Expert Systems with Applications 41 (2014) 6676–6687

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Forest Optimization Algorithm

Manizheh Ghaemi [a,*], Mohammad-Reza Feizi-Derakhshi [b]

[a] *Department of Computer Science, University of Tabriz, Tabriz, Iran*
[b] *Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran*

ARTICLE INFO

Article history:
Available online 17 May 2014

Keywords:
Forest Optimization Algorithm (FOA)
Evolutionary algorithms
Nonlinear optimization
Data mining
Feature weighting

ABSTRACT

In this article, a new evolutionary algorithm, Forest Optimization Algorithm (FOA), suitable for continuous nonlinear optimization problems has been proposed. It is inspired by few trees in the forests which can survive for several decades, while other trees could live for a limited period. In FOA, seeding procedure of the trees is simulated so that, some seeds fall just under the trees, while others are distributed in wide areas by natural procedures and the animals that feed on the seeds or fruits. Application of the proposed algorithm on some benchmark functions demonstrated its good capability in comparison with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Also we tested the performance of FOA on feature weighting as a real optimization problem and the results of the experiments showed the good performance of FOA in some data sets from the UCI repository.

1. Introduction

Optimization is the process of making something better (Rajabioun, 2011). In other words, optimization is the process of adjusting the inputs to find the minimum or maximum output or result. The input consists of variables and the output is the cost or fitness. Some of the optimization methods are inspired by the nature; for example Genetic Algorithm (GA) (Sivanandam & Deepa, 2007), Cuckoo Optimization Algorithm (COA) (Rajabioun, 2011) and Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995) are all nature inspired algorithms. Forest Optimization Algorithm (FOA), which is inspired by the nature's process in the forests, is another attempt to solve nonlinear optimization problems.

It has been for million years that trees are governing in the forests and different kinds of trees use different ways to survive and to continue their generations. But considering the rule of the nature, after some years most of the trees deem to die and aging is inevitable. This time, the flow of the nature replaces the old trees with the new ones and rarely some trees succeed to live for several decades. The distinguished trees, which could survive for a long time, are often the ones that are in suitable geographical habitats and also they have the best growing conditions. In other words, plant species immediately disperse their seeds to place the propagules in safe sites where they can grow and survive (Green, 1983). The procedure of the Forest Optimization Algorithm (FOA) attempts to find these distinguished trees (near-optimal solutions) in the forest with the help of natural procedures like seed dispersal.

In some forests like tropical dry forests, all species are either clumped or randomly dispersed (Hubbell, 1979); where the mode of dispersal affects the clumping of the trees. Different natural procedures distribute the seeds of all trees in the entire forest; these procedures are known as seed dispersal. Seed dispersal deals with the departure of diaspora, where diaspora is a unit of a plant like seed or fruit (Howe & Judith, 1982). Mostly joint procedure of dispersal and establishment is considered and not just movement of seeds to places where they can not establish (Cain, Milligan, & Strand, 2000). Two kinds of seed dispersal methods exist in nature: local seed dispersal and long-distance seed dispersal (Cain et al., 2000; Murrell, 2009).

In the nature when the seeding process begins, some seeds fall just near the trees and begin to sprout. This procedure is named as local seed dispersal (Cain et al., 2000; Murrell, 2009) and we will refer to this process as "Local seeding". But most of the times, the interference of animals that feed on the seeds and also other natural processes such as the flow of the water and wind carry the seeds to faraway places. Also some trees give elegant wings and plumes to their seeds to be transported to far places (Howe & Judith, 1982). This way, the territory of various trees is expanded in the entire forest. This procedure is named as long-distance seed dispersal (Cain et al., 2000); which we will name it as "Global seeding".

There is a hypothesis named as "Escape Hypothesis" and it implies disproportionate success for seeds that move far from the

* Corresponding author. Tel.: +98 9363206231.
E-mail addresses: m_ghaemi89@ms.tabrizu.ac.ir (M. Ghaemi), mfeizi@tabrizu.ac.ir (M.-R. Feizi-Derakhshi).

http://dx.doi.org/10.1016/j.eswa.2014.05.009

vicinity of their parent, while comparing those seeds that have fallen nearby (Howe & Judith, 1982). Also, “Colonization Hypothesis” assumes that parents can use the advantage of uncompetitive environment with the help of long-distance dispersal (Howe & Judith, 1982). Many studies have proved these hypothesis so, global seeding in forests gains a great importance. In addition to these hypothesis, Cain et al. expressed the importance of long-distance dispersal in comparison with local seed dispersal (Cain et al., 2000).

The seeds after being fall on the land – as the result of local or global seeding – begin to sprout and soon they turn into young trees. But not every seed gets the chance to grow up and become a tree in the forest. This may happen because of many reasons. Some trees themselves show unusual behavior; for example most of the trees habitually produce a large amount of empty and dead seeds in addition to live seed (Gosling, 2007). This strategy in addition to save energy, will encourage predators from wasting time and energy in sorting through empty seeds. Another phenomenon in the behavior of some trees is ‘Malingering’ seeds (Gosling, 2007) or ‘Dormancy’; it means that some seeds need absolute requirements to be ‘pretreated’ before they will germinate at all. As the result it is a realistic thought that all the trees of the future are in a few seeds of today.

Das, Battles, Stephenson, and van Mantgem (2011) listed 3 main factors that affects the death of trees: biotic (evidence of tree-killing pathogens or insects), suppression and mechanical (evidence of crushing, snapping, or uprooting). Among the listed factors for death of a tree, suppression which encompasses mostly density-dependant mortality or competition, plays a significant role. As the result, one reason is obviously the rule of “survival of the fittest” or competition (Das et al., 2011).

In the forests when the seeds fall on the land just near to the tree itself, some of them began to sprout and they turn into seedlings. But even there is a competition among neighboring seedlings to use the sunlight and other growing essentials. It is seen in nature that as local density increases, mortality due to competition increases too (LepS & Kindlmann, 1987; Murrell, 2009) and the competition for limited resources will remove the nearby neighbors. In other words, the primary reason of mortality is often considered to be competition and that competition for resources is the only non-climatic, non-catastrophic, non-random mechanism that affects the likelihood of mortality (Das et al., 2011). Also Green (1983) studied the relation between dispersal and safe side abundance. He reported that, although near the parent tree there exist an overabundant of propagules, but there is too few safe sites to accommodate. So, the winners of this competition are those seedlings that have begun the competition sooner than the others. Others become losers because of the lack of sunlight and also other life essentials at the beginning of the competition. As a result, few seeds that have fallen near to each other have the chance to survive and just the fittest one can survive. This competition for survival takes place at very early age of the seedlings and when one of them succeeds to be the winner, no other trees can grow just under that tree in later years.

Population of the trees in forests can exchange individuals or colonize empty and suitable habitats by long-distance dispersal. The unusual events that moves seeds long distances are of critical importance because most seeds move short distances (zero to a few tens of meters) (Cain et al., 2000). Although most of the seeds that are carried to better places have a good chance of survival, but some limitations should be considered on the number of whole seeds that can grow even for a few years.

In this paper, we have introduced a new evolutionary optimization algorithm, which is inspired by the procedure of a few trees in the forests that could survive for many years. It seems that we can use the procedures that the nature has shown us for millions of years to optimize our solutions in some optimization problems. To do so, we have simulated the local seed dispersal and long-distance seeds dispersal of the trees (Howe & Judith, 1982) as local seeding and global seeding in FOA respectively. Local seeding helps the trees to locally spread their seeds in better growing conditions. Also, animals or other natural procedures distribute the seeds by global seeding in wide areas to get rid of local optima.

