The feedback artificial tree (FAT) algorithm

Q. Q. Li[1]*, Z. C. He[2], Eric Li[3*]

[1]*College of Automotive and Mechanical Engineering, Changsha University of Science and Technology, 410114, Yuhua District, Changsha City, Hunan Province, P. R. China*

[2]*State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha, 410082 P. R. China*

[3]*School of Science, Engineering & Design, Teesside University, Middlesbrough, UK*

Abstract

Inspired by the transport of organic matters and the update theories of branches, the artificial tree (AT) algorithm was proposed recently. This work presents an improved version of AT algorithm that is called the feedback artificial tree (FAT) algorithm. In FAT, besides the transfer of organic matters, the feedback mechanism of moistures is introduced. Meanwhile, the self-propagating operator and dispersive propagation operator are also put forward. Some typical benchmark problems are applied to test the performance of FAT. The experimental results have clearly demonstrated the higher performance of FAT compared with AT over the tested set of problems. In addition, some well-known heuristic algorithms and their improved algorithms are also applied to validate the performance of FAT, and the computational results of FAT listed in this study are the best among these algorithms. In addition, sensitive analyses on the specific parameters of FAT algorithm are carried out, and the performance of FAT is validated.

***Key words*:** artificial tree algorithm; heuristic algorithms; feedback mechanism; self-propagating operator; dispersive propagation operator.

* Corresponding author.
E-mail address: hdliqiqi@163.com (Q. Q. Li); ericsg2012@gmail.com (Eric Li)

1 Introduction

Heuristic algorithms (Fister Jr et al. 2013; Glibovets & Gulayeva 2013; Ming et al. 2014) are the common optimization algorithms that have many advantages compared with traditional deterministic optimization theories. For example, they do not require the continuity and differentiability of the optimization equation (Hamzaçebi 2008). Therefore, heuristic algorithms have been applied to solve a large range of practical optimization problems efficiently, such as design of metamaterials (Li et al. 2018; Li et al. 2019a), structural optimization (Duan et al. 2019a; Duan et al. 2019b; Li et al. 2019b), load identification (Xu et al. 2019), traffic forecast (Li et al. 2015) and image processing (Malik et al. 2016; Zhong et al. 2016b). Various heuristic algorithms have been proposed and studied, such as the genetic algorithm (GA) (Holland 1992) , the differential evolution (DE) algorithm (Storn & Price 1997), the particle swarm optimization (PSO) algorithm (Kennedy & Eberhart 1997), the ant colony optimization (ACO) (Dorigo & Caro 1999), the artificial fish swarm algorithm (AFSA) (Li & Qian 2003) and the artificial bee colony (ABC) algorithm (Karaboga & Basturk 2007). Heuristic algorithms are mainly inspired by the biological and natural phenomena. For example, GA was proposed based on the Darwinian theory of survival of the fittest (Holland 1992). PSO, ABC, ACO and AFSA are inspired by the foraging behaviors of bird flocks (Kennedy & Eberhart 1997), honey bees (Karaboga & Basturk 2007), ant colonies (Dorigo & Caro 1999) and fish schooling (Li & Qian 2003). Gravity search algorithm (GSA) and biogeography-based optimization (BBO) are inspired by the gravity field (Rashedi et al. 2009) and the migration behavior of island species (Simon 2016).

Besides these standard heuristic algorithms, their improved versions (Zhong et al. 2016a; Lin et al. 2017; Yang et al. 2017; Chen et al. 2018; Huang et al. 2019; Singh & Deep 2019; Zandevakili et al. 2019) are also widely studied, such as the adaptive particle swarm optimization (APSO) algorithm (Zhang et al. 2014), the modified artificial bee colony (MABC) algorithm (Gao & Liu 2012) and the self-adaptive differential evolution (SaDE) (Coelho et al. 2013) algorithm. Compared with the standard algorithms, the improved versions enhance their performances in some aspects. APSO (Zhang et al. 2014) has higher search efficiency than classical PSO. The optimization process of APSO mainly consists of two parts. First, the evolutionary state of particles is evaluated in real time through the evaluation of population distribution and particle fitness. It can adaptively control the parameters of the algorithm to improve the search efficiency. Then, when the evolutionary state reaches the convergence state, the elite learning strategy is implemented. This strategy helps particles to jump out of the local optimum solution. Compared with the standard ABC, the improvement of the MABC (Gao & Liu 2012) mainly includes three parts. The first one is to improve the search equation of ABC based on the DE algorithm. Then, the selection probability P is introduced by the second part to balance the influence of the original search equation and the new proposed search equation on the search of solutions. Finally, the chaotic systems and opposition-based learning theories are applied to produce the initial population to enhance the global convergence. Compared with DE, the enhancement of SaDE (Coelho et al. 2013) is mainly based on its adaptive process of parameters and solutions. Through the learning of previous promising solutions, both the test vector generation strategies and the values of control parameters are gradually self-adapted. Therefore, more appropriate solution generation strategy and parameter setting process can be obtained adaptively according to different stages of the search process.

Inspired by the transport of organic matters and the update of branches, the artificial tree (AT) algorithm was proposed by Li et al. (2017). Some well-known heuristic algorithms were applied to test the performance of AT algorithm, and experimental results proved the high accuracy of AT. Obviously, the performance of AT algorithm has a lot to do with the rationality of its bio-inspired model. For the normal grow of trees, the exchange of materials of trees should both contain the transfer of organic matters from leaves to roots and the delivery of moistures from roots to leaves. Therefore, the exchange process of materials is a feedback cycle. The delivery of moistures is the feedback process of the transport of organic matters. Therefore, the bio-inspired model of the standard AT algorithm is not complete since it only considers the transfer of organic matters. Due to this reason, the feedback mechanism of moistures is introduced into AT to further enhance the performance of AT algorithm.

In this work, the improved version of AT algorithm, named the feedback artificial tree (FAT) algorithm, is developed, and the performances of FAT are investigated through some typical test problems. The results of FAT are first compared with AT, and FAT obtains the better solutions among these test functions with the same parameter values and function evaluation number. Then, the results of FAT on ten high dimensional problems are compared with some well-known heuristic algorithms (namely, PSO, DE and ABC) and their improved versions (namely, APSO, SaDE and MABC). The performance of FAT is proved for it obtains more optimum solutions compared with these six algorithms. Finally, the effects of the parameters which control the initial branch number in the feedback process and the update number of branch population in the organic matter transport process, on the performance of FAT are also studied.

This paper is organized as follows: the basic theory of AT algorithm is presented in Section 2. Section 3 illustrates the principle of FAT algorithm. The feedback mechanism of moistures, the self-propagating operator and the dispersive propagation operator are studied in detail. Section 4 shows the computational results of FAT, AT and other algorithms with some typical benchmark functions, and the sensitive analyses of FAT on parameters r and h are also studied. Finally, Section 5 gives the conclusions of this work.

2 The basic theory of artificial tree algorithm

Figure 1 illustrates the bio-inspired model of a tree which consists of leaves and branches. In Fig. 1, the branches themselves are the solutions. A thicker branch means a better solution, and the thickest tree trunk is the best solution. The tiny branches connected to the leaves represent the initial branch population. The brackets outside the branches represent the branch territories. Each branch has its own territory, which is the growth space of the branch. A thicker branch tends to have a larger territory.

The transport process of organic matters and the update process of branches are also depicted in Fig. 1. The organic matters are first produced in the leaves, and they spreads from top to bottom in all branches. The transfer direction of organic matters is the same as the update direction of branches, and the transfer of organic matters depends on the update of branches. Therefore, the update of branches determines the implementation of the AT algorithm. In addition, as the branches are updated, the branches become thicker (better solutions). Three branch update theories that are the self-evolution operator, the crossover operator and the random operator exist in AT algorithm. The self-evolution and crossover operators are the main branch evolution operators, and the random operator is a supplement operator to prevent the optimization from falling into local optimum. Figure 1 shows the crossover operator and the self-evolution operator. The crossover operator combines two branches into a thicker branch. The self-evolving operator makes the branches themselves thicker. The crossover of branches occurs within the blue circle, and the self-evolution of branches occurs within the red square.

The way to update one branch depends on its territory and the number of other branches in this territory. As in the upper right part of Fig. 1, when the branches in one branch territory are too many, the crossover operator is suppressed and the self-evolution operator is adopted to update this branch. Otherwise, as in the upper left part of Fig. 1, the crossover operator is applied. Because, when there are too many branches around a branch, it will affect the growth of this branch. The self-evolution operator should be applied to make the branch jump out of the dense area. Furthermore, AT algorithm requires the newly generated branch to be better (thicker) than the original one. If the new branch is better than the original branch, the new branch replaces the original branch in the branch population. If the new branch is not better than the original branch, the new branch is discarded and another new branch is produced. If the newly generated branch is still worse than the original branch after many attempts, the original operator (the crossover operator or the self-evolution operator) will be discarded, and the random operator is enabled. A new branch is randomly produced in the design space by the random operator, and this new branch replaces the original branch regardless of whether the new branch is better or not. Through these three branch update operators, all branches in the population are updated and the branch population is also constantly updated from generation to generation. The best branch (thickest branch) in each generation of branch population is recorded.

Eventually, with the constant renewal of the branch population, the organic matters are delivered to the thickest tree trunk, which means the transfer of organic matters ends, and the best solution is found. The optimization process is over.

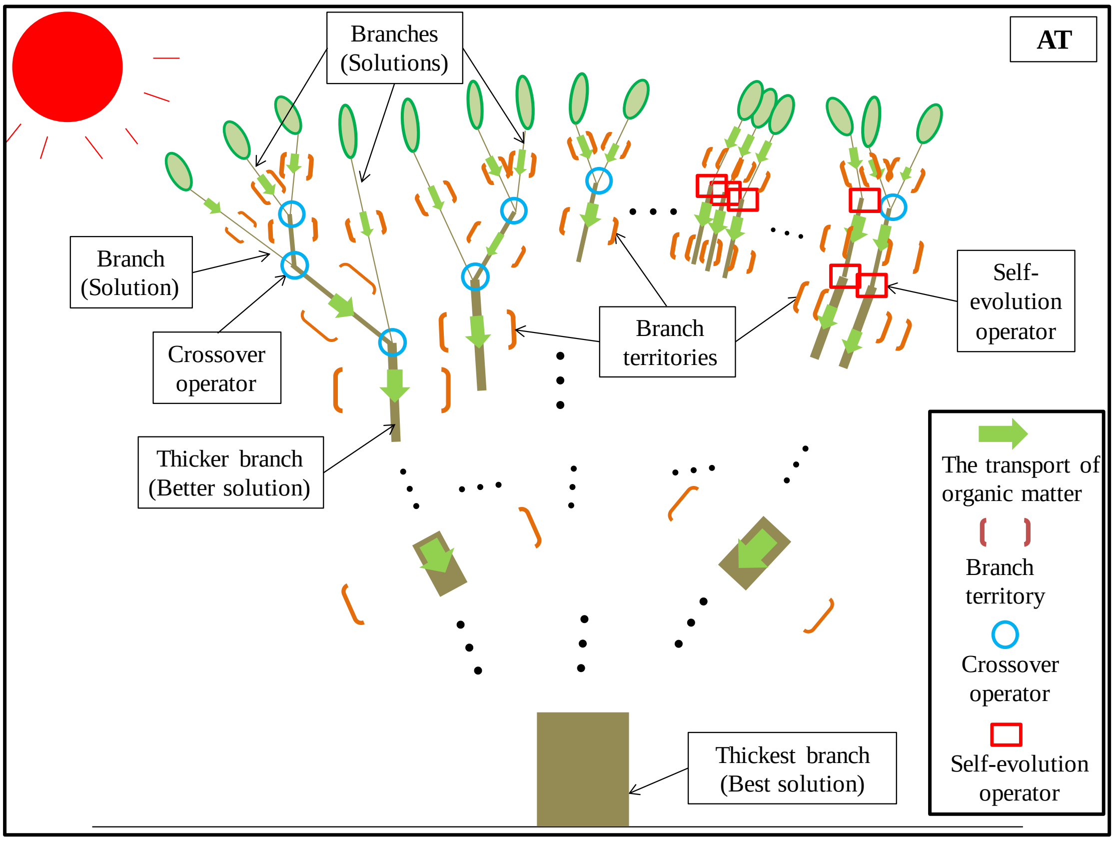


Fig. 1. The branches renewal process and the organic matters transfer process.

