evolutionary computing (ICGEC 2010)*, Shenzhen, China, 13–15 December 2010, pp.783–786. New York: IEEE.

30. Pan JS, Hu P and Chu SC. Binary fish migration optimization for solving unit commitment. *Energy* 2021; 226: 120329.

31. Song XF, Zhang Y, Gong DW, et al. Feature selection using bare-bones particle swarm optimization with mutual information. *Pattern Recog* 2021; 112: 107804.

32. Wu TY, Lin JCW, Zhang Y, et al. A grid-based swarm intelligence algorithm for privacy-preserving data mining. *Appl Sci* 2019; 9: 774.

33. Shafipour M, Rashno A and Fadaei S. Particle distance rank feature selection by particle swarm optimization. *Exp Syst Appl* 2021; 185: 115620.

34. Song B, Wang Z and Zou L. An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. *Appl Soft Comput* 2021; 100: 106960.

35. Song PC, Chu SC, Pan JS, et al. Simplified Phasmatodea population evolution algorithm for optimization. *Comp Intell Syst* 2021; 7: 1–19.

36. Song PC, Chu SC, Pan JS, et al. Phasmatodea population evolution algorithm and its application in length-changeable incremental extreme learning machine. In: *Proceedings of the 2020 2nd international conference on industrial artificial intelligence (IAI 2020)*, Shenyang, China, 23–25 October 2020, pp.1–5. New York: IEEE.

37. Chu SC, Tsai PW and Pan JS. Cat swarm optimization. In: *Proceedings of the Pacific Rim international conference on artificial intelligence (PRICAI 2006)*, Guilin, China, 7–11 August 2006, pp.854–858. Berlin: Springer.

38. Li J, Gao M, Pan JS, et al. A parallel compact cat swarm optimization with multi-strategy for global optimization tasks. *Exp Syst Appl* 2021; 185: 115499.
the Pacific Rim international conference on artificial intelligence (PRICAI 2004), Auckland, New Zealand, 9–13 August 2004, pp.534–543. Berlin: Springer.

41. Imtiaz ZB, Manzoor A, ul Islam S, et al. Discovering communities from disjoint complex networks using multi-layer ant colony optimization. *Fut Gener Comput Syst* 2021; 115: 659–670.

42. Guan B, Zhao Y and Li Y. An improved ant colony optimization with an automatic updating mechanism for constraint satisfaction problems. *Exp Syst Appl* 2021; 164: 114021.

43. Tao Y, Gao H, Ren F, et al. A mobile service robot global path planning method based on ant colony optimization and fuzzy control. *Appl Sci* 2021; 11: 3605.

44. Pan WT. A new fruit fly optimization algorithm: taking the financial distress model as an example. *Knowl Based Syst* 2012; 26: 69–74.

45. Fan Y, Wang P, Heidari AA, et al. Rationalized fruit fly optimization with sine cosine algorithm: a comprehensive analysis. *Exp Syst Appl* 2020; 157: 113486.

46. Yang Y, Chen H, Li S, et al. Orthogonal learning harmonizing mutation-based fruit fly-inspired optimizers. *Appl Math Model* 2020; 86: 368–383.

47. Zheng XL and Wang L. An improved fruit fly optimization algorithm and its application to joint replenishment problems. *Exp Syst Appl* 2015; 42: 4310–4323.

48. Zheng XL and Wang L. A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. *Exp Syst Appl* 2016; 65: 28–39.

49. Kanarachos S, Dizqah AM, Chrysakis G, et al. Optimal design of a quadratic parameter varying vehicle suspension system using contrast-based fruit fly optimisation. *Appl Soft Comput* 2018; 62: 463–477.

50. Pan JS, Meng Z, Xu H, et al. QUasi-Affine TRansformation Evolution (QUATRE) algorithm: a new simple and accurate structure for global optimization. In: *Proceedings of the international conference on industrial, engineering and other applications of applied intelligent systems (IEA/AIE 2016)*, Morioka, Japan, 2–4 August 2016, pp.657–667. Berlin: Springer.

51. Meng Z, Pan JS and Xu H. QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: a cooperative swarm based algorithm for global optimization. *Knowl Based Syst* 2016; 109: 104–121.

52. Du ZG, Pan JS, Chu SC, et al. Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks. *IEEE Access* 2020; 8: 8583–8594.

53. Meng Z and Pan JS. QUasi-Affine TRansformation Evolution with External ARchive (QUATRE-EAR): an enhanced structure for differential evolution. *Knowl Based Syst* 2018; 155: 35–53.

54. Altabeeb AM, Mohsen AM, Abualigah L, et al. Solving capacitated vehicle routing problem using cooperative firefly algorithm. *Appl Soft Comput* 2021; 108: 107403.

55. Accorsi L and Vigo D. A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. *Transport Sci* 2021; 55: 832–856.

56. Fallah H, Didehvar F and Rahmati F. Approximation algorithms for the load-balanced capacitated vehicle routing problem. *Bull Math Soc Sci Math Roumanie* 2021; 47: 1261–1288.

57. Molina JC, Salmeron JL, Eguia I, et al. The heterogeneous vehicle routing problem with time windows and a limited number of resources. *Eng Appl Artif Intell* 2020; 94: 103745.

58. Mojtahedi M, Fathollahi-Fard AM, Tavakkoli-Moghaddam R, et al. Sustainable vehicle routing problem for coordinated solid waste management. *J Ind Inform Integr* 2021; 23: 100220.

59. Wang Y, Li Q, Guan X, et al. Two-echelon collaborative multi-depot multi-period vehicle routing problem. *Exp Syst Appl* 2021; 167: 114201.

60. Niu Y, Zhang Y, Cao Z, et al. MIMOA: a membrane-inspired multi-objective algorithm for green vehicle routing problem with stochastic demands. *Swarm Evol Comput* 2021; 60: 100767.