In Section 2, Forest Optimization Algorithm (FOA) is introduced and its stages are studied in details. The application of the proposed algorithm on different benchmark functions is tested in Section 3. Section 4 is devoted to feature weighting as a real optimization problem. In Section 5 we have provided a comparison between FOA and other search methods. Finally, the conclusions and the future works are presented in Section 6.

2. The proposed Forest Optimization Algorithm

The proposed Forest Optimization Algorithm (FOA) involves three main stages: 1- local seeding of the trees 2- population limiting 3- global seeding of the trees. Fig. 1 shows the flowchart of the proposed algorithm. The overall procedure of FOA is as the following and later in Sections 2.1–2.6 we will discuss the algorithm in more details.

Like other evolutionary algorithms, FOA starts with the initial population of trees, so that each tree represents a potential solution of the problem. A tree, in addition to the values of the variables, has a part that represents the age of the related tree. The age of a tree is set to ‘0’. After initialization of the trees, the local seeding operator will generate new young trees (or seeds in fact) from the trees with age 0 and add the new trees to the forest. Then, all the trees, except new generated ones, get old and their age increases by ‘1’. Next, there is a control on the population of

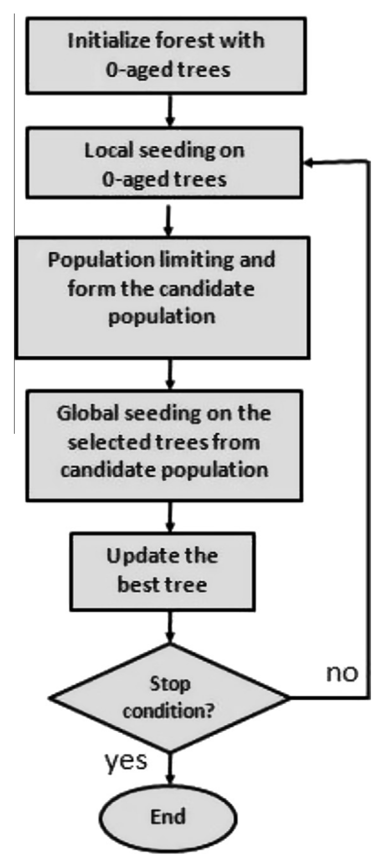


Fig. 1. Flowchart of FOA.

the trees in the forest and some trees will be omitted from the forest and they will form the candidate population for global seeding stage.

In the global seeding stage a percentage of the candidate population is chosen to move far in the forest. Global seeding stage adds some new potential solutions to the forest in order to get rid of local optimums. Now, the trees of the forest are ranked according to their fitness value and the tree with the biggest fitness value is chosen as the best tree and its age is set to 0 in order to avoid the aging and afterward removing the best tree from the forest (because local seeding stage increases the age of all trees including the age of best tree). These stages will continue until the termination criterion is met. In the following, the stages of the proposed algorithm are studied with more details.

2.1. Initialize trees

In Forest Optimization Algorithm (FOA), the potential solution of each problem has been considered as a tree. Each tree shows the values of the variables. In addition to the variables, each tree has a part related to the *"Age"* of that tree. The *"Age"* of each tree is set to '0' for each newly generated tree – as the result of local seeding or global seeding. After each local seeding stage the age of the trees, except the new generated trees in local seeding stage, increases by '1'. This increase in the age of the trees is used later as a controlling mechanism in the number of trees in the forest. Fig. 2 shows a tree for an N_{var}-dimensional problem domain, where v_is are the values of the variables and the *"Age"* part shows the age of the related tree.

A tree can also be considered as an array of length $1 \times (N_{var} + 1)$ as in Eq. (1), where N_{var} is the dimension of the problem and *"Age"* represents the age of the related tree.

$$Tree = [Age, v_1, v_2, \ldots, v_{N_{var}}] \tag{1}$$

The maximum allowed age of a tree is a predefined parameter and is named as *"life time"* parameter. *"life time"* parameter should be determined at the start of the algorithm. When a tree's *"Age"* reaches to the *"life time"* parameter, that tree is omitted from the forest and is added to the candidate population. If we choose a big number for this parameter, each iteration of the algorithm just increases the age of the trees and the forest will be full of old trees, which do not take part in the local seeding stage. Otherwise, if we choose a very small value for this parameter, the trees will get old very soon and they will be omitted at the beginning of the competition. Therefore, this parameter should provide a good chance of local search.

We have done experiments on the optimal value of *"life time"* parameter and the related results are presented later in Section 3.2 as Table 3.

2.2. Local seeding of the trees

In the nature when seeding procedure of the trees begins, some seeds fall just near the trees and after some time they turn into young trees. Now, the competition between near trees starts and those trees that have better growing conditions like enough sunlight and better location, become the winners of this competition to survive. Local seeding of the trees attempts to simulate this

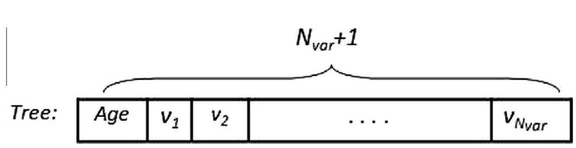


Fig. 2. A solution representation of FOA.

procedure of the nature. This operator is performed on the trees with age 0 and adds some neighbors of each tree to the forest. Two iterations of this operator on one tree are illustrated as Fig. 3. The written numbers inside the trees of Fig. 3 show the related tree's *"Age"* value; which is considered to be zero for each newly generated tree. After the local seeding is executed on the trees with age 0, the age of all trees, except the new generated trees in this stage, is increased by 1.

Increasing the age of the trees acts like a controlling mechanism on the number of the trees of the forest and effects in this way: if a tree is promising, the procedure of the algorithm resets the age of that tree to '0' and as the result, it will be possible to add neighbors of the good tree to the forest through performing local seeding stage. Otherwise, non-promising trees get old with each iteration of the algorithm and eventually die after some iterations.

The number of the seeds that fall on the land near the trees and then turn into trees as neighbors is considered as a parameter of this algorithm and is named as "Local Seeding Changes" or *"LSC"*. The value of this parameter is 3 in Fig. 3. As the result, performing local seeding operator on one tree with age 0 will produce 3 new trees. This parameter should be determined according to the dimension of the problem domain. We have done an experiment to find the optimal value of *"LSC"* parameter and the results are reported on Section 3.2 as Table 4.

At first iteration of the algorithm that all trees have the age 0, local seeding operator is executed on all trees of the forest. So, for each zero-Aged tree of the forest, the number of *"LSC"* new trees are added to the forest. At next iterations, the number of added trees by local seeding operator decreases, because there will be trees with age bigger than 0, which do not take part in local seeding stage. Local seeding operator simulates local search for this algorithm.

Fig. 4 illustrates an example of local seeding operator for real problems in 4 dimensional continuous space and where the value of *"LSC"* is considered to be 2. r and r' are two randomly generated values in the range $[-\triangle x, \triangle x]$. $\triangle x$ is a small value and it is smaller than the related variable's upper limit. This way the search procedure is done in a limited interval and local search can be simulated.

In order to perform this operator, a variable is selected randomly. Then its value is added with a small random value $r \in [-\triangle x, \triangle x]$. This procedure is repeated for *LSC* times for each tree with age 0. Numerical example for local seeding operator is shown as Fig. 5. The value of *"LSC"* is 1 in Fig. 5 and $\triangle x$ is considered to be 1. As a result, the value of one variable will be added with a randomly generated value in the range $[-1, 1]$ like 0.4. Now the new tree with age 0 will be added to the forest.

In the case of adding values, it may be situations where the values of the variables become less or more than the related variables lower and upper limits. In order to avoid these situations, values less than variable's lower limits and values bigger than upper limits are truncated to the limits.