The whole optimization process can be summarized as follows: First, the branch population is randomly produced in the design space. Then, these branches are updated based on these three operators, and the branch population is also updated from generation to generation. The best solutions of all generation are obtained. Finally, the maximum number of function evaluation is reached, and the global best solution is acquired. The concepts of branch territory, crowd distance and branch update operators of AT are described in the next sections:

2.1 Branch territory

Each branch has one territory, and the range of the territory depends on the thickness of the branch. A thicker branch trends to have a larger branch territory. The equation of the branch territory is written as Eq. (1).

$$
\begin{aligned}
V_i &= (L + L \times fit(\mathbf{x}_i)) \times 2 \\
&= 2L(1 + fit(\mathbf{x}_i))
\end{aligned}
\tag{1}
$$

where L is the territory parameter which is defined in advance to calculate the branch territory. The value of L is recommended between 0 and 1. $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iD})$ is the spatial position of branch i and D is the dimension of the search space. $fit(\mathbf{x}_i)$ is the fitness value of $\mathbf{x}_i$. V_i is the branch territory which is a number, and different branches can have the same value of territory if their fitness values are the same. If $fit(\mathbf{x}_i) = 0$, the branch territory of $\mathbf{x}_i$ is $V_i = 2L$. In addition, the territory of branch i is a hypersphere with the branch position $\mathbf{x}_i$ as the center of the sphere and the radius of V_i. This work focuses on solving the minimization problem, and the equation of $fit(\mathbf{x}_i)$ is only suitable for the type of minimization problem, which is written as follows:

$$fit(\mathbf{x}_i) = \begin{cases} 1/(f(\mathbf{x}_i)+1) & if \quad f(\mathbf{x}_i) \geq 0 \\ 0 & if \quad f(\mathbf{x}_i) < 0 \end{cases} \tag{2}$$

where $f(\mathbf{x}_i)$ is the objective value of the solution $\mathbf{x}_i$, and the better solution $\mathbf{x}_i$ tends to have the higher values of $fit(\mathbf{x}_i)$ and V_i. The fitness value of solution is only used to calculate the branch territory and the maximum search number (Section 2.5). In AT, which branch is better is determined by directly comparing the solutions of different branches. Therefore, if for all $\mathbf{x}_i$, $f(\mathbf{x}_i) < 0$ (the global optimum solution of the optimization problem is a negative value), the fitness value of all solutions are the same, and the execution of AT algorithm is not affected.

2.2 Crowd distance

The crowd distance is applied to evaluate the spacing between the branches. The crowd distance between branch *i* and branch *j* is calculated as follows:

$$Dis_{ij} = \left\| \mathbf{x}_i - \mathbf{x}_j \right\|_2 \tag{3}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the positions of branch *i* and branch *j*, respectively, and Dis_{ij} is a number which is the spatial distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Based on the crowd distance and the branch territory, the concept of crowded tolerance *Tol* is put forward. For one branch whose branch position and territory are $\mathbf{x}_i$ and V_i, the crowd distance of branch *i* and all other branches can be calculated by Dis_{ij} (j=1, 2, …, Bn, $j \neq i$). *Bn* is the number of branch in the branch population. The other branches whose positions are in the territory of branch *i* can be calculated by the equation $Dis_{ij} < V_i$. Then, the number of branches in the territory of branch *i* is obtained and recorded as Nb_i. Whether this territory is crowded can be examined by comparing Nb_i with *Tol*. For the current branch position $\mathbf{x}_i$, if $Nb_i \leq Tol$, it implies that the branches in this territory are sparse. The crossover operator is implemented to update the branch. If $Nb_i > Tol$, which implies that the branches in current branch territory is crowded, the self-evolution operator is executed.

2.3 Crossover operator

A branch is randomly generated in half of the branch territory (a hypersphere which center is the branch position $\mathbf{x}_i$ and the radius is $V_i/2 = L \times (1 + fit(\mathbf{x}_i))$), and it combines with current branch by linear interpolation to produce a new branch. The mathematical model of the crossover operator is presented as follows:

$$\begin{aligned} x_{0j} &= x_{ij} + rand(-1,1) \times (V_i / 2) \\ &= x_{ij} + rand(-1,1) \times L \times (1 + fit(\mathbf{x}_i)) \end{aligned} \tag{4}$$

$$\mathbf{x}_{\text{new}} = rand(0,1) \times \mathbf{x}_0 + rand(0,1) \times \mathbf{x}_i \tag{5}$$

where j=1, 2,..., D, $rand(-1,1)$ is a random number between -1 and 1, $rand(0,1)$ is a random number between 0 and 1, $\mathbf{x}_0$ is the randomly generated branch position in the neighborhood of $\mathbf{x}_i$ which radius is $L \times (1 + fit(\mathbf{x}_i))$, and $\mathbf{x}_{\text{new}}$ is the position of the new produced branch. This new branch will be compared with the original branch to determine whether it can replace the original one (Section 2.5). By substituting Eq. (4) into Eq. (5), the crossover operator can be simplified as below:

$$x_{\text{new},j} = (k_1 + k_2) \times x_{ij} + k_1 \times rand(-1,1) \times L \times (1 + fit(\mathbf{x}_i)) \tag{6}$$

where $k_1 = rand(0,1)$ and $k_2 = rand(0,1)$.

2.4 Self-evolution operator

The mathematical model of this operator can be written as

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + rand(0,1) \times (\mathbf{x}_{\text{best}} - \mathbf{x}_i) \tag{7}$$

where $\mathbf{x}_{\text{best}}$ is the best branch position that has been found so far.

2.5 Random operator

The new branch produced by the crossover operator or the self-evolution operator is compared with the original branch. If the new branch is better than the original one, the new branch replaces the old one. Otherwise, the new branch is abandoned. Another new branch is generated, and a new comparison between this new branch and the original branch is carried out. Repeat this process until a better branch is found. If the better branch isn't found within a predefined number of cycles, the random operator is enabled. A new branch is randomly produced in the design space, and this new branch replaces the original branch. Obviously, the predetermined number of cycles is a important parameter for AT, which is called the maximum search number *Li*. For different branches, their maximum search number $Li(\mathbf{x}_\text{i})$ is different which can be calculated as follows:

$$\begin{aligned} Li(\mathbf{x}_i) &= N \times fit(\mathbf{x}_i) + N \\ &= N \times (1 + fit(\mathbf{x}_i)) \end{aligned} \tag{8}$$

where N is the search parameter which is a constant, $Li(\mathbf{x}_i)$ is the maximum search number of branch position $\mathbf{x}_i$ that is proportional to the fitness value $fit(\mathbf{x}_i)$. If $fit(\mathbf{x}_i) = 0$, the maximum search number of $\mathbf{x}_i$ is $Li(\mathbf{x}_i) = N$.

In order to fully study the AT algorithm, the following pseudocode is presented to illustrate the implementation process of the whole algorithm.

The artificial tree algorithm

Initialize the parameters *L*, *N*, *Tol*, *Bn* and *MEN* (the maximum function evaluation number)
Initialize the branch population $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_{Bn})$
Evaluate the initial population
repeat
Calculate the maximum search number of branch population $\mathbf{x}$
    for i=1 to *Bn* **do**
        for j=1 to $Li(\mathbf{x}_i)$ **do**
            if the territory of branch i is not crowd
                Perform the crossover operator to generate a new branch
            else
                Perform the self-evolution operator to generate a new branch
            end if
            If the new branch is better than the branch i
                Break out of the current For loop
            end if
        end for
        if a better branch compared with branch i is not found
            Perform the random operator to generate a new branch
        end if
        Update the branch i with the new branch
    end for
    Obtain the new branch population $\mathbf{x}$ of current generation
    Update the best solution $f(\mathbf{x}_{best})$ and the best variable $\mathbf{x}_{best}$ found so far
until (the function evaluation number reaches *MEN*)

3 The improved artificial tree algorithm

In nature, the transmission of organic matters from leaves to roots and the transport of moistures from roots to leaves ensure the normal growth of trees. In addition, the delivery of moistures is the feedback process of the transport of organic matters, and the branches which spread more organic matters get more moisture from the feedback. In this section, the feedback mechanism of moistures is introduced in AT. Therefore, FAT contains two processes: the transfer process of organic matters and the feedback process of moistures. The moistures are absorbed from the soil through the roots and passed to the thickest tree trunk. Then, the moistures pass through the thinner branches. Finally, they reach the leaves. Therefore, the moistures pass through all the branches, and the whole process is efficient.

The same as the transport of organic matters, there are also three update operators for the transfer of moistures. These operators are the self-propagating operator, the dispersive propagation operator and the random operator. In addition, the concepts that are used in the organic transfer process are still applicable to the moisture transfer process, such as branch territory, crowded tolerance, fitness value and maximum search number. The branch territory is also used to judge which operator should be selected. If the branches are crowded in one branch territory, the self-propagating operator takes place. Otherwise, the dispersive propagation operator is carried out. Differing from the organic matter transfer process, a thinner branch in the moisture feedback process represents a better solution. Therefore, in the organic matter transfer process, the thickest branch is the best solution, and in the moisture feedback process, the thinnest branch means the best solution. Meanwhile, the tiny branches in the bio-inspired model of a tree mean the local optimum solutions of the optimization problem. Regarding the dispersive propagation operator, when the moistures reach the junction of one branch, the branch is divided into two thinner branches. The moistures are then transferred from the thicker branch to the thinner branches. In this operator, two new thinner branches are found in half of the territory of the original branch. If both of these two new branches are thicker than the original one, these two new branches are abandoned, and another two new branches are generated again. If both of these two new branches are thinner than the original one, these two new branches are retained for the next optimization cycle, and the original branch is abandoned. If one of these two new branches is thinner than the original branch, the thinner new branch replaces the original one. The next optimization cycle is carried out with this new branch, and the original branch and another new generated branch are abandoned. If both of these two new branches are always thicker than the original one and the try number reaches the maximum search number, the random operator replaces the dispersive propagation operator. The second operator is the self-propagating operator. The same as the dispersive propagation operator, if the new generated branch is thinner than the original branch, the new branch substitutes the original one. Otherwise, the new branch is discarded. If a thinner branch is not found after the attempt number reaches the maximum search number, the random operator takes the place of the self-propagating operator.

In FAT, a parameter of maximum update number h of the branch population is defined for the organic matter transfer process. The whole optimization process of FAT is summarized as follows: First, the transfer of organic matters and the update of branches begin. When the update number of branch population reaches h, the organic matters transfer process ends and the delivery of moistures begins. Then, some branches are randomly selected from the branch population that is acquired from the organic matters transfer process to implement the feedback operation. It should be noted that the initial branch number used in the feedback process of moistures is less than or equal to the branch number Bn. In the feedback process, the branches found in previous cycle and current cycle merge together to be the initial branch population of the next cycle. Therefore, the branch number in feedback process increases with the increase of the cycle number. When the branches found in the feedback process exceeds the branch number Bn, the feedback process ends. Then, the branches found by the feedback process and previous organic matter transfer process are put together, and *Bn* better branches are selected as the initial branch population for the next cycle of the organic matter transfer process. The organic matter transfer process and the moisture feedback process are continuously performed until the maximum number of function evaluation is reached, and these two processes form the entire FAT algorithm. The key operations of the feedback process of FAT are summarized as follows:

3.1 Initialize the branch population of feedback process

When the feedback process begins, some branches are randomly selected from the branch population that is acquired from the organic matter transfer process. The selection process is calculated by Eq. (9).

$$\mathbf{x}_{\text{new}} = randchoose(\mathbf{x}, r) \tag{9}$$

where $\mathbf{x}$ is the branch population, $\mathbf{x}_{\text{new}}$ is the selected branch population and r is the ratio of the new selected branches to the branch population.

3.2 Self-propagating operator

The concepts of branch territory, crowded tolerance, fitness value and maximum search number that are used in the organic matters transfer process are also applied in the feedback process of moistures. For branch position $\mathbf{x}_i$, if its branch territory is crowded ($Nb_i > Tol$), the self-propagating operator is carried out to renew the branch. The mathematical expression of the self-propagating operator is given as follows:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + (rand(0,1) \times \mathbf{x}_{\text{best}} - rand(0,1) \times \mathbf{x}_i) \times c \tag{10}$$

where c is a small constant. 0.382 is a relatively reasonable value, which comes from the golden section theory. In this work, we recommend c as 0.382. Because, it can ensure the computational efficiency of the self-propagating operator while taking into account the computational accuracy.

3.3 Dispersive propagation operator

If $Nb_i \leq Tol$, the dispersive propagation operator is carried out to achieve the evolution of branch i. One new branch is produced randomly within its half territory, and the other branch is found based on the positions of the original branch and the new branch. The mathematical models of this operator are shown as follows:

$$x_{oj} = x_{ij} + rand(-1,1) \times (Vi / 2) \times c \tag{11}$$

$$x_{tj} = x_{ij} - rand(0,1) \times x_{oj} \tag{12}$$

where x_{oj} and x_{tj} are the j-th element of $\mathbf{x}_o$ and $\mathbf{x}_t$, j=1, 2, …, D. $\mathbf{x}_o$ and $\mathbf{x}_t$ are the produced two branch positions.

The growth behavior of a tree is depicted in Fig. 2. Unlike Fig. 1, the feedback process of moistures is also illustrated. In Fig. 2, the transport of organic matters is from the leaves to the roots and the feedback of moistures is from the roots to the leaves. The transport of organic matters and moistures depends on the update of branches. The update of branches in the organic matter transport process finds the thicker branches, while, the renewal of branches in the moisture feedback process searches for the thinner branches. Therefore, the thickest branch and the thinnest branch represent the best solution in the organic matter transfer process and the moisture feedback process, respectively.