Local seeding operator adds many trees to the forest, so there must be a limitation on the number of the trees. This control is done by the next stage of the proposed algorithm.

2.3. Population limiting

Number of the trees in the forest must be limited to prevent infinite expansion of the forest. There are 2 parameters which limit the population of trees: *"Life time"* and *"area limit"* parameters. At first the trees whose age exceeds the "life time" parameter are removed from the forest and they will form the candidate population. Second limitation is "area limit" in which after ranking the trees according to their fitness value, if the number of the trees is bigger than the limitation of the forest, extra trees are removed from the forest and are added to the candidate population. The

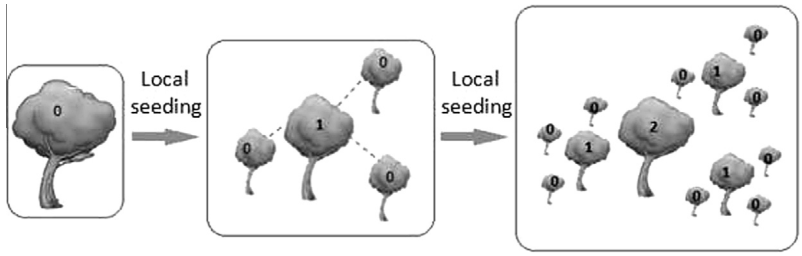


Fig. 3. An example of local seeding on one tree for 2 iterations.

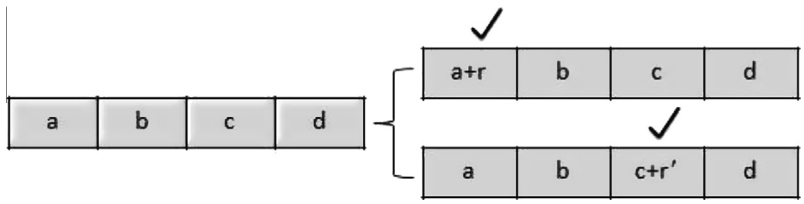


Fig. 4. An example of local seeding for continuous search space.

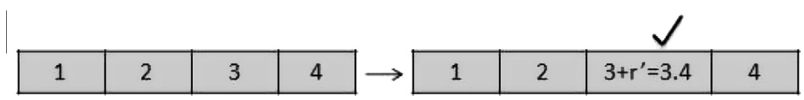


Fig. 5. A numerical example of local seeding on one tree, $LSC = 1$, $r' = 0.4. \in [-\triangle x, \triangle x] = [-1, 1]$.

limitation of the forest is another parameter and is named as *"area limit"*. In the tests of this article the value of the *"area limit"* parameter is considered to be the same as the number of the initial trees. So, after performing this stage, the number of the trees in the forest will be equal to the number of the initial trees.

After population limiting of the forest, global seeding stage is performed on some percentage of the candidate population as will be describe later.

2.4. Global seeding of the trees

There are different kinds of trees in the forests and different animals and birds feed on the seeds and fruits of these trees. So, the seeds of the trees are distributed in the entire forest and as a result, the habitat of the trees becomes wider. Also other natural processes like wind and the flow of the water helps distributing the seeds in the entire forest and guarantees the empire of the different kinds of trees in different regions. Global seeding stage attempts to simulate the distribution of the seeds of the trees in the forest.

Global seeding operator is performed on a predefined percentage of the candidate population from the previous stage. This percentage is another parameter of the algorithm that should be defined first and is named as *"transfer rate"*.

Global seeding operator works as the following: at first, the trees from the candidate population are selected. Then, some of the variables of each tree are selected randomly. This time, the value of each selected variable is exchanged with another randomly generated value in the related variable's range. This way the whole search space is considered and not a limited area. As a result, a tree with age 0 is added to the forest. This operator performs a global search in the search space. The number of the variables whose values will be changed, is another parameter of the algorithm and is named as "Global Seeding Changes" or *"GSC"*.

Fig. 6 is an example of the performing global seeding operator for one tree in continuous space. In Fig. 6, the value of *"GSC"* parameter is considered to be 2 so, two variables are selected randomly and their values are exchanged with two other randomly generated values like r and r' in the related variable's ranges.

The numerical example for global seeding operator is illustrated as Fig. 7, where the value of *"GSC"* parameter is 2 and the range of all variables is the same and it is $[-5, 5]$. As the result, the value of 2 randomly selected variables are exchanged with other values in the range $[-5, 5]$ like -0.7 and 1.5.

2.5. Updating the best so far tree

In this stage, after sorting the trees according to their fitness value, the tree with the highest fitness value is selected as the best tree. Then the age of best tree will be set to 0 in order to avoid the aging of the best tree as the result of local seeding stage. In this way, it will be possible for the best tree to locally optimize its location by local seeding operator; because as mentioned before, local seeding is performed on trees with age '0'.

2.6. Stop condition

Like other evolutionary algorithms, three stop conditions can be considered: 1- predefined number of iterations 2- observance of no change in the fitness value of the best tree for several iterations 3- reaching to the specified level of accuracy.

The main stages of FOA are shown as a pseudo code in Fig. 8. In the next part, FOA is applied to some benchmark optimization problems. As the optimal solutions are known for benchmark functions in advance so, reaching to a specified level of accuracy is considered as stop condition in the experiments of Section 3.

3. Benchmarks on Forest Optimization Algorithm

In this section, Forest Optimization Algorithm (FOA) is tested with 4 benchmark functions. All these problems are minimization problems. Because the global optimum of the benchmark functions

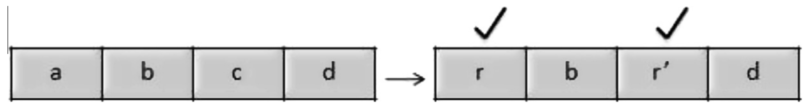


Fig. 6. An example of global seeding on one tree.

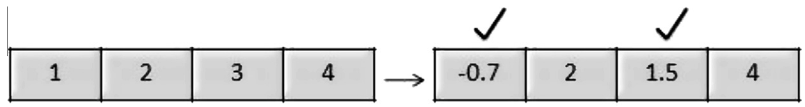


Fig. 7. A numerical example of global seeding on one tree $GSC = 2$.

```
Algorithm FOA (life time, LSC, GSC, transfer rate, area limit)

Input: life time, LSC, GSC, transfer rate, area limit
Output: near optimal solution for objective function f(x)
1. Initialize forest with random trees
   1.1 Each tree is a (D+1)-dimensional vector x,   x= (age, x1, x2, . . ., xD) for a D-dimensional
       problem
   1.2 The "age" of each tree is initially zero
2. While stop condition is not satisfied do
   2.1 Perform local seeding on trees with age 0
         • For i=1: "LSC"
             – Randomly choose a variable of the selected tree
             – add a small amount dx- dx ε [-Δx, Δx]- to the randomly selected variable
         • Increase the age of all trees by 1 except for new generated trees in this stage
   2.2 Population limiting
         • Remove the trees with age bigger than "life time" parameter and add them to
           the candidate population
         • Sort trees according to their fitness value
         • Remove the extra trees that exceed the "area limit" parameter from the end of
           forest and add them to the candidate population
   2.3 Global seeding
         • Choose "transfer rate" percent of the candidate population
         • For each selected tree
             – Choose "GSC" variables of the selected tree randomly
             – Change the value of each variable with other randomly generated value in
               the variable's range and add a new tree with age 0 to the forest
   2.4 Update the best so far tree
         • Sort trees according to their fitness value
         • Set the age of the best tree to 0
3. Return the best tree as the result
```

Fig. 8. Pseudo code of Forest Optimization Algorithm (FOA).

is known in advance so, reaching to the specified level of accuracy is considered to be the stop condition of the experiments in this section. In order to do a comparison, continuous GA with selection with elitism and uniform cross-over and also PSO algorithm are applied to the benchmark functions and the results of the FOA is compared with both GA and PSO. To well study the performance of FOA in comparison with GA and PSO, three of the functions are implemented in 5 and 10 dimensions; this will increases the complexity of the optimization task.

3.1. Test functions

The test functions that we have used to validate our experiments are summarized in Table 1. F2, F3 and F4 are chosen from Marcin and Smutnicki (2005).

F1 is a test function with the minimum value of $f(x) = -18.5547$ in position (9.039, 8.668). F2 is Griewangk function and it has many widespread local minima regularly distributed. It has global minimum of $f(x) = 0$ at $x_i = 0$ for $i = 1, \ldots, n$. F3 is Sum of different powers function and it is a commonly used test function. Its global minimum of $f(x) = 0$ is obtainable for $x_i = 0,\ i = 1, \ldots, n$ in the range $-1 \leqslant x_i \leqslant 1$. F4 is Rastrigin function and it is a commonly used multimodal test function. Its global minimum of $f(x) = 0$ is obtainable for $x_i = 0,\ i = 1, \ldots, n$ in the range $-5.12 \leqslant x_i \leqslant 5.12$.

Table 1
Test functions adopted for our experiments.

Function	Equation	Search range
F1	$f_1(x) = x \times \sin(4x) + 1.1y \times \sin(2y)$	$0 < x,\ y < 10$
F2	$f_2(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 \leqslant x_i \leqslant 600$
F3	$f_3(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	$-1 \leqslant x_i \leqslant 1$
F4	$f_4(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$	$-5.12 \leqslant x_i \leqslant 5.12$

We will study F1 in details but for summarization, we will just list the results for other test functions in 5 and 10 dimensions.

3.2. Results of the experiments on benchmark functions

F1 is a test function and Fig. 9 shows a 3D plot of it. Initial forest with 30 trees is illustrated as Fig. 10(a). Fig. 10(a–h) show the

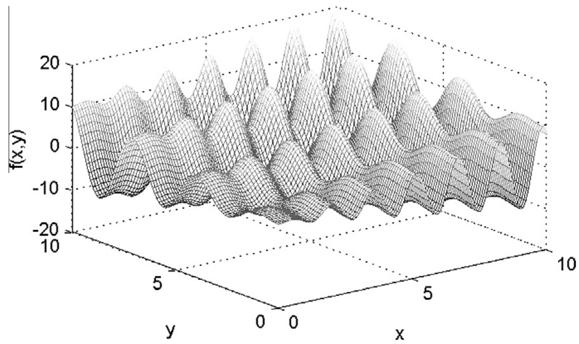


Fig. 9. 3D plot of F1.

population of the forest in iterations 1–4, 8–9, 12 and 15. As it is obvious from the Fig. 10(a–h), FOA has obtained the global minimum just in 15 iterations with level of accuracy 0.1. In iteration 12 most of trees are in global minimum and in iterations 12–15 the algorithm continues in order to just reaching to the specified level of accuracy. Finally at iteration 15 most of trees are near to the best location of the forest, which is the global minimum of the problem. This location is (9.039, 8.668) with the cost value of -18.5547. Those trees that are in different positions are due to global seeding stage at final iteration.

For better illustration the procedure of FOA, digital values of the 4 iterations on a forest with 5 trees on F1 is shown as Fig. 11. In each iteration, the best tree is shown with an arrow in front. In Fig. 11 the first and second column shows the values of the variables of F1 and the third column represents the fitness of the corresponding tree. The age of each tree is illustrated as the forth column.

As it can be seen from Fig. 11, in the first iteration of the algorithm the age of all trees is 0 so, local seeding operator will generate the neighbors of all trees. In next iterations, there are trees with age bigger than 0 in the forest; which do not take part in the local seeding stage.

In order to show the percentage of zero-aged trees on which local seeding operator will affect, we have done an experiment. To do so, we have chosen "Sum of different powers" (a unimodal function) and "Griewangk" (a multimodal function) test functions (Marcin & Smutnicki, 2005) in 5 dimensions. The results will definitely depend on the values of the parameters like the number of initial trees, *"area limit"* parameter and *"transfer rate"*. For our experiments we set the parameters as the following; initial size of forest to be 30, "area limit"=30, *LSC*=1, *GSC*=1, *"transfer rate"*=10% and *"life time"*=6; with these parameters the results are reported as Table 2. The results relate to 3 runs on each function and they show the average of the zero-aged trees in each iteration. This experiment shows that in each iteration of both test functions – regardless of being unimodal or multimodal – approximately 1/5 of the trees in the forest take part in local seeding.

While the execution of the algorithm continuous, the age of the best tree remains 0; this will give the chance for the best tree to locally optimize its location by local seeding operator and at later iterations the forest will be full of the neighbors of the best tree. Convergence to the best tree can be seen in Fig. 10(h). In Fig. 10(h) most of the trees converge to the location of the best tree. This convergence will continue until the location of the best tree changes due to global seeding stage. The *"life time"* parameter in Fig. 11 is considered to be 2; this means that if a tree is not the best tree of the forest, at worst case and if it is not removed because of limitation of the forest, it will be removed from the forest after 2 iterations.

Our experiments are done on MATLAB R2013a environment and on a system with 2.5 GHz of CPU, 4GB of RAM and on Windows7 (64 bit) operating system. Due to the fact that different initial populations of each method affect directly to the final result and the speed of algorithm, a series of test runs – here 30 runs – are done to have a mean expectance of performance for each method.

Fig. 12 depicts the fitness of best tree versus iteration number for F1 in 2 dimensions with level of accuracy 0.1. It can be seen from Fig. 12 that, FOA converges to the optima position in less than 10 iterations and later iterations continue until reaching to the specified level of accuracy. In order to test the stability of FOA, Fig. 13 depicts the stability diagram for 30 independent runs; which shows a stable behavior of FOA.

In this article we have done experiments in order to find the optimal values of *"life time"* parameter and *"LSC"* parameter. To do so, we have tested different values for *"life time"* parameter – *"life time"*$\in \{1, 2, \ldots, 8\}$ – on F1 and the results are summarized as Table 3. The results are compared in the case of number of evaluations (*num_eval*) and iterations (iter). As the small number of evaluations in experiments is desirable, we have chosen the optimal value for *"life time"* parameter according to small number of evaluations. From the results of experiments in Table 3, it seems that optimal value for *"life time"* parameter is 6; as the number of evaluations does not decrease so much after the *"life time"* reaches to 6. In the later experiments, we will set the value of *"life time"* parameter to 6.