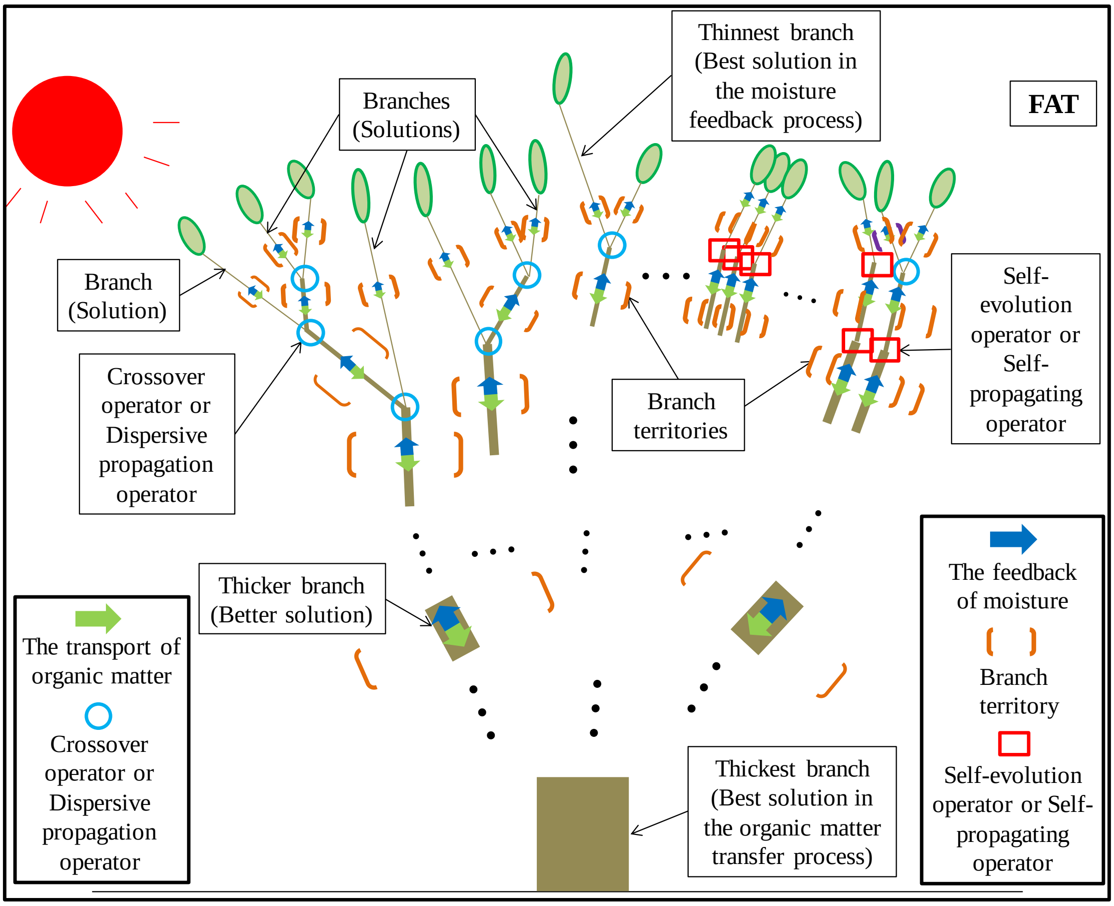


Fig. 2. Materials exchange process of a tree.

The following pseudo-code shows the implementation process of the entire FAT algorithm.

The feedback artificial tree algorithm

Initialize the parameters L, N, Tol, Bn, c, r, h and MEN (maximum function evaluation number)
Initialize the branch population $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_{Bn})$
Evaluate the initial population
cycle = 1
repeat
    Count the number of branches in the branch population S
    if $S \geq Bn$
        if cycle > 1
            Combine current branch population $\mathbf{x}$ and branch population obtained by previous organic matter

transfer process into a new group of branches
Consider *Bn* better branches among the new group of branches as the initial branch population $\mathbf{x}$ for the organic matter transfer process
end if
for j = 1 to h **do**
for i=1 to *Bn* **do**
for k = 1 to $Li(\mathbf{x}_i)$ **do**
if the territory of branch i is not crowd
Perform the crossover operator to generate a new branch
else
Perform the self-evolution operator to generate a new branch
end if
If the new branch is better than the branch i
Break out of the current For loop
end if
end for
if a better branch compared with branch i is not found
Perform the random operator to generate a new branch
end if
Update the branch i with the new branch
end for
Obtain the new branch population $\mathbf{x}$ of current generation
Update the best solution $f(\mathbf{x}_{\text{best}})$ and the best variable $\mathbf{x}_{\text{best}}$ found so far
end for
Select the initial branch population for the feedback process $\mathbf{x} = randchoose(\mathbf{x}, r)$
else
for i=1 to S **do**
for k = 1 to $Li(\mathbf{x}_i)$ **do**
if the territory of branch i is not crowd
Perform the dispersive propagation operator to generate a new branch
else
Perform the self-propagating operator to produce a new branch
end if
If the new branch is better than the branch i
Break out of the current For loop
end if
end for
if a better branch compared with branch i is not found
Perform the random operator to generate a new branch
end if
Update the branch i with the new branch
end for
Obtain the new branch population $\mathbf{x}$ of current generation
Update the best solution $f(\mathbf{x}_{\text{best}})$ and the best variable $\mathbf{x}_{\text{best}}$ found so far

Combine current branch population with previous branch population found by the feedback process into a new branch population **x**

end if

cycle = cycle+1

until (the function evaluation number reaches *MEN*)

4 Numerical experiments

Experimental analyses are conducted with various problems, and results of FAT are compared with AT and some other algorithms to fully study the performance of FAT. In addition, these experiments are calculated under the professional version of the Win10 operating system, and the computer hardware used included 8 GB RAM and an Intel (R) Core (TM) i5-6200U 2.40 GHz processor. Meanwhile, all the algorithms that appear in the work are coded in Matlab.

4.1. Comparison between FAT and AT

The performance of FAT is first compared with AT. The branch population *Bn*, territory parameter *L*, crowded tolerance *Tol* and search parameter *N* are set as 50, 0.5, 1 and 10 for both AT and FAT. Furthermore, regarding FAT, the additional parameters of *r* and *h* are set as 0.2 and 20. The maximum number of function evaluation for AT and FAT is set as 400,000. Thirty independent runs are carried out on all instances with different random seeds, and the results of FAT and AT contain the standard deviations (SDs), means, medians and best of these test problems. It is regarded as 0 when the computational result is less than 10^{-20}. In addition, in order to avoid the familywise errors and make a more complete comparison of AT and FAT, statistical tests (Derrac et al. 2011; Zhu et al. 2013; Guo et al. 2014) are also applied.

4.1.1 Comparing the results of FAT and AT with low dimensional problems

Twenty typical low dimensional problems (Zhan et al. 2009) exhibited in Tab. A1 of the appendix are applied to evaluate these two algorithms. The formulations, dimensions (D) of these problems, intervals of the design variables and the global optimum solutions are also presented in Tab. A1.

Table 1. Experimental results of AT and FAT on the twenty low dimensional problems.

	Michalewicz2				Michalewicz5			
	Best	Mean	SD	Median	Best	Mean	SD	Median
AT	-1.80130341	**-1.80130341**	4.44000E-16	-1.80130341	-4.645895368	-4.42938384	0.220501687	-4.49589321
FAT	-1.80130341	**-1.80130341**	1.18424E-16	-1.80130341	-4.659528776	**-4.523603835**	0.041927844	-4.537655997
	Michalewicz10				Langerman2			
AT	-6.782176134	-5.96383290	0.283984	-5.88952323	-1.080938442	**-1.08093844**	1.98603E-16	-1.08093844
FAT	-8.330611638	**-6.92200258**	0.161377093	-6.88893566	-1.080938442	**-1.080938442**	5.53877E-17	-1.080938442
	Langerman5				Langerman10			

AT	-1.495166236	-1.21376744	0.321472145	-1.46582165	-0.797593898	-0.32610607	0.17603243	-0.35576874
FAT	-1.499999223	**-1.309658579**	0.075966881	-1.499998025	-0.797693836	**-0.407399426**	0.067437089	-0.254625007
	Hartman3				Hartman6			
AT	-3.862782033	-3.86242638	0.000240651	-3.86242172	-3.321988659	-3.27061395	0.072856518	-3.32192056
FAT	-3.862782146	**-3.862564323**	6.2259E-05	-3.862646005	-3.321989121	**-3.302893332**	0.013459067	-3.32196861
	Shekel5				Shekel7			
AT	-10.15319968	-10.1027863	0.188625298	-10.1531997	-10.40294057	-10.4024158	0.001963436	-10.4029406
FAT	-10.15319968	**-10.15319968**	4.59158E-05	-10.15319968	-10.40294057	**-10.40294057**	2.91338E-09	-10.40294057
	Shekel10				Kowalik			
AT	-10.53640982	-10.5364063	1.30363E-05	-10.5364098	0.000307486	0.000315279	2.91E-05	0.000307487
FAT	-10.53640982	**-10.53640982**	3.827321E-15	-10.53640982	0.000307486	**0.000307546**	3.18E-08	0.000307493
	Foxholes				Ackley			
AT	0.998003838	**0.998003838**	2.547880E-16	0.998003838	2.75335E-14	6.89819E-14	4.58776E-14	6.30607E-14
FAT	0.998003838	**0.998003838**	4.186913E-17	0.998003838	2.66454E-15	**3.99680E-15**	4.58653E-16	2.66454E-15
	SixHumpCamelBack				Penalized			
AT	-1.031628453	**-1.031628453**	0	-1.031628453	0.055872317	0.088094129	0.017111219	0.091456558
FAT	-1.031628453	**-1.031628453**	0	-1.03162845	0.006871617	**0.012386788**	0.00135093	0.010688062
	Penalized2				FletcherPowell2			
AT	0.326510992	0.531771435	0.129141776	0.521481417	0	**0**	0	0
FAT	0.190594683	**0.423777906**	0.039149167	0.398159016	0	**0**	0	0
	FletcherPowell5				FletcherPowell10			
AT	1.935922879	67.2644496	85.1924839	33.1318967	430.1475651	3554.484661	2363.81321	3371.711309
FAT	0.010854798	**4.886797808**	3.996586195	4.32989828	55.22882953	**1778.8479138**	499.005175	822.7358134

Table 2. Comparison between AT and FAT based on t test and Wilcoxon rank sum test.

Function	Michalewicz2	Michalewicz5	Michalewicz10	Langerman2	Langerman5	Langerman10	Hartman3
t test	≈	+	+	≈	+	+	+
Wilcoxon	≈	≈	+	≈	+	≈	+
Function	Hartman6	Shekel5	Shekel7	Shekel10	Kowalik	Foxholes	Ackley
t test	+	+	+	+	+	≈	+
Wilcoxon	+	+	≈	+	+	≈	+
Function	SixHump CamelBack	Penalized	Penalized2	FletcherPowell2	FletcherPowell5	FletcherPowell10	
t test	≈	+	+	≈	+	+	
Wilcoxon	≈	+	+	≈	≈	+	

Note: "≈", "-" and "+" mean the result of FAT is equal to, worse than and better than that of AT, respectively, based on t test and Wilcoxon rank sum test at a significance level 0.05.

Table 3. Comparisonal results between AT and FAT.

	FAT better	FAT worse	FAT equal	Success rate
Computational results	15	0	5	100.00%
t test	15	0	5	100.00%