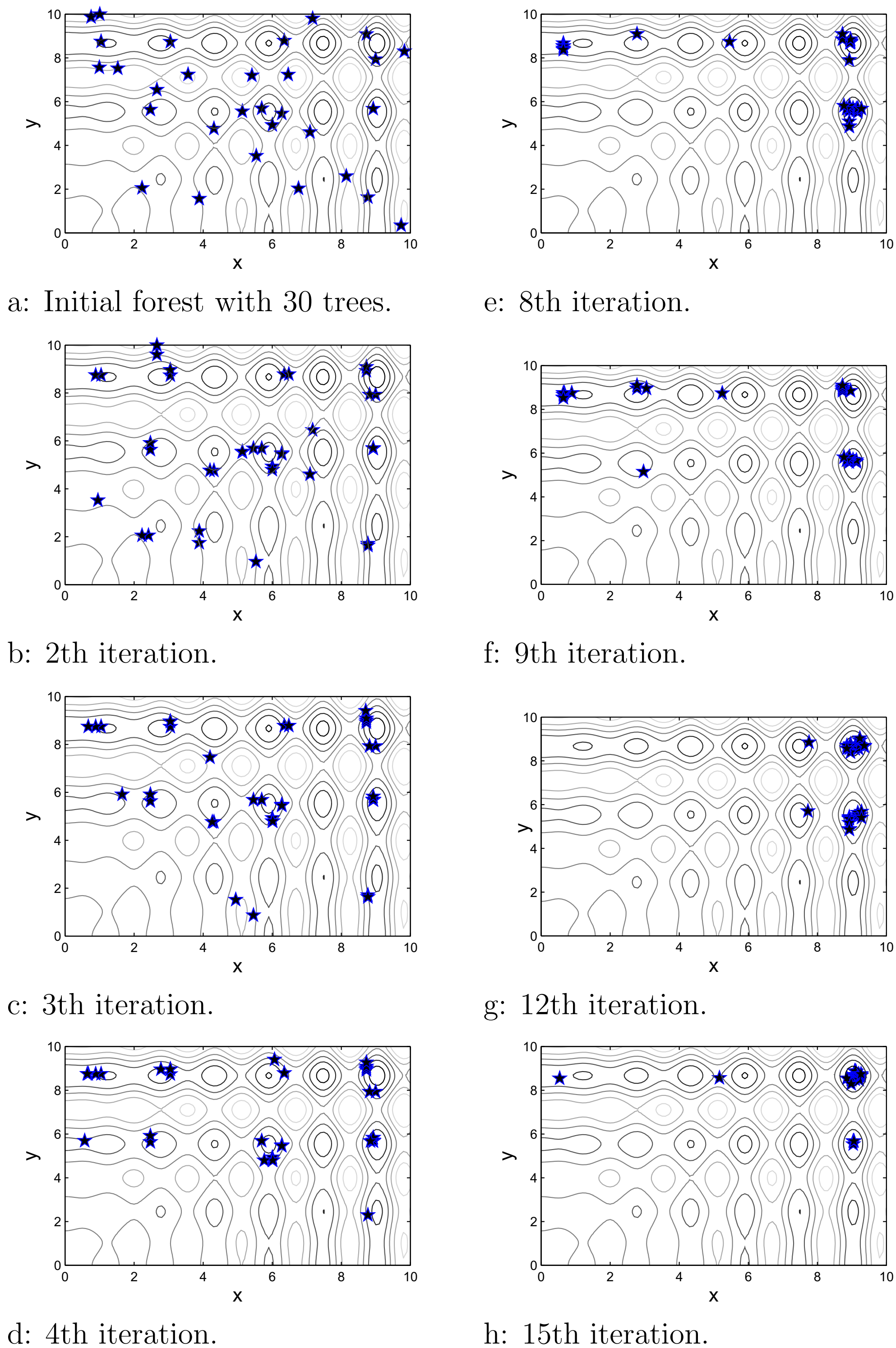
Also, in order to find the optimal value for *LSC* parameter, we performed experiments on F4 with 10 dimensions where "life time" is 6 and *GSC*=1. The results of the experiments are shown as Table 4. Again the results are compared in the case of number of evaluations (*num_eval*) and iterations (iter). According to Table 4, the value 1 for *LSC* parameter seems suitable according to small number of evaluations but, while comparing the number of iterations the value *LSC*=2 is desirable than value *LSC*=1.

As we mentioned earlier, the optimal value for this parameter depends on the dimension of the problem domain. From the experiments in this section, we will set the value of *LSC* parameter to be 2/10 of the problem dimension in our later experiments when the dimension is bigger than 5; for dimensions less than 5, the value of *LSC* will be 1.

After finding the optimal value for *"life time"* parameter and *LSC* parameter, we just list the results for the benchmark functions from Table 1. Table 5 shows the parameters for implementation of GA, PSO and FOA on benchmark functions and the results for these implementations are presented as Table 6. For each function the best performance according to the number of evaluations is highlighted in bold form. F1 is implemented in 2 dimensions but F2, F3 and F4 are implemented in 5 and 10 dimensions. Results are averaged over 30 runs with the parameters listed in Table 5. "iter" indicates the average of iteration numbers to reach to the specified level of accuracy in 30 individual runs and 'num_eval' indicates the average of the number of evaluations for each experiments over 30 runs. 'level of acc.' indicates the accuracy of the best tree found by FOA from the optimal solution of the test function. In order to avoid infinite number of runs in the case of getting stuck in the local optima, we have stopped the algorithm execution when the number of evaluations exceeds 70,000 evaluations.

The results of Table 6 show that, FOA outperforms both GA and PSO in reaching to the near optimal solution for F1, because the number of evaluations and iterations for FOA is less than the others (according to the 2D plot of F1 in Fig. 9, F1 has many local optima).

The experiments on F2 with many widespread local minima shows that PSO reaches to the near optimal solution with less number of evaluations than GA and FOA in 5 dimensions and with less number of iterations. When the dimension for F2 increases from 5 to 10, the results show a good improvement of FOA in



a: Initial forest with 30 trees.
e: 8th iteration.
b: 2th iteration.
f: 9th iteration.
c: 3th iteration.
g: 12th iteration.
d: 4th iteration.
h: 15th iteration.

Fig. 10. Location of trees on 2 dimensional test function F1 in iterations 1–4, 8–9, 12 and 15 of FOA. The best tree is found at the 15th iteration.

comparison with GA and PSO both in number of evaluations and iterations.

It can be seen from Table 6 that, FOA outperforms both GA and PSO in F3 in 5 dimensions; which is a good improvement. Similar results can be seen for F3 in 10 dimensions. F3 is a unimodal function so, the results show the good performance of FOA in unimodal functions like F3 in comparison with GA and PSO. Because the number of evaluations of FOA is less than both GA and PSO in F3 in 5 and 10 dimensions.

F4 is a highly multimodal function (Marcin & Smutnicki, 2005) and the search methods tend to get stuck in local optima. The results of Table 6 for F4 show that, FOA noticeably performs better than both GA and PSO because the number of fitness evaluations for FOA is much less than GA and PSO.

initial

x	y	f(x,y)	age
2.011	5.354	-3.675	0
1.367	5.643	-6.943	0
9.228	1.748	-7.209	0
3.036	6.472	1.376	0
3.567	2.018	1.807	0

iter. 1

x	y	f(x,y)	age
5.709	5.978	-8.041	0
9.228	1.886	-7.766	0
9.228	1.748	-7.209	1
1.367	5.643	-6.943	1
9.228	1.609	-6.679	0
1.367	5.190	-5.652	0

iter. 2

x	y	f(x,y)	age
5.709	5.857	-9.125	0
9.228	2.306	-9.066	0
5.709	5.978	-8.041	1
9.228	1.886	-7.766	1
1.367	5.570	-7.058	0
9.228	3.223	-5.966	0

iter.3

x	y	f(x,y)	age
5.709	5.664	-10.166	0
9.228	2.498	-9.180	0
5.709	5.857	-9.125	1
9.228	2.306	-9.066	1
9.228	2.114	-8.599	0
5.709	9.916	4.792	0

iter. 4

x	y	f(x,y)	age
9.228	8.313	-13.800	0
5.894	5.664	-11.785	0
9.147	2.498	-10.839	0
8.941	2.114	-10.408	0
5.709	5.664	-10.166	1
9.228	2.498	-9.180	1

Fig. 11. Digital values of 4 consequent iterations of FOA on F1.

Table 2
Percentage of zero-aged trees on 2 Test function in 5 dimensions.