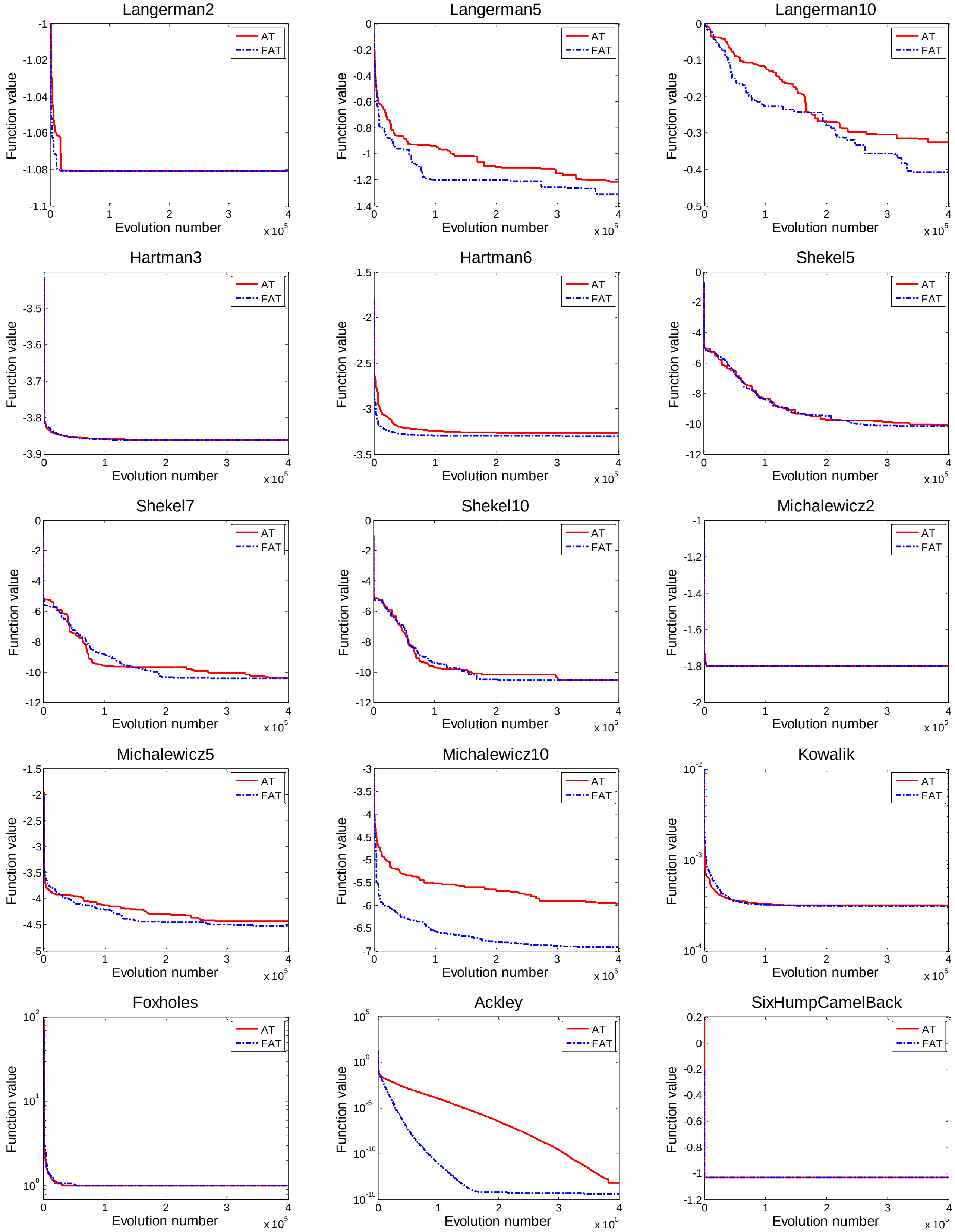
Wilcoxon	11	0	9	100.00%

Note: “FAT equal”, “FAT worse” and “FAT better” are the number of results of FAT that are equal to, worse than and better than that of AT, respectively. “Success rate” is the ratio of the sum of the numbers of "FAT better" and "FAT equal" to the total number of the test functions.

Tables 1 and 3 show the computational results, and the better solutions of mean values in the Tab. 1 are mark as bolded and gray. The convergence curves of these test functions calculated by AT and FAT are presented in Fig. 3. From Tab. 1 and Fig. 3, the results of FAT are better than those of AT for functions Michalewicz5, Michalewicz10, Langerman5, Langerman10, Hartman3, Hartman6, Shekel5, Shekel7, Shekel10, Kowalik, Ackley, Penalized, Penalized2, FletcherPowell5 and FletcherPowell10. Regarding the problems Michalewicz2, Langerman2, Foxholes, SixHumpCamelBack and FletcherPowell2, FAT and AT exhibit the same results. The results of FAT are better than those of AT on fifteen functions, and the same results are achieved on five problems by FAT and AT. In addition, when the overall performances of these two algorithms on functions Michalewicz2, Langerman2 and Foxholes are considered, the results of FAT are also better than those of AT for its smaller value of SDs. In Fig. 3, we can see that the convergence rates of FAT are significantly better than those of AT for problems Langerman5, Langerman10, Michalewicz5, Michalewicz10, Penalized, Ackley, FletcherPowell2, FletcherPowell5 and FletcherPowell10, slightly better than AT on functions Hartman6, Langerman2, and similar to AT on instances Hartman3, Michalewicz2, Shekel5, Shekel7, Shekel10, Kowalik, Foxholes, SixHumpCamelBack and Penalized2. Therefore, from the computational results of FAT and AT, it is clear that the performance of FAT is obviously better than AT with the same control parameters for the considered set of problems.

In addition, t test and Wilcoxon rank sum test between AT and FAT at a significance level of 0.05 are performed, and the symbolic results are shown in Tab. 3. Obviously, based on t test, the results of FAT are better than those of AT on questions Michalewicz5, Michalewicz10, Langerman5, Langerman10, Hartman3, Hartman6, Shekel5, Shekel7, Shekel10, Kowalik, Ackley, Penalized, Penalized2, FletcherPowell5 and FletcherPowell10. Therefore, the results of FAT are significantly better than those of AT on fifteen problems. The results of FAT and AT on functions Michalewicz2, Langerman2, Foxholes, SixHumpCamelBack and FletcherPowell2 are not significant, which means the performance of FAT is similar as AT on these five questions. Based on the results of Wilcoxon rank sum test, FAT performs better than AT on functions Michalewicz10, Langerman5, Hartman3, Hartman6, Shekel5, Shekel10, Kowalik, Ackley, Penalized, Penalized2 and FletcherPowell10. The results between FAT and AT are not significant on instances Michalewicz2, Michalewicz5, Langerman2, Langerman10, Shekel7, Foxholes, SixHumpCamelBack, FletcherPowell2 and FletcherPowell5, which means the performances of FAT and AT on these questions are similar. On the whole, FAT obtains eleven better and nine similar results compared with

AT. Based on the results of t test and Wilcoxon rank sum test, the better performance of FAT than that of AT is proved for this set of problems.



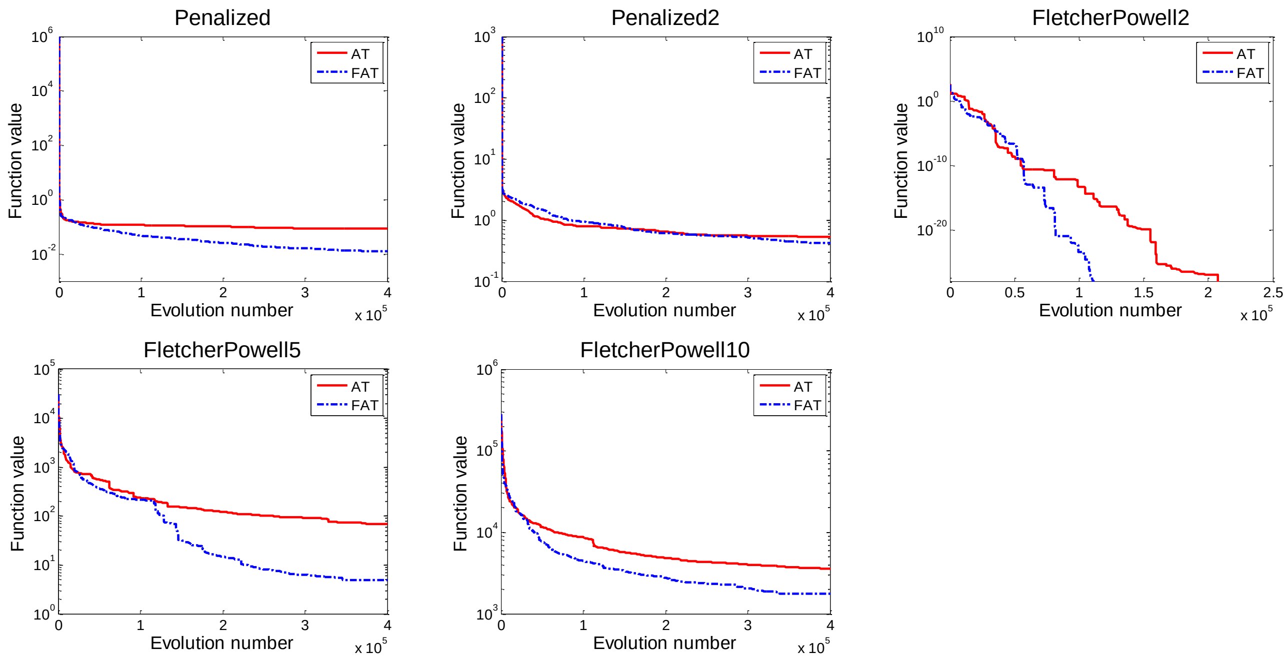


Fig. 3. Convergence curves of AT and FAT on the twenty low dimensional functions.

4.1.2 Comparing the results of FAT and AT with high dimensional problems

Table A2 of the appendix shows ten high dimensional benchmark problems. The dimensions of these test problems are taken as 30, 60, 90, 200, 500 and 1000, respectively.

Table 4. Experimental results of AT and FAT on functions Sphere, Rosenbrock, Dixon-Price, SumSquares and Matyas.

Function	Algorithm		D					
			30	60	90	200	500	1000
Sphere	FAT	Best	0	0	0	0	0	0
		Mean	**0**	**0**	**0**	**0**	**0**	**0**
		SD	0	0	0	0	0	0
		Median	0	0	0	0	0	0
	AT	Best	1.69267E-23	2.21553E-13	9.31514E-11	2.47352E-07	1.16405E-05	1.69555E-05
		Mean	5.1805E-21	2.25748E-12	8.91419E-10	4.28442E-07	1.45254E-05	2.14519E-05
		SD	1.38502E-20	7.86863E-13	2.23293E-10	1.23352E-07	2.37247E-06	2.77907E-06
		Median	5.70929E-22	8.39064E-13	6.74929E-10	4.42591E-07	1.37283E-05	2.3168E-05
	Significance	t-test	+	+	+	+	+	+
		Wilcoxon	+	+	+	+	+	+
Rosenbrock	FAT	Best	27.15386414	58.047164098	87.92979069	197.0064526	495.4844460	994.6042921
		Mean	**27.32005766**	**58.12778137**	**87.98373192**	**197.0622829**	**495.6779665**	**994.9096085**
		SD	0.020920886	0.014897096	0.007859538	0.008688401	0.022698987	0.051108497
		Median	27.33311413	58.12758456	87.98291386	197.0574128	495.6805542	994.8731007
	AT	Best	28.39216926	58.24939911	88.02155187	197.3486341	496.8698456	996.6506896
		Mean	28.40877175	58.26310797	88.03458021	197.373001	496.9035914	996.7370544
		SD	0.011321872	0.002629604	0.00218016	0.016533911	0.020767969	0.057767069
		Median	28.41046769	58.26360899	88.0352612	197.3751131	496.9175435	996.7473747
	Significance	t-test	+	+	+	+	+	+

		Wilcoxon	+	+	+	+	+	+
Dixon-Price	FAT	Best	0.666715640	0.66830635	0.70457456	0.899926091	0.990270541	0.998356926
		Mean	**0.667026551**	**0.67604095**	**0.718262868**	**0.918625838**	**0.991815652**	**0.998557738**
		SD	9.00876E-05	0.001297306	0.001668671	0.002490562	0.000256784	2.85058E-05
		Median	0.666864315	0.676067685	0.719408317	0.917704986	0.991898705	0.998555783
	AT	Best	0.701327551	0.795843934	0.851678463	0.94733153	0.989152337	0.998826485
		Mean	0.706219945	0.803562183	0.861727159	0.948617533	0.992657681	1.003501717
		SD	0.003280477	0.00096478	0.001303788	0.000926207	0.00095663	0.002865744
		Median	0.705717999	0.803078686	0.86081954	0.948571067	0.991048777	1.005661487
	Significance	t-test	+	+	+	+	+	+
		Wilcoxon	+	+	+	+	≈	+
SumSquares	FAT	Best	0	0	0	0	0	0
		Mean	**0**	**0**	**0**	**0**	**0**	**0**
		SD	0	0	0	0	0	0
		Median	0	0	0	0	0	0
	AT	Best	4.24464E-10	8.0375E-07	9.13508E-07	0.000121316	0.000910345	0.002195447
		Mean	1.24063E-07	6.0941E-06	2.07955E-05	0.000156299	0.001377147	0.00731237
		SD	1.45204E-07	1.22844E-06	3.9354E-06	3.73299E-05	0.000285891	0.003329647
		Median	5.95629E-08	4.92685E-06	1.50279E-05	0.000130556	0.001604237	0.002195447
	Significance	t-test	+	+	+	+	+	+
		Wilcoxon	+	+	+	+	+	+
Matyas	FAT	Best	0	0	0	0	0	0
		Mean	**0**	**0**	**0**	**0**	**0**	**0**
		SD	0	0	0	0	0	0
		Median	0	0	0	0	0	0
	AT	Best	0	0	0	0	0	0
		Mean	0	0	0	0	0	0
		SD	0	0	0	0	0	0
		Median	0	0	0	0	0	0
	Significance	t-test	≈	≈	≈	≈	≈	≈
		Wilcoxon	≈	≈	≈	≈	≈	≈

Note: "≈", "-" and "+" mean the result of FAT is equal to, worse than and better than that of AT, respectively, based on t test and Wilcoxon rank sum test at a significance level 0.05.

Table 5. Experimental results of AT and FAT on functions Schwefel2.2, Quartic, Schaffer, Griewank and Rastrigin.