Test function	Perc. of 0-aged trees (average ± standard deviation)
"Sum of different powers"	21.54% ± 7.6
"Griewangk"	25.81% ± 6.8

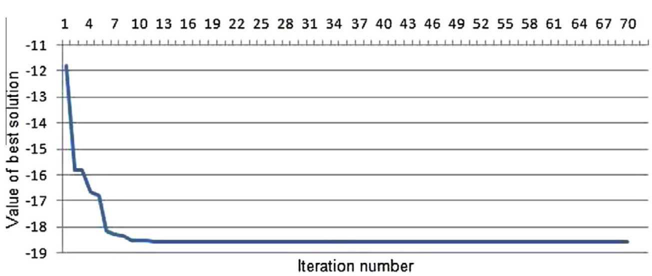


Fig. 12. Fitness of best tree versus iteration of FOA on F1 with level of accuracy 0.1.

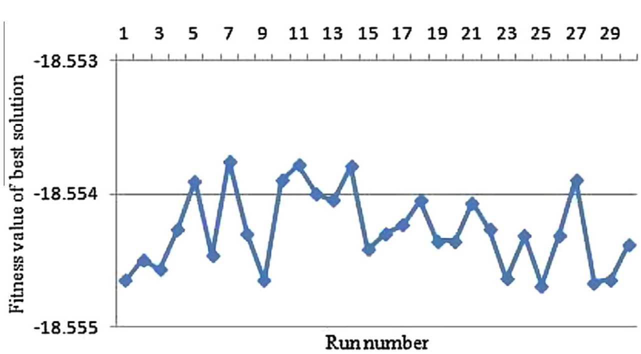


Fig. 13. Stability diagram of FOA for 30 individual runs on F1 with level of accuracy 0.1.

The results of the experiments show that local seeding of FOA helps it to have better level of accuracy; because the number of evaluations for FOA with fixed level of accuracy is less than both GA and PSO in almost all of the experiments. Also the results for F3 which is a unimodal function, illustrates that local search of FOA helps it to have better performance.

Table 3
Performance evaluation of FOA on F1 according to different values of "life time" parameter.

"life time"	1	2	3	4	5	6	7	8
num_eval	3363	1611	1568	1264	1228	1092	1038	1086
iter	35	31	35	40	40	42	41	50

Table 4
Performance evaluation of FOA on F4 in 10 dimensions according to different values of *LSC* parameter.

LSC	1	2	3	4	5	6
num_eval	13803	14063	21226	23527	31337	41643
iter	2222	820	660	431	350	278

In the following, we will study the performance of FOA on feature weighting as a real optimization problem.

4. Application of FOA on feature weighting

After that the great performance of FOA is proven in test functions, it is needed to investigate its performance in real optimization problems. To do so, feature weighting, as one of the useful methods for improving the performance of *K*-nearest neighbor (KNN) classification algorithm, is studied. In this section, we have attempt to find optimal weights for the features using FOA in order to alleviate the drawbacks of KNN.

4.1. Feature weighting

Lazy learning algorithms are machine learning algorithms that are welcome members of procrastinators anonymous (Aha David, 1998). The most famous lazy learner is the one that uses similarity function to answer queries. KNN is the bases of many lazy learning algorithms; but it has several drawbacks, such as high storage requirements and sensitivity to noise. So, many researchers have attempted to address these drawbacks. Feature selection, prototype generation/selection and feature weighting (FW) methods

Table 5
Parameters for implementation of GA, PSO and FOA.

Algorithm	Parameters
GA	# initial population = 30, pc = 0.8, pm = 0.1
PSO	# initial population = 30, c1=2, c2 = 1, inertia weight = 0.8
FOA in 2 and 5 dimensions	# initial population = 30, life time = 6, *LSC* = 1, ares limit = 30, transfer rate=10%, *GSC* = 1
FOA in 10 dimensions	# initial population = 30, life time = 6, *LSC* = 2, area limit = 30, transfer rate = 10%, *GSC* = 3

Table 6
Results for implementation of FOA, PSO and GA for test functions in different dimensions. F2, F3 and F4 are "Griewangk", "Sum of different powers" and "Rastrigin" test functions respectively. The smallest number of evaluations (num_eval) is highlighted in bold form for each function.

Func.	Dim.	Algorithm	Level of acc.	num_eval	iter
F1	2	FOA	0.001	**1092.9**	141.6
F1	2	PSO	0.001	22258.31	104
F1	2	GA	0.001	43966	1523.5
F2	5	FOA	0.5	8847.1	1402.3
F2	5	PSO	0.5	**4320.72**	71
F2	5	GA	0.5	9911.3	324.53
F3	5	FOA	0.001	**351.9**	44.9
F3	5	PSO	0.001	1756	29.26
F3	5	GA	0.001	720.3	24.8
F4	5	FOA	0.1	**3865.4**	610.73
F4	5	PSO	0.1	36722	611.01
F4	5	GA	0.1	15919	551.87
F2	10	FOA	0.5	**7510.5**	173
F2	10	PSO	0.5	12057	199.95
F2	10	GA	0.5	19566	677.57
F3	10	FOA	0.001	**924.4**	27.6
F3	10	PSO	0.001	3432	57.02
F3	10	GA	0.001	2045.7	70.7
F4	10	FOA	0.1	**14063**	820.5
F4	10	PSO	0.1	70000	1160
F4	10	GA	0.1	52475.1	1818.5

are all used to further improve the performance of KNN (Triguero, Derrac, Garcia, & Herrera, 2012).

Many studies have shown the sensitivity of KNN to the definition of distance function. Feature weighting is a continuous space search problem and attempts to influence the distance function by giving different weights to different features. In other words, the aim of FW is to reduce the sensitivity of KNN to the existence of irrelevant and redundant features by modifying its distance function with weights (Triguero et al., 2012). This change in the distance function improves the classification accuracy of KNN.

The most well known distance or dissimilarity measure for the NN (Nearest Neighbor) rule is the Euclidean Distance (Triguero et al., 2012) (Eq. (2)).

$$Euclidean\ Distance(X,Y) = \sqrt{\sum_{i=0}^{D}(x_{pi} - x_{qi})^2} \tag{2}$$

$$FW\ Dist(X,Y) = \sqrt{\sum_{i=0}^{D}W_i(x_{pi} - x_{qi})^2} \tag{3}$$

where x_p and x_q are two examples and D is the number of features. FW adds weights to Eq. (2); so, the distance function changes as Eq. (3) where $w_i's$ are the weights of the features.

Many researchers have attempt to address feature weighting problem. Tahir et al. in 2007 used Tabu search/*K*-nearest neighbor classifier to simultaneous feature selection and feature weighting (Tahir, Bouridane, & Kurugollu, 2007). They reported their method's superiority to both simple Tabu search and sequential search algorithms. Ozsen et al. applied GA to solve feature weighting problem and they evaluated the weights that GA produces with Artificial Immune System (AIS) (Ozsen & Gunes, 2009). In 2009, Tosun et al. used feature weighting in software cost estimation (Tosun, Turhan, & Bener, 2009). They assigned weights to project features by benefiting from Principal Component Analysis (PCA). Triguero et al. in 2011 tried to improve the performance of KNN using prototype selection and generation methods (Triguero, Garc, & Herrera, 2011) and later, Triguero and et al. integrated a different evolution feature weighting scheme into prototype generation (Triguero et al., 2012). In the following we will apply Forest Optimization Algorithm to solve feature weighting problem and we will investigate the performance of FOA in FW problem.

4.2. Feature weighting using Forest Optimization Algorithm (FWFOA)

In order to apply FOA in real problems, we attempt to solve feature weighting problem using FOA. So, FOA is used to learn the weights of the features before classification and then each solution is evaluated by the KNN classification algorithm for $k = 1$. The stages of FOA for feature weighting problem are adapted as the following:

4.2.1. Initialize trees

As it is mentioned before, first the forest is formed with randomly generated trees. Because feature weighting (FW) is a continues space search problem, so the weights can be any number in the range $[0,1]$. High weight for a feature shows the related feature's high importance in calculating the distance and as a result in classification task; while low weight shows otherwise. At first iteration, all the features have the relevant degree of 1; then, local and global seeding will decrease or increase the weights in order to find optimal weights.