Function	Algorithm		D					
			30	60	90	200	500	1000
Schwefel2.2	FAT	Best	0	0	0	0	0	0
		Mean	**0**	**0**	**0**	**0**	**0**	**0**
		SD	0	0	0	0	0	0
		Median	0	0	0	0	0	0
	AT	Best	0.003730566	0.004405071	0.005423725	0.024662852	0.070144873	0.075200133
		Mean	0.018799917	0.021805605	0.025595841	0.029069422	0.077326457	0.113991097
		SD	0.007982416	0.002586229	0.00294474	0.002826466	0.006621005	0.036725132

		Median	0.015778937	0.021001448	0.026322344	0.030083408	0.073917673	0.133130476
	Significance	t-test	+	+	+	+	+	+
		Wilcoxon	+	+	+	+	+	+
Quartic	FAT	Best	0.000012464	0.000043197	0.00006612	0.00006967	0.000165681	0.000252689
		Mean	0.000752416	**0.00060042**	**0.000570353**	**0.00057419**	**0.001075424**	**0.001275274**
		SD	0.000153371	0.000101048	8.23611E-05	9.97686E-05	0.000208688	0.000204068
		Median	0.000585214	0.000504282	0.000517597	0.0004223	0.000931744	0.0011872
	AT	Best	8.22168E-05	9.43344E-05	3.49135E-05	0.000861139	0.00076448	0.001044042
		Mean	**0.000505072**	0.001025834	0.000682715	0.001045561	0.001381573	0.002106071
		SD	0.000336718	0.000155989	0.000159217	0.000798559	0.000644802	0.000656693
		Median	0.000507457	0.000871769	0.000475345	0.002165275	0.002428995	0.002508386
	Significance	t-test	-	+	+	+	+	+
		Wilcoxon	≈	+	≈	+	≈	+
Schaffer	FAT	Best	0	0	0	0	0	0
		Mean	**0**	**0**	**0**	**0**	**3.18044E-14**	**1.1531E-08**
		SD	0	0	0	0	2.81913E-14	9.12554E-09
		Median	0	0	0	0	5.55112E-17	1.30407E-11
	AT	Best	0	0	0	0	2.74957E-10	9.97541E-09
		Mean	**0**	1.11126E-09	5.03301E-15	3.95672E-10	1.98626E-07	2.90710E-07
		SD	0	1.15027E-09	4.84864E-15	3.99027E-10	2.37392E-07	2.86589E-07
		Median	0	0	0	1.10190E-14	5.86249E-07	7.55357E-07
	Significance	t-test	≈	+	+	+	+	+
		Wilcoxon	≈	+	+	+	+	+
Griewank	FAT	Best	0	0	0	0	0	0
		Mean	**0**	**0**	**0**	**0**	**0**	**6.93889E-18**
		SD	0	0	0	0	0	7.16646E-18
		Median	0	0	0	0	0	0
	AT	Best	5.55112E-16	7.39631E-13	1.32268E-09	1.96949E-08	1.37111E-07	1.44704E-07
		Mean	2.03541E-15	2.66742E-10	7.84176E-09	1.02537E-07	1.96081E-07	2.56865E-07
		SD	8.55527E-16	4.52484E-11	1.09531E-09	5.85445E-08	6.80902E-08	7.224E-08
		Median	1.88738E-15	2.13385E-10	7.55252E-09	1.85284E-07	1.43929E-07	2.8129E-07
	Significance	t-test	+	+	+	+	+	+
		Wilcoxon	+	+	+	+	+	+
Rastrigin	FAT	Best	0	0	0	0	0	0
		Mean	**0**	**0**	**0**	**0**	**0**	**0**
		SD	0	0	0	0	0	0
		Median	0	0	0	0	0	0
	AT	Best	1.77636E-15	4.01457E-13	7.70192E-09	1.55548E-05	0.001649997	0.003680249
		Mean	1.12503E-14	7.91071E-12	3.03403E-08	3.68564E-05	0.001767723	0.004735443
		SD	6.14126E-15	3.11178E-12	4.31611E-09	1.41236E-05	7.42357E-05	0.00065103
		Median	1.06581E-14	3.02158E-12	2.42853E-08	3.98502E-05	0.001801541	0.00515077
	Significance	t-test	+	+	+	+	+	+
		Wilcoxon	+	+	+	+	+	+

Note: "≈", "-" and "+" mean the result of FAT is equal to, worse than and better than that of AT, respectively, based on t test and

Wilcoxon rank sum test at a significance level 0.05.

Table 6. Comparison results of AT and FAT on the ten high dimensional problems.

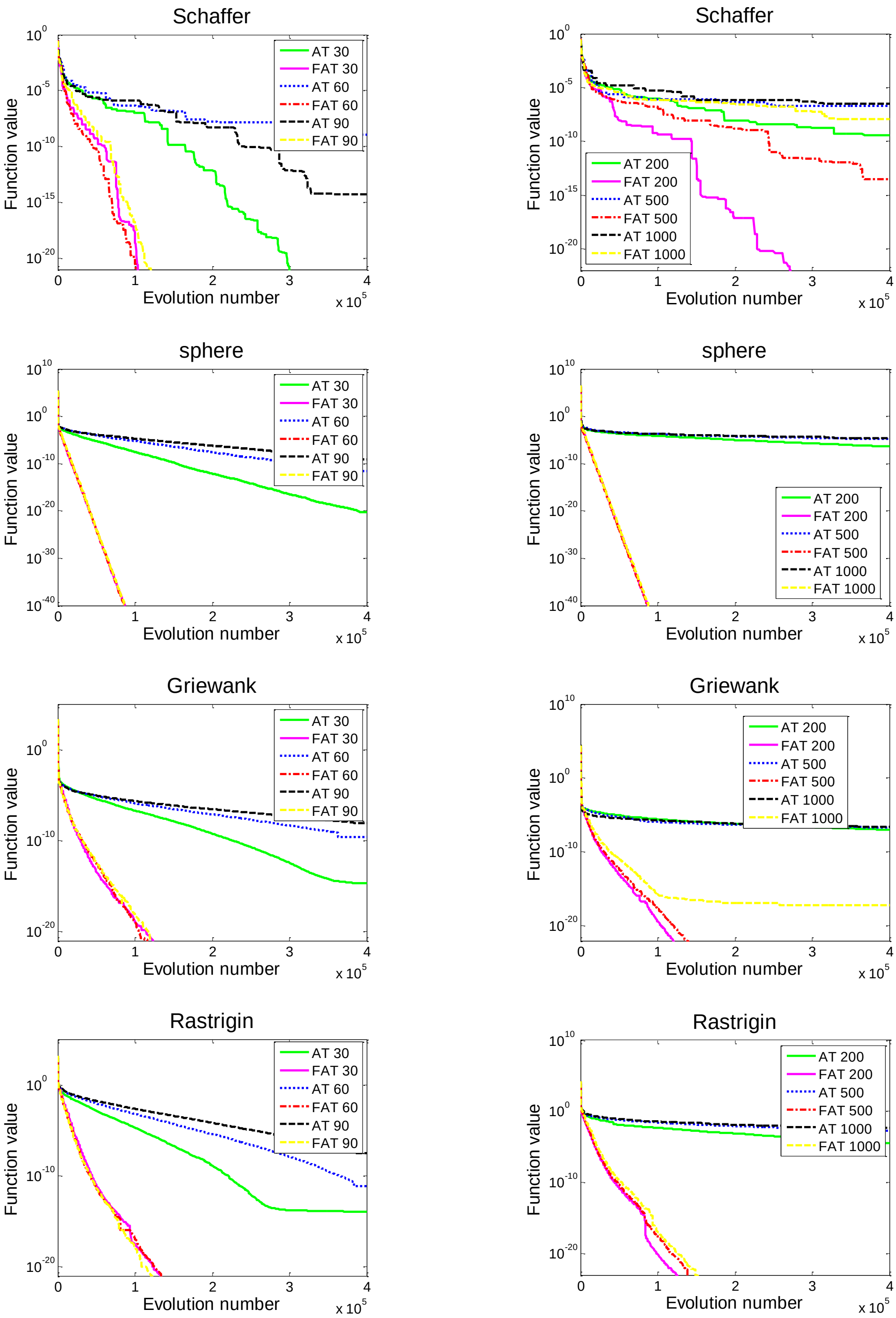
	D	FAT better	FAT worse	FAT equal	Success rate
Results	30	7	1	2	90.00%
	60	9	0	1	100.00%
	90	9	0	1	100.00%
	200	9	0	1	100.00%
	500	9	0	1	100.00%
	1000	9	0	1	100.00%
t test	30	7	1	2	90.00%
	60	9	0	1	100.00%
	90	9	0	1	100.00%
	200	9	0	1	100.00%
	500	9	0	1	100.00%
	1000	9	0	1	100.00%
Wilcoxon	30	7	0	3	100.00%
	60	9	0	1	100.00%
	90	8	0	2	100.00%
	200	9	0	1	100.00%
	500	7	0	3	100.00%
	1000	9	0	1	100.00%

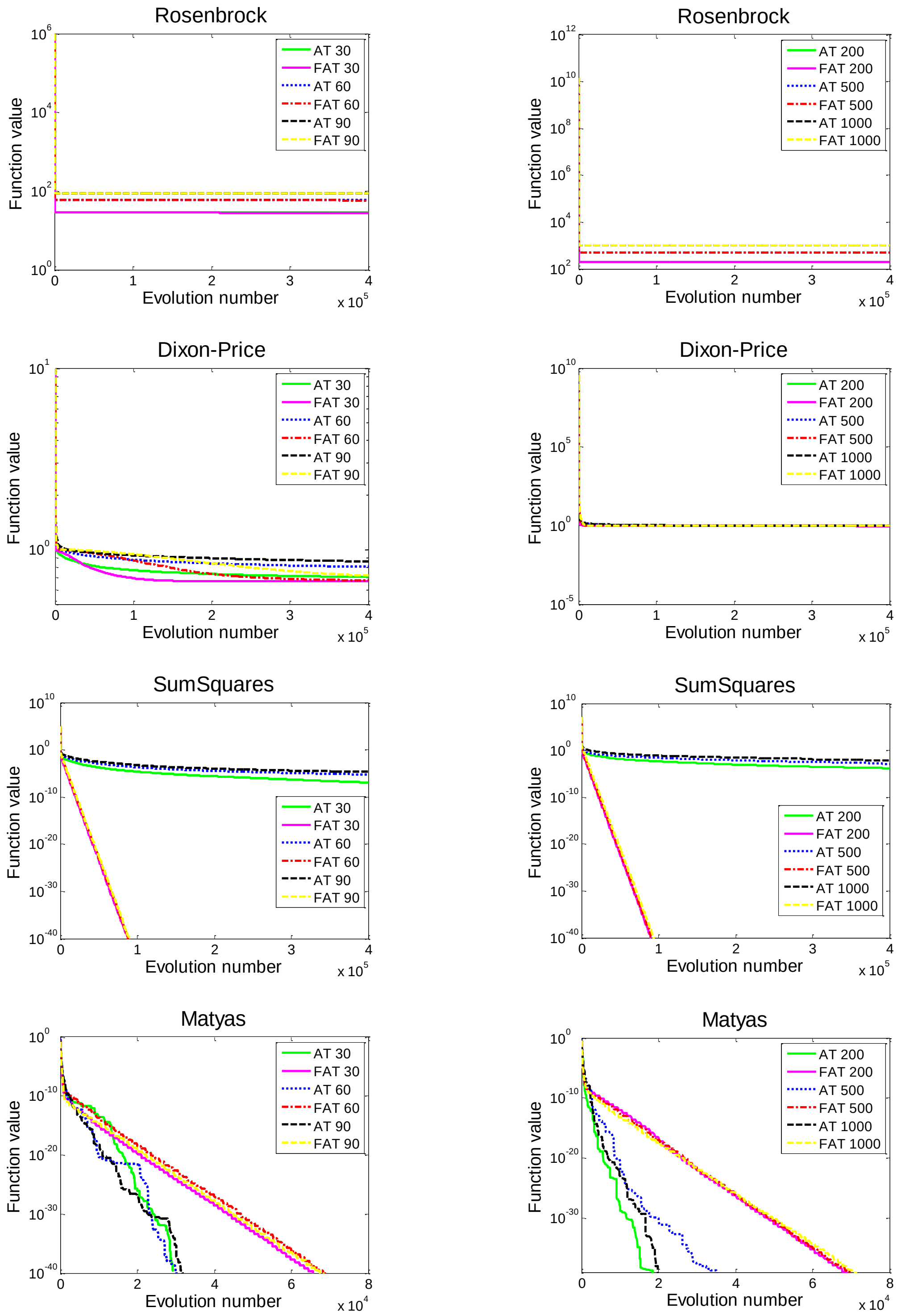
Note: “FAT equal”, “FAT worse” and “FAT better” are the number of results of FAT that are equal to, worse than and better than that of AT, respectively. “Success rate” is the ratio of the sum of the numbers of "FAT better" and "FAT equal" to the total number of the test functions.