4.2.2. Local seeding

After initializing the trees, FWFOA enters in a loop in which local seeding, population limiting, global seeding and updating the best tree guide the optimization of feature weights by generating new trees. Local seeding operator is performed on the trees with age 0. In the local seeding stage, neighbors are formed by adding or subtracting $d_x = 0.1$ from the weights of some randomly selected features. After applying this operator, we check if there have been values out of the range $[0,1]$. If a computed value is greater than 1, we truncate it to 1. Furthermore, if this value is lower than 0, we consider this feature to be irrelevant, and therefore, its weight is set to be 0. After finding the neighbors, the age of all trees except new generated ones is increased by 1.

4.2.3. Population limiting

As it is mentioned before, the trees with age bigger than "life time" parameter are removed from the forest and added to the candidate population. Also, after ranking the trees according to their fitness value, if the number of the trees is bigger than the "area limit" parameter, extra trees are removed from the forest and are added to the candidate population.

4.2.4. Global seeding

After choosing some percentage of the candidate population, for each chosen tree, "*GSC*" features are selected randomly. Then, the weight of each selected feature is replaced with an other randomly generated value in the range [0, 1]. As the result, new trees with age 0 are added to the forest.

4.2.5. Updating the best so far tree

After sorting the trees according to their fitness value, the tree with the maximum fitness value is selected as the best tree, and its age will be set to 0.

4.2.6. Stop condition

In our experiments for FWFOA, observance of no change in the fitness of best tree for 100 iterations is considered to be the stop condition of the algorithm.

4.3. Experiments

In order to evaluate FWFOA, real-world data sets are chosen from the UCI-Irvine repository (Blake, Keogh, & Merz, 1998). 1NN of the WEKA software is used to evaluate the fitness of each tree with Euclidean Distance as the distance function. Classification accuracy (CA) of the 1NN shows the fitness of each tree. CA is defined as the number of successful hits (correct classification) (Triguero et al., 2012). All experiments have been run on a machine with 2.40 GHz CPU and 4 GB of RAM.

The data sets are partitioned using ten fold cross validation (10-fcv) procedure and their values are normalized in the interval [0, 1] to equalize the influence of attributes with different range domains.

4.3.1. Data sets from UCI

All the data sets for our experiments have the features with real values. The summary of these data sets are listed as Table 7. In feature selection problem, data sets are of small scale, medium scale, or large scale if n belongs to $[0,19]$, $[20,49]$, or $[50,\infty]$, respectively (Tahir et al., 2007). So, six data sets among seven ones are small scale data sets and one of them is a medium scale data set.

4.3.2. Comparison algorithms and parameters

We have compared FWFOA with the existing methods reported by Triguero et al., 2012 on the same data sets. Triguero et al. integrated feature weighting (FW) with prototype generation methods. Table 8 shows the average and standard deviation of classification accuracy for the selected data sets reported by Triguero et al. (2012). The NN rule with $k = 1$ (1NN) has been included as a baseline limit of performance in their article. In all of the reported techniques in Table 8, Euclidean Distance is used as a similarity function (like our experiments) and those which are stochastic methods have been run three times per partition. The configuration parameters, which are common to all problems in Triguero et al. (2012), are shown as Table 9.

Table 7
Summary of the selected data sets from UCI.

Data set	# examples	# features	# class
Bupa	345	6	2
Cleveland	297	13	5
Dermatology	366	33	6
Glass	214	9	7
Iris	150	4	3
Pima	768	8	2
Heart	270	13	2

Table 9
Parameter specification for all the methods reported in Triguero et al. (2012).

Algorithm	*parameters*
SSMA	Population = 30, Evaluations = 10,000, Crossover Probability = 0.5, Mutation probability = 0.001
SSMA-DEPGFW	MAXITER = 20, PopulationSFLSDE = 40, IterationsSFLSDE = 50 PopulationDEFW = 25 , IterationsDEFW = 200, iterSFGSS = 8, iterSFHC = 20, Fl = 0.1, Fu = 0.9
IPADECS	Population = 10, iterations of Basic DE = 500, iterSFGSS = S, iterSFHC = 20, Fl = 0.1, Fu = 0.9
IPADECS-DEFW	MAXITER = 20, PopulationIPADECS = 10, iterations of Basic DE = 50 PopulationDEFW = 25 , IterationsDEFW = 200, iterSFGSS = 8, iterSFHC = 20, Fl = 0.1, Fu = 0.9
TSKNN	Evaluations = 10,000, M = 10, N = 2, P = ceil($\sqrt{\#\text{Features}}$)

Table 10
Common parameters of FWFOA for all selected data sets.

Life time	Area limit	Transfer rate
6	30	10%

Table 11
The value of *"LSC"* and *"GSC"* for each data set.

Data set	Bupa	Cleveland	Dermatology	Glass	Iris	Pima	Heart
"LSC"	1	3	7	2	1	2	2
"GSC"	2	4	10	3	1	2	4

4.3.3. Parameters of FWFOA

Table 10 depicts the value of the parameters of FWFOA which are the same for all the data sets. Because the value of *"LSC"* and *"GSC"* parameters depend on the number of the features of each data set, so the value of these parameters are shown separately as Table 11.

4.3.4. Results and discussion

Table 12 illustrates the result of FWFOA. For each data set, *ACC* and *SD* show the average classification accuracy and standard deviation of 10 times execution respectively. In each run, the algorithm execution stops when the value of the best solution does not change for 100 consequent iterations. Classification accuracy for data sets that FWFOA outperformed to other methods is highlighted in bold form.

As it is obvious from comparing Tables 8 and 12, FWFOA outperforms the methods from Triguero et al. (2012) in 5 data sets among 7 ones; because classification accuracy for Bupa, Cleveland, Dermatology, Glass and Iris data sets have been improved while applying FWFOA. So, it can be concluded that FWFOA performs

Table 8
Classification accuracy and standard deviation obtained reported in Triguero et al. (2012). The best classification accuracy (ACC) is highlighted in bold form for each data set.

	INN		**SSMA**		**SSMA-DEPGFW**		**1 PA DECS**		**IPADECS-DEFW**		**TSKNN**	
Data sets	ACC	SD	ACC	SD	ACC	SD	ACC	SD	ACC	SD	ACC	SD
Bupa	61.08	6.88	62.7	8.47	**67.41**	7.96	65.67	8.48	67.25	5.27	62.44	7.90
Clevland	53.14	7.45	54.7	6.29	55.80	6.11	52.46	4.48	54.14	6.20	**56.43**	6.84
Dermatology	95.35	3.45	95.1	5.64	94.02	4.31	96.18	3.01	**96.73**	2.64	96.47	4.01
Glass	73.61	11.9	68.8	8.19	73.64	8.86	69.09	11.13	71.45	11.94	**76.42**	13.21
Iris	93.33	516	**96.0**	4.42	94.67	4.99	94.67	4.00	94.67	4.00	94.00	4.67
Pima	70.33	74.3	74.23	4.01	73.23	5.43	**76.84**	4.67	71.63	7.35	75.53	5.85
Heart	77.04	8.89	83.7	10.1	**85.19**	8.11	83.70	9.83	80.74	9.19	81.48	6.42

Table 12
Classification accuracy (*ACC*) and standard deviation (*SD*) obtained by FWFOA.