The comparison results between AT and FAT on the ten high dimensional problems are presented in Tabs. 4 - and 6. In addition, the better solutions of mean values in the Tabs. 4 and 5 are mark as bolded and gray. From Tabs. 4 - 6, as the dimension of test problems increases, the results of FAT and AT deteriorate, and the worst results of each problem are obtained at their 1000 dimensional states. Regarding the problems Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Griewank and Rastrigin, no matter what the dimensions of these problems are (30, 60, 90, 200, 500 or 1000), the results of FAT are better than those of AT. In addition, it is noticed that both FAT and AT find the optimum solutions on 30 dimensional Schaffer, and FAT performs better than AT for its 60, 90, 200, 500 and 1000 dimensional conditions. For 30 dimensional Quartic, AT hits the better result compared with FAT. However, FAT performs better than AT on the 60, 90, 200, 500 and 1000 dimensional Quartic. Regarding 30, 60, 90, 200, 500 and 1000 dimensional Matyas, both FAT and AT acquire the optimum solutions. Therefore, FAT performs better than AT for seven problems with their 30, 60, 90, 200, 500 and 1000 dimensional conditions as well as two functions with their 60, 90, 200, 500 and 1000 dimensional states. In addition, FAT obtains the similar results as AT on one problem with its 30, 60, 90, 200, 500 and 1000 dimensional states and one problem with its 30 dimensional condition. Figure 4 shows the convergence curves of FAT and AT with different dimensions. The left column of Fig. 4 exhibits the convergence curves of AT and FAT with 30, 60, and 90 dimensional functions, and the right column is the results of 200, 500 and 1000 dimensional functions. From Fig. 4, it is obvious that the convergence speed of FAT is better than that of AT on functions Schaffer, Sphere, Rosenbrock, SumSquares, Schwefel2.2, Griewank and Rastrigin. Therefore, these experimental results and figures demonstrate that the performance of FAT is obviously better than that of AT.

The t test and Wilcoxon rank sum test of each problem at a 0.05 significance are also performed, and the results are also shown in Tabs. 4 and 5. According to the t test, the results of FAT are better than those of AT for functions Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Griewank and Rastrigin with their 30, 60, 90, 200, 500 and 1000 dimensional states. Regarding functions Quartic and Schaffer, the performances of FAT are better than those of AT on their 60, 90, 200, 500 and 1000 dimensional states. In addition, as the results of FAT and AT on functions Matyas and 30 dimensional Schaffer are not significant, the performance of FAT is almost the same as AT. In short, FAT gives better results compared with AT for seven problems with their all dimensional conditions and two problems with their 60, 90, 200, 500 and 1000 dimensional statuses. In addition, the t test results of FAT and AT are similar with each other for one function with its all dimensional conditions and one function with its 30 dimensional status.

Based on the Wilcoxon rank sum test, FAT performs better than AT on instances Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Griewank and Rastrigin with their 30, 60, 90, 200, 500 and 1000 dimensional states. Regarding problem Schaffer, the performances of FAT are better than those of AT on its 60, 90, 200, 500 and 1000 dimensional states. For Quartic, the results of FAT are better on its 60, 200 and 1000 states compared with AT. In addition, the results of FAT and AT on functions Matyas, 30 dimensional Schaffer as well as 30, 90 and 500 dimensional Quartic are not significant, and the performances of FAT and AT are similar. It is clear FAT obtains better results on seven problems with their all dimensional conditions, one function with its 60, 90, 200, 500 and 1000 dimensional statuses and one instance with its 90, 200 and 1000 dimensional states compared with AT. In addition, the t-test results of FAT and AT are similar with each other for one problem with its all dimensional conditions, one function with its 30 dimensional status and one instance with its 30, 60 and 500 dimensional states. Therefore, based on the results of t test and Wilcoxon rank sum test, the performance of FAT is demonstrated.


Schaffer
sphere
Griewank
Rastrigin
Function value
Evolution number
x 10^5
AT 30
FAT 30
AT 60
FAT 60
AT 90
FAT 90
AT 200
FAT 200
AT 500
FAT 500
AT 1000
FAT 1000


Rosenbrock
Function value
Evolution number
x 10^5
AT 30
FAT 30
AT 60
FAT 60
AT 90
FAT 90
Rosenbrock
AT 200
FAT 200
AT 500
FAT 500
AT 1000
FAT 1000
Dixon-Price
SumSquares
Matyas
x 10^4

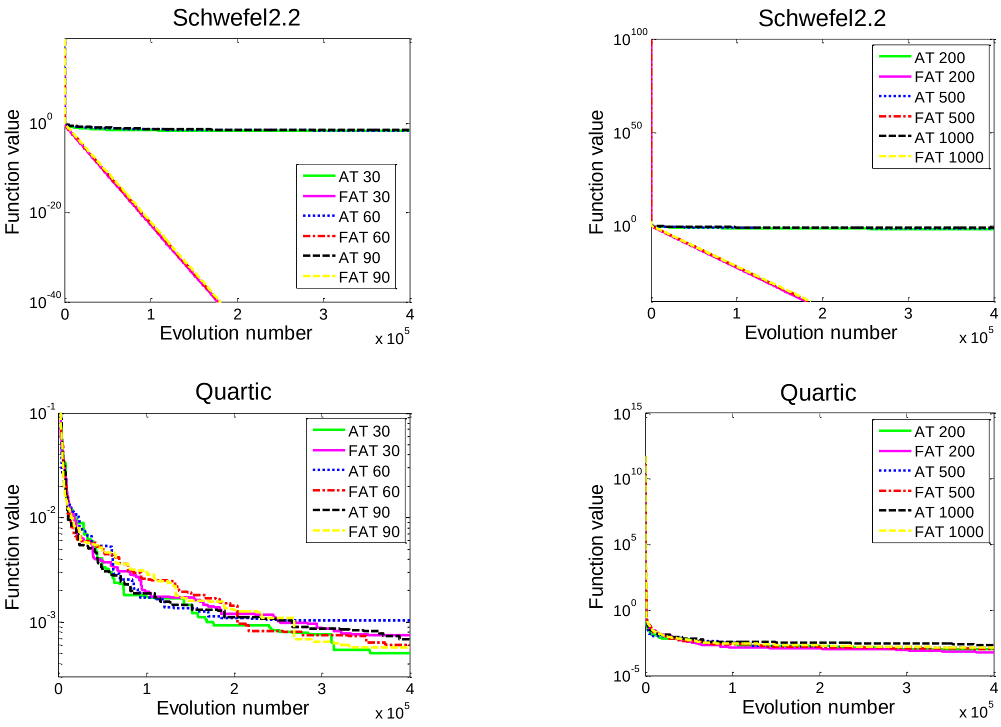


Fig. 4. Convergence curves of AT and FAT on the ten high dimensional test functions.

4.2. Comparison between FAT and PSO, APSO, DE, SaDE, ABC, MABC

Besides AT, the results of FAT are compared with other well-known heuristic algorithms (PSO, APSO, DE, SaDE, ABC and MABC) to fully evaluate the performance of FAT. The high dimensional problems listed in Tab. A2 of the appendix are applied, and the dimensions of these test functions are all taken as 1000. Table 7 illustrates the parameter values of algorithms PSO, APSO, DE, SaDE, ABC and MABC that are obtained from references (Karaboga & Basturk 2008; Koombhongse et al. 2008; Karaboga & Akay 2009; Zhang et al. 2014; Ghambari & Rahati 2018). The parameters of FAT are the same as those of Section 4.1. The results of all algorithms are obtained by thirty independent runs for all instances, and the maximum function evaluation number for all functions is set as 400,000.

Table 7. The specific parameters of other algorithms.

PSO		APSO		DE		SaDE		ABC		MABC	
Pop	50	*Pop*	50	*Pop*	50	*Pop*	50	*Pop*	*50*	*Pop*	*50*
ω	0.6	ω	0.9	*f*	0.5	*f*	0.5	*Limit*	$Ne \times D$	*Limit*	$Ne \times D$
Φ_1	1.8	Φ_1	2.0	*Cr*	0.9	*Cr*	0.3	*ns*	1	*ns*	1
Φ_2	1.8	Φ_2	2.0							*P*	*0.7*

Pop, population size; ω, inertia weight; Φ_1, Φ_2,cognitive and social components; *f*, scaling factor; *Cr*, crossover operation rate for DE; *D* , dimension of the problem; *ns*, scout number; *Limit*, maximum trial number; *P*, the selective probability; *Ne*, the number of employed bees.

Table 8 shows the experimental results of these algorithms, and Tab. 9 is a summary of the computational results of all functions. The better solutions of mean values in the Tab. 8 are mark as bolded and gray. From Tabs. 8

and 9, FAT performs better than PSO and MABC for all these test functions. Regarding problems Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Quartic, Griewank and Rastrigin, FAT acquires the better solutions compared with DE, SaDE and ABC. The results of FAT are the same as DE, SaDE and ABC on function Matyas. Compared with APSO, FAT obtains the better and similar results for nine functions (Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Quartic, Schaffer, Griewank and Rastrigin) and one problem (Matyas), respectively. In summary, FAT performs better than PSO and MABC on all these ten questions, better than DE, SaDE and ABC on eight instances and better than APSO on nine functions. Table 9 shows that the "Success rate" of FAT are 100.00%, 100.00%, 90.00%, 90.00%, 90.00% and 100.00% compared with PSO, APSO, DE, SaDE, ABC and MABC, respectively. The performance of FAT is demonstrated.

Table 8 also shows the symbol results of t test and Wilcoxon rank sum test at a significance level of 0.05. There are six pairwise comparisons, which are FAT to PSO, FAT to APSO, FAT to DE, FAT to SaDE, FAT to ABC and FAT to MABC. From Tabs. 8 and 9, it is obvious that the t test and Wilcoxon rank sum test results of FAT are the best among these algorithms. The results of FAT are significantly better than those of PSO, APSO, DE, SaDE, ABC and MABC on ten problems, nine problems (Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Quartic, Schaffer, Griewank and Rastrigin), eight examples (Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Quartic, Griewank and Rastrigin), eight questions (Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Quartic, Griewank and Rastrigin), eight functions (Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Quartic, Griewank and Rastrigin) and ten functions, respectively. In addition, t test and Wilcoxon rank sum test of FAT to APSO, FAT to DE, FAT to SaDE and FAT to ABC are not significant for function Matyas, which means the results of FAT and these algorithms are similar with each other. From Tab. 9, the "Success rate" of the t test and Wilcoxon rank sum test of FAT algorithm are also very high, which are 100.00%, 100.00%, 90.00%, 90.00%, 90.00% and 100.00% compared with PSO, APSO, DE, SaDE, ABC and MABC, respectively. Therefore, based on the results of t test and Wilcoxon rank sum test, the solutions obtained by FAT are obviously better than those of other six algorithms, and the performance of FAT is validated again.

Figure 5 shows the convergence curves of PSO, APSO, DE, SaDE, ABC, MABC and FAT. From Fig. 5, it is clear that the convergence speed of FAT is significantly better than that of the other six algorithms on functions Sphere, Rosenbrock, Dixon-Price, SumSquares, Schwefel2.2, Quartic, Griewank and Rastrigin. For functions Matyas and Schaffer, the convergence speed of FAT ranks fourth among all seven algorithms. Therefore, these figures demonstrate the performance of FAT again.

In summary, the update operators of branches in the organic matter transport process, the renewal theories of branches in the moisture feedback process, the branch territory strategy and the crowded strategy constitute the whole FAT algorithm, and the combination of these processes ensure the efficiency of FAT in dealing with different problems.

Table 8. Experimental results acquired by PSO, APSO, DE, SaDE, ABC, MABC and FAT.

	Function		PSO	APSO	DE	SaDE	ABC	MABC	FAT
f21	Sphere	Best	221.47664	54968.90051	3062837.082	3295.165889	3032248.128	21778.30493	0
		Mean	239.4501853	65351.12521	3123328.275	6195.286824	3118962.092	25948.20954	**0**
		SD	2.631168517	1867.859664	9159.913594	460.3813976	15680.98395	426.0374151	0
		Median	238.5354791	65157.84647	3122900.303	6195.286824	3134306.428	25948.20954	0
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	
f22	Rosenbrock	Best	12059.67569	68356892.53	14313357183	334733.4673	13577846357	10080084.28	994.6042921
		Mean	13152.15069	90593876.38	14749014006	549238.7175	14540928210	15767563.97	**994.9096085**
		SD	184.7279319	2993473.32	69369952.92	79041.63114	90438963.86	755393.8642	0.051108497
		Median	13276.90526	92002466.42	14732417652	549238.7175	14593356378	15767563.97	994.8731007
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	
f23	Dixon-Price	Best	6744.857674	15765633.22	3376740387	82519.64587	3418215550	3206762.473	0.998356926
		Mean	7860.158352	24485896.69	3590160818	134183.0329	3591643774	4617450.097	**0.998557738**
		SD	159.5433303	1244418.318	25412358.26	10509.61614	24069559.77	233248.7893	2.85058E-05
		Median	7916.190082	24716261.44	3610392361	134183.0329	3612102174	4617450.097	0.998555783
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	
f24	SumSquares	Best	3793.336724	575391.4051	14985936.55	15861.05584	14543770.24	89353.75645	0
		Mean	4149.492549	737624.086	15566352.08	22558.3103	15458269.49	119146.8379	**0**
		SD	60.26508549	26098.2979	55472.62417	2392.666497	86021.74682	2708.461396	0
		Median	4143.272579	710113.0631	15574204.19	22558.3103	15527780.24	119146.8379	0
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	
f25	Matyas	Best	1.09463E-16	0	0	0	0	5.26301E-05	0
		Mean	6.49662E-14	**0**	**0**	**0**	**0**	0.001813562	**0**
		SD	2.48659E-14	0	0	0	0	0.001698123	0
		Median	3.83175E-14	0	0	0	0	0.001813562	0
		t-test	+	≈	≈	≈	≈	+	
		Wilcoxon	+	≈	≈	≈	≈	+	
f26	Schwefel2.2	Best	364.3432933	774.109421	1.35733E+51	172.0988135	4.73741E+18	116.6980543	0
		Mean	6853602887	822.0757371	4.70239E+81	204.8403274	1.67066E+31	440.4123106	**0**
		SD	7094152431	13.88087198	2.7175E+81	3.983076016	1.71912E+31	273.3556939	0
		Median	432.3634677	807.9105274	3.9235E+81	209.4081786	4.85831E+24	447.557332	0
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	
f27	Quartic	Best	35766.63948	867824283.8	5.34023E+11	4287905.742	5.3439E+11	185216882.3	0.000252689

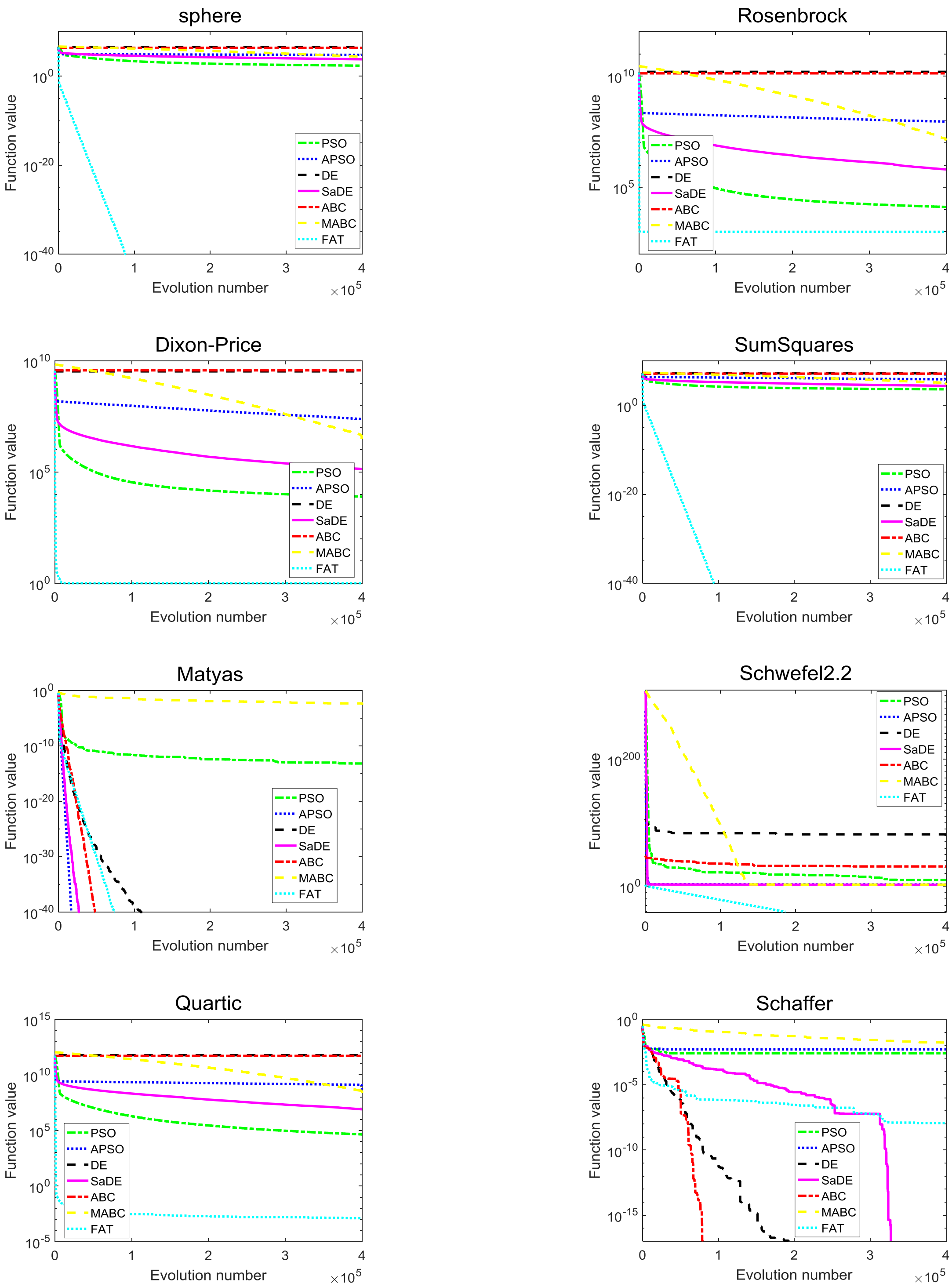
		Mean	44218.04121	1285157276	5.57238E+11	7383297.553	5.53645E+11	383151784.7	**0.001275274**
		SD	1566.411322	60796016.87	3351256833	1028463.379	3522896918	22943661.02	0.000204068
		Median	41893.53552	1292256273	5.58837E+11	7383297.553	5.5549E+11	383151784.7	0.0011872
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	
f28	Schaffer	Best	4.04876E-12	0	0	0	0	0.009716089	0
		Mean	0.002590909	0.005181819	**0**	**0**	**0**	0.012309449	1.1531E-08
		SD	0.001188601	0.001340923	0	0	0	0.002676847	9.12554E-09
		Median	1.84699E-10	0.00971591	0	0	0	0.012309449	1.30407E-11
		t-test	+	+	-	-	-	+	
		Wilcoxon	+	+	-	-	-	+	
f29	Griewank	Best	3.305865075	1688.936579	29384.22675	53.4664495	29050.1323	188.4502227	0
		Mean	3.450565886	2023.915726	30269.08757	95.54509755	30171.33431	231.0115901	**6.93889E-18**
		SD	0.031439839	38.34485886	108.4156336	9.659980223	122.9693254	5.143535122	7.16646E-18
		Median	3.414996912	2020.663977	30421.86796	95.54509755	30222.09311	231.0115901	0
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	
f30	Rastrigin	Best	594.7535566	4202.444066	17614.85921	751.8004632	17621.85114	2704.787513	0
		Mean	749.2007909	4665.894185	17815.27639	957.7268294	17785.16852	2751.14135	**0**
		SD	29.78765386	70.96685824	31.11438935	36.7572592	24.93619026	9.173110E+00	0
		Median	732.109899	4687.945899	17822.43889	957.7268294	17781.13944	2751.14135	0
		t-test	+	+	+	+	+	+	
		Wilcoxon	+	+	+	+	+	+	

Note: “≈”, “-” and “+” mean the result of FAT is equal to, worse than and better than that of AT, respectively, based on t test and Wilcoxon rank sum test at a significance level 0.05.

Table 9. Comparison between FAT and PSO, APSO, DE, SaDE, ABC, MABC based on the computational results.

	Function	PSO	APSO	DE	SaDE	ABC	MABC
Results	FAT better	10	9	8	8	8	10
	FAT worse	0	0	1	1	1	0
	FAT equal	0	1	1	1	1	0
	Success rate	100.00%	100.00%	90.00%	90.00%	90.00%	100.00%
t test	FAT better	10	9	8	8	8	10
	FAT worse	0	0	1	1	1	0
	FAT equal	0	1	1	1	1	0
	Success rate	100.00%	100.00%	90.00%	90.00%	90.00%	100.00%
Wilcoxon	FAT better	10	9	8	8	8	10
	FAT worse	0	0	1	1	1	0
	FAT equal	0	1	1	1	1	0
	Success rate	100.00%	100.00%	90.00%	90.00%	90.00%	100.00%

Note: “FAT equal”, “FAT worse” and “FAT better” are the number of results of FAT that are equal to, worse than and better than that of AT, respectively. “Success rate” is the ratio of the sum of the numbers of "FAT better" and "FAT equal" to the total number of the test functions.


sphere
Rosenbrock
Dixon-Price
SumSquares
Matyas
Schwefel2.2
Quartic
Schaffer
Function value
Evolution number
$\times 10^5$
PSO
APSO
DE
SaDE
ABC
MABC
FAT

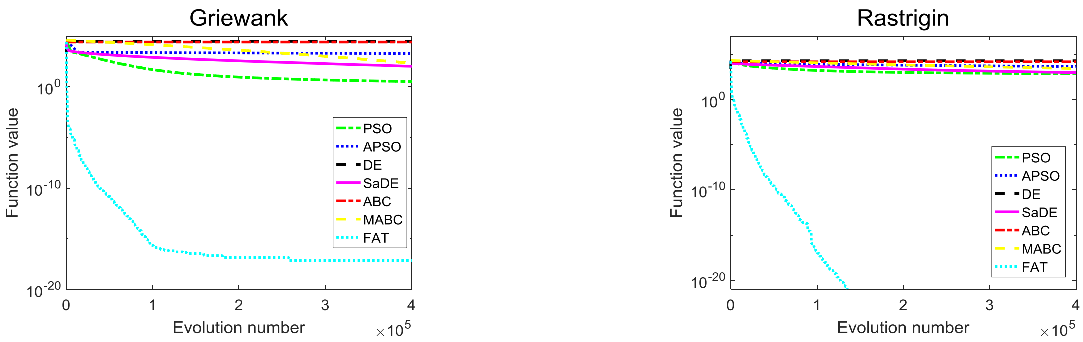


Fig. 5. Convergence curves of FAT and other six algorithms.

4.3. Sensitive analyses of FAT

The problems Sphere, Rosenbrock, Rastrigin and Griewank in Tab. A2 of the appendix are taken to study how the change of parameters r and h affects the performance of FAT. The characters of these problems are different, and the dimensions of these problems are taken as 30. The maximum evaluation number for all problems is set as 400,000. The branch population Bn, territory parameter L, crowded tolerance Tol and search parameter N are set as 50, 0.5, 1 and 10. The computational results of means, SDs and medians of 30 independent runs are illustrated.

4.3.1. Experimental analyses on parameter r

A larger r means a higher initial branch number in the feedback process. When the value of r increases, the efficiency of each round of optimization in the feedback process improves. However, with the increases of r, the update number of branch population in the feedback process decreases, which is not conducive to the search of the optimum solution in the feedback process of moistures. Therefore, a proper r helps to enhance the performance of FAT. In the experiments, the performance of FAT is tested on these four functions for different r (0.02, 0.1, 0.2, 0.4, 0.6, 0.8 and 1.0), and parameter h is taken as 20. The computational results are presented in Tab. 10. From Tab. 10, it is obvious that the performance of FAT on these four problems increases first and then decreases with the increase of r.

For Sphere, when the range is $0.1 \leq r \leq 0.8$, FAT always hits the optimum results. Regarding Griewank, FAT finds the optimum solutions within the interval of $0.1 \leq r \leq 0.4$. In addition, when the value of r takes 0.6 or 0.8, the results of FAT are very close to the optimum solution. On function Rastrigin, FAT obtains the optimum solutions with $r = 0.1$ and $r = 0.2$, and the mean values, SDs and median values of Rastrigin are similar for the parameter regions of $0.4 \leq r \leq 0.8$. Regarding Rosenbrock, FAT hits the best result with $r = 0.8$, and the results are similar within the parameter regions of $0.1 \leq r \leq 0.8$. Furthermore, the worst results of these four functions are achieved with $r = 1$. In addition, when $r = 1$, the results of FAT are significantly worse than that of $r = 0.8$ for all problems.

Table 10. The effects of parameter r on the performance of FAT.

r	Sphere			Griewank		
	Mean	SD	Median	Mean	SD	Median
0.02	2.17873E-17	6.38319E-18	1.32246E-17	2.93932E-14	8.14343E-15	2.12608E-14
0.1	0	0	0	0	0	0
0.2	0	0	0	0	0	0
0.4	0	0	0	0	0	0
0.6	0	0	0	6.93889E-18	7.16646E-18	0
0.8	0	0	0	1.38778E-17	9.79125E-18	0
1	7.51434E-15	3.5445E-15	3.83384E-15	3.42837E-13	1.26297E-13	1.11022E-13
r	Rastrigin			Rosenbrock		
	Mean	SD	Median	Mean	SD	Median
0.02	1.07692E-14	1.36541E-15	9.76996E-15	28.42564693	0.004345012	28.42809042
0.1	0	0	0	27.3097483	0.031495643	27.29439006
0.2	0	0	0	27.38142298	0.024850736	27.37586468
0.4	1.11022E-16	1.14663E-16	0	27.40619681	0.037701584	27.4127171
0.6	2.22045E-16	1.5666E-16	0	27.21202948	0.023803568	27.22741195
0.8	4.44089E-16	3.55271E-16	0	27.22774403	0.022537626	27.21656686
1	2.33147E-14	3.07845E-15	2.22045E-14	28.44895803	0.002592099	28.4444981

4.3.2. Experimental analyses on parameter h

The same four benchmark problems used in the above section are applied to evaluate the performance of FAT with different h (5, 20, 50, 200, 500, 1500 and 8000), and parameter r is set as 0.2. A large h means that the proportion of the transfer process of organic matter increases in the entire optimization process, which is in favor of the search of the optimum solution in the delivery process of organic matter. However, the increase of h reduces the contribution of the feedback process to FAT algorithm. If h takes a small value, the influence of the feedback process on FAT is large, and the effect of the organic matter transfer process on the performance of FAT decreases. Therefore, a reasonable h is crucial to control the performance of FAT. The computational results of mean, SDs and median values of FAT with 30 independent runs are given in Tab. 11.

On functions Sphere, Griewank and Rastrigin, the results of FAT tend to deteriorate as h increases. However, the condition of function Rosenbrock is different, and the results of FAT get better first and then get worse as h increases. FAT achieves the optimum results when the value of h is between 5 and 50 for functions Sphere, Griewank and Rastrigin, and the best solution is achieved with h = 50 for function Rosenbrock. Regarding the mean values, the SDs and the median values, the worst results of Sphere, Griewank, Rastrigin and Rosenbrock are achieved with h = 8000.

Table 11. The effects of parameter h on the performance of FAT.

h	Sphere	Griewank

	Mean	SD	Median	Mean	SD	Median
5	0	0	0	0	0	0
20	0	0	0	0	0	0
50	0	0	0	0	0	0
200	4.41524E-33	2.35006E-33	1.27599E-34	2.08167E-17	1.15556E-17	0
500	1.46958E-27	9.65205E-28	5.3063E-28	3.60822E-16	1.06489E-16	2.22045E-16
1500	1.21315E-26	3.28045E-27	6.29879E-27	5.41234E-16	6.77347E-17	4.996E-16
8000	1.39867E-22	6.0219E-23	3.30166E-23	6.245E-16	1.03555E-16	5.55112E-16
h	Rastrigin			Rosenbrock		
	Mean	SD	Median	Mean	SD	Median
5	0	0	0	28.0921001	0.054595542	28.12205391
20	0	0	0	27.32005766	0.020920886	27.33311413
50	0	0	0	27.0038504937	0.011920128	27.00272575
200	1.11022E-16	1.14663E-16	0	27.28362584	0.006668224	27.28372431
500	2.33147E-15	7.80497E-16	1.77636E-15	27.66098549	0.011134545	27.65862507
1500	5.995204E-15	1.19456E-15	5.32907E-15	28.1622884	0.00943242	28.15683474
8000	5.995204E-15	1.29592E-15	3.55271E-15	28.36322687	0.002455592	28.36539666

5 Conclusion

In this work, the improved version of artificial tree (AT) algorithm named the feedback artificial tree (FAT) algorithm is proposed. Differing from the standard AT algorithm, the entire material exchange process in the tree growth process is considered simultaneously, which means that both of the transfer of organic matters and the feedback of moistures are taken into account. Meanwhile, with the moisture feedback mechanism, two new branch update operators that are the self-propagating operator and the dispersive propagation operator are proposed. Some typical test functions are used to assess the performance of FAT, and the results of FAT algorithm are compared with AT algorithm first and then with other well-known algorithms. The results of these experiments show that FAT performs better than AT, and FAT also has a competitive advantage compared with other algorithms. Finally, sensitivity analyses are performed to assess the effects of specific parameters on the performance of FAT. The computational results presented in this work have clearly demonstrated that the proposed FAT has a great potential to solve a wide range of optimization problems efficiently.

Compliance with Ethical Standards

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix A

Table A1. Twenty low dimensional problems.

No.	Function	D	Interval	Min	Formulation
f1	Michalewicz2	2	$[0,\pi]$	Fmin=-1.8013	$f(\mathbf{x})=-\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m},\ m=10$
f2	Michalewicz5	5	$[0,\pi]$	Fmin=-4.6877	$f(\mathbf{x})=-\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m},\ m=10$
f3	Michalewicz10	10	$[0,\pi]$	Fmin=-9.6602	$f(\mathbf{x})=-\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m},\ m=10$
f4	Langerman2	2	[0,10]	Fmin=-1.08	$f(\mathbf{x})=-\sum_{i=1}^{m}c_i(\exp(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2)\cos$ $(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2))$
f5	Langerman5	5	[0,10]	Fmin=-1.5	$f(\mathbf{x})=-\sum_{i=1}^{m}c_i(\exp(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2)\cos$ $(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2))$
f6	Langerman10	10	[0,10]	Fmin=-1.4	$f(\mathbf{x})=-\sum_{i=1}^{m}c_i(\exp(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2)\cos$ $(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2))$
f7	Hartman3	3	[0,1]	Fmin=-3.86	$f(\mathbf{x})=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2]$
f8	Hartman6	6	[0,1]	Fmin=-3.32	$f(\mathbf{x})=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2]$
f9	Shekel5	4	[0,10]	Fmin=-10.15	$f(\mathbf{x})=-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
f10	Shekel7	4	[0,10]	Fmin=-10.4	$f(\mathbf{x})=-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
f11	Shekel10	4	[0,10]	Fmin=-10.53	$f(\mathbf{x})=-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
f12	Kowalik	4	[-5,5]	Fmin=0.00031	$f(\mathbf{x})=\sum_{i=1}^{11}(a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4})^2$
f13	Foxholes	2	[-65.536,65.536]	Fmin=0.998	$f(\mathbf{x})=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$
f14	Ackley	30	[-32,32]	Fmin=0	$f(\mathbf{x})=20+e-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))$
f15	SixHumpCamelBack	2	[-5,5]	Fmin=0	$f(\mathbf{x})=4x_1^2+2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$
f16	Penalized	30	[-50,50]	Fmin=0	$f(\mathbf{x})=\frac{\pi}{n}\{10\sin^2(\pi y_1)+(y_n-1)^2+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$
f17	Penalized2	30	[-50,50]	Fmin=0	$f(\mathbf{x})=0.1\{\sin^2(\pi x_1)+(x_n-1)^2[1+\sin^2(2\pi x_n)]+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$

f18	FletcherPowell2	2	$[-\pi,\pi]$	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j)$ $B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$
f19	FletcherPowell5	5	$[-\pi,\pi]$	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j)$ $B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$
f20	FletcherPowell10	10	$[-\pi,\pi]$	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j)$ $B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$

From Tab. A1, parameters *a*, *c*, *b*, *p* and *α* in problems Langerman2, Langerman5, Langerman10, Hartman3, Hartman6, Shekel5, Shekel7, Shekel10, Kowalik, FoxHoles, FletcherPowell2, FletcherPowell5 and FletcherPowell10 are from Karaboga and Akay (2009).

Table A2. Ten high dimensional benchmark functions.

No.	Function	Interval	Min	Formulation
f21	Sphere	[-100,100]	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{D}x_i^2$
f22	Rosenbrock	[-30,30]	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$
f23	Dixon-Price	[-10,10]	Fmin=0	$f(\mathbf{x})=(x_1-1)^2+\sum_{i=2}^{D}i(2x_i^2-x_{i-1})^2$
f24	SumSquares	[-10,10]	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{D}ix_i^2$
f25	Matyas	[-10,10]	Fmin=0	$f(\mathbf{x})=0.26(x_1^2+x_2^2)-0.48x_1x_2$
f26	Schwefel2.2	[-10,10]	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$
f27	Quartic	[-1.28,1.28]	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{D}ix_i^4+random[0,1)$
f28	Schaffer	[-100,100]	Fmin=0	$f(\mathbf{x})=0.5+\frac{\sin^2(\sqrt{\sum_{i=1}^{D}x_i^2})-0.5}{(1+0.001(\sum_{i=1}^{D}x_i^2))^2}$
f29	Griewank	[-600,600]	Fmin=0	$f(\mathbf{x})=\frac{1}{4000}(\sum_{i=1}^{D}x_i^2)-(\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}))+1$
f30	Rastrigin	[-5.12,5.12]	Fmin=0	$f(\mathbf{x})=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$

References

Chen K, Zhou F, Yin L et al. (2018) A hybrid particle swarm optimizer with sine cosine acceleration coefficients. Information Sciences 422**,** 218-241.

Coelho LdS, Ayala HVH, Freire RZ (2013) Population's variance-based Adaptive Differential Evolution for real parameter optimization. In: 2013 IEEE Congress on Evolutionary Computation. pp. 1672-1677.

Derrac J, García S, Molina D et al. (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1**,** 3-18.

Dorigo M, Caro GD (1999) Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat No 99TH8406). pp. 1470-1477 Vol. 1472.

Duan L, Jiang H, Cheng A et al. (2019a) Multi-objective reliability-based design optimization for the VRB-VCS FLB under front-impact collision. Structural and Multidisciplinary Optimization 59**,** 1835-1851.

Duan L, Jiang H, Geng G et al. (2019b) Parametric modeling and multiobjective crashworthiness design optimization of a new front longitudinal beam. Structural and Multidisciplinary Optimization 59**,** 1789-1812.

Fister Jr I, Yang X-S, Fister I et al. (2013) A brief review of nature-inspired algorithms for optimization. arXiv preprint

arXiv:13074186.
Gao WF, Liu SY (2012) A modified artificial bee colony algorithm. Computers & Operations Research 39, 687-697.
Ghambari S, Rahati A (2018) An improved artificial bee colony algorithm and its application to reliability optimization problems. Applied Soft Computing 62, 736-767.
Glibovets NN, Gulayeva NM (2013) A Review of Niching Genetic Algorithms for Multimodal Function Optimization. Cybernetics and Systems Analysis 49, 815-820.
Guo H, Li Y, Li J et al. (2014) Differential evolution improved with self-adaptive control parameters based on simulated annealing. Swarm and Evolutionary Computation 19, 52-67.
Hamzaçebi C (2008) Improving genetic algorithms' performance by local search for continuous function optimization. Applied Mathematics & Computation 196, 309-317.
Holland JH (1992) Genetic Algorithms. Scientific American 267, 66-73.
Huang H, Lv L, Ye S et al. (2019) Particle swarm optimization with convergence speed controller for large-scale numerical optimization. Soft Computing 23, 4421-4437.
Karaboga D, Akay B (2009) A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation 214, 108-132.
Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 39, 459-471.
Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing 8, 687-697.
Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International conference on systems, man, and cybernetics Computational cybernetics and simulation. IEEE. pp. 4104-4108.
Koombhongse S, Eby R, Jones S et al. (2008) A colony optimization for continuous domains. European Journal of Operational Research 185, 1155-1173.
Li MW, Han DF, Wang WL (2015) Vessel traffic flow forecasting by RSVR with chaotic cloud simulated annealing genetic algorithm and KPCA. Neurocomputing 157, 243-255.
Li QQ, He ZC, Li E (2019a) Dissipative multi-resonator acoustic metamaterials for impact force mitigation and collision energy absorption. Acta Mechanica 230, 2905-2935.
Li QQ, He ZC, Li E et al. (2018) Design and optimization of three-resonator locally resonant metamaterial for impact force mitigation. Smart Materials and Structures 27, 095015.
Li QQ, He ZC, Li E et al. (2019b) Improved impact responses of a honeycomb sandwich panel structure with internal resonators. Engineering Optimization, 1-22.
Li QQ, Song K, He ZC et al. (2017) The artificial tree (AT) algorithm. Engineering Applications of Artificial Intelligence 65, 99-110.
Li X, Qian J (2003) Studies on artificial fish swarm optimization algorithm based on decomposition and coordination techniques. Journal of circuits and systems 1, 1-6.
Lin Q, Hu B, Tang Y et al. (2017) A local search enhanced differential evolutionary algorithm for sparse recovery. Applied Soft Computing 57, 144-163.
Malik M, Ahsan F, Mohsin S (2016) Adaptive image denoising using cuckoo algorithm. Soft Computing 20, 925-938.
Ming N, Can W, Zhao X (2014) A review on applications of heuristic optimization algorithms for optimal power flow in modern power systems. Journal of Modern Power Systems and Clean Energy 2, 289-297.
Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: A Gravitational Search Algorithm. Information Sciences 179, 2232-2248.
Simon D (2016) Biogeography-based optimization. In: International Conference on Mobile Computing and NETWORKING. pp. 465-466.

Singh A, Deep K (2019) Artificial Bee Colony algorithm with improved search mechanism. Soft Computing, 1-24.

Storn R, Price K (1997) Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341-359.

Xu H, Zhang L, Li Q (2019) A novel inverse procedure for load identification based on improved artificial tree algorithm. Engineering with Computers.

Yang Q, Chen WN, Yu Z et al. (2017) Adaptive Multimodal Continuous Ant Colony Optimization. IEEE Transactions on Evolutionary Computation 21, 191-205.

Zandevakili H, Rashedi E, Mahani A (2019) Gravitational search algorithm with both attractive and repulsive forces. Soft Computing 23, 783-825.

Zhang Z, Jiang Y, Zhang S et al. (2014) An adaptive particle swarm optimization algorithm for reservoir operation optimization. Applied Soft Computing Journal 18, 167-177.

Zhong F, Li H, Zhong S (2016a) A modified ABC algorithm based on improved-global-best-guided approach and adaptive-limit strategy for global optimization. Applied Soft Computing 46, 469-486.

Zhong Y, Zhu Z, Ong YS (2016b) Soft computing in remote sensing image processing. Soft Computing 20, 4629-4630.

Zhu W, Tang Y, Fang J-a et al. (2013) Adaptive population tuning scheme for differential evolution. Information Sciences 223, 164-191.