Data set	Bupa	Cleveland	Dermatolgy	Glass	Iris	Pima	heart
ACC	**67.60**	**58.14**	**97.37**	**78.62**	**96.66**	71.11	81.35
SD	0.43	0.82	0.31	0.7	0	0	0.37

better search in weight space of some data sets in comparison with other methods. Also, the small standard deviation in all the results of Table 12 proves the stability of FWFOA while comparing other methods.

5. Comparison of Forest Optimization Algorithm with other search methods

In this section we will compare FOA with other search methods. To do so, we have compared FOA with Gradient-Based local optimization, hill climbing, simulated annealing and random search all as conventional search methods. Then we have provided comparisons between FOA and other evolutionary algorithms like GA and PSO. These comparisons will make clear the advancements of FOA over other search strategies.

5.1. Comparison of FOA with conventional methods

Gradient-Based local optimization method is used when the objective function is smooth and one needs efficient local optimization. This method needs computing derivations so, it may be problematic with not derivable functions. Gradient-Based methods duo to their need to auxiliary information like derivative in comparison to evolutionary algorithms are not mostly used in multidisciplinary engineering problems (Foster & Dulikravich, 1997). For more details please refer to (Sivanandam & Deepa, 2007; Snyman, 2005).

Stochastic Hill Climbing is another conventional method for search problems (Sivanandam & Deepa, 2007). It uses a kind of gradient to guide the direction of search and is suitable for well-behaved continuous spaces. The obvious disadvantage with hill climbing methods is their inefficiency in noisy spaces with many local optimum; as they are probable of getting stuck in local optimum. As the result, they are not welcomed in noisy spaces. Also, hill climbing methods are expensive in using resources and its time complexity is high.

Random search, which is a basic method, explores the search space by randomly selecting solutions. Although random search never gets stuck in a local optimum, but if the search space is not finite, this method seems useless (Sivanandam & Deepa, 2007).

Simulated Annealing (SA) mixes features of random search and exploitation features of hill climbing to give quite good results (Sivanandam & Deepa, 2007). SA is a serious competitor to Genetic Algorithms, but evolutionary algorithms like GAs have two main differences with SA which makes them more efficient. First, evolutionary algorithms are population-based methods whereas SA only examines one potential solution at each iteration. The great advantage of evolutionary algorithms is their exceptional ability to be parallelized, whereas SA does not have this advantage. More details about SA and its derivations can be seen in Ingber (1993).

5.2. Comparison of FOA with other evolutionary algorithms

The procedure of most evolutionary algorithms is as the following:

1. Generating an initial population randomly.
2. Calculating the fitness for each individual.

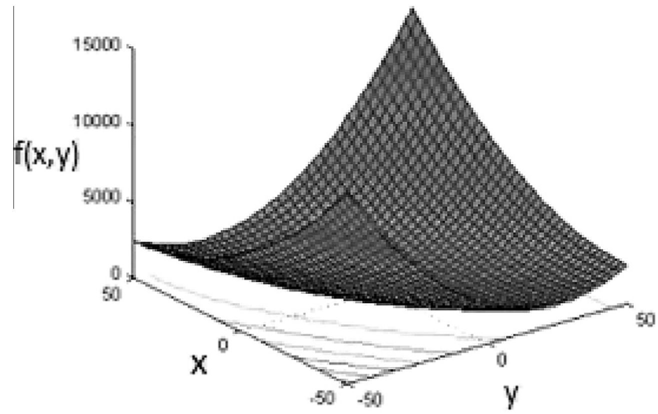


Fig. 14. 3D plot of *"Rotated hyper-ellipsoid"* test function.

3. Reproduction of the population based on fitness values.
4. If requirements are met, then stop. Otherwise go back to 2.

While considering these steps, all the evolutionary algorithms share many common points with other methods but, the operations make them different with each other. If we compare the information sharing of GA, PSO and FOA, we will see that information sharing of them are significantly different and we reach to the following conclusions: Because in GA all the chromosomes share information with each other, so the whole population moves towards the optimal solution like a one group. But, in PSO just the 'global best' particle shares its information with other particles and all the particles converge to the best solution quickly (Sivanandam & Deepa, 2007). In contrary in FOA, some of the trees (zero-aged trees including the best tree) share their information with just their neighbors and not all the trees. As the result, the information sharing of GA, PSO and FOA is different. From the other side, PSO uses extra memory to store best particle. In the following we will statistically study the behavior of FOA ,GA and PSO in a smooth function.

5.2.1. Study the behavior of FOA, GA and PSO in a smooth function

In order to compare the behavior of FOA, GA and PSO, we have chosen *"Rotated hyper-ellipsoid function"* function as a unimodal smooth function (Blake et al., 1998); which is plotted as Fig. 14 and its global minimum of $f(x) = 0$ is obtainable for $x_i = 0,\ i = 1, \ldots, n$ in the range $-65.536 \leqslant x_i \leqslant 65.536$. We have compared the results in 2D and the results are the average of 25 independent runs. We have done two series of experiments with the same parameters of Table 5 for GA, PSO and FOA in 2 dimension: At first we considered the level of accuracy of the best solution from the optimal solution to be fixed as 0.0001 for FOA, GA and PSO and compared them in the case of number of evaluations (num-evals) as Table 13. Next in another attempt, we fixed the number of evaluations to be 2000 and compared them in the level of accuracy of the best solution from the optimal solution and summarized the results as Table 14.

As it is obvious from Table 13, FOA needs much less evaluations than both GA and PSO algorithms to reach to the specified level of accuracy (0.0001) of the best solution from the optimal solution. Also considering the results of Table 14 illustrates that after the fixed number of evaluations (2000 evaluations) FOA moves closer to the optimal solution with better level of accuracy from the optimal solution in comparison with GA and PSO. In conclusion, the experiments show that FOA behaves much better than both GA and PSO in *"Rotated hyper-ellipsoid"* function as a smooth unimodal test function. Another result of this experiment could be the acceptable performance of FOA in finding local optimum; because

Table 13
Comparison of FOA, GA and PSO in the number of evaluations *(num_evals)* with fixed level of accuracy (0.0001).

Algorithm	FOA	GA	PSO
(num_evals)	1155.6	1754.4	5012

Table 14
Comparison of FOA, GA and PSO in the level of accuracy with fixed number of evaluations *(num_evals)* (2000).

Algorithm	FOA	GA	PSO
(level_of_accuracy)	7.2076e−05	1.3473e−04	0.0521

as we mentioned before local seeding stage performs local search so, higher level of accuracy could represent the good performance of FOA in performing local search.

6. Conclusions and future work

In this article, a new evolutionary algorithm – Forest Optimization Algorithm (FOA) – is proposed. FOA is inspired by some procedures in the forests and simulates the most obvious procedure in the forest which is known as seed dispersal. In the forests, the trees with enough sunlight and good growing conditions can live longer than other trees and different procedures help trees to continue their generation for millions of years, which is named as seed dispersal. When the seeding procedure begins, some seeds fall just beneath the parent trees themselves and this event is known as local dispersal (referred as local seeding in FOA). Mostly natural procedures like animals and the wind distributes the seeds in wide areas and this procedure is known as long-distance seed dispersal (referred as global seeding in FOA). Also, there is always a competition between neighboring trees to use the life essentials and the winners are those trees with better life conditions and other trees eventually get old and die. In this article, we proposed FOA to simulate these procedures to further optimize the solutions of the nonlinear optimization problems.

We have tested the performance of FOA in 4 benchmark functions and feature weighting as a real optimization problem. We chose 4 (unimodal and multimodal) benchmark functions and evaluated the performance of FOA on them. Also we have done experiments to find the optimal value for the parameters of FOA. In order to better compare the behavior of FOA in comparison with GA and PSO, we tested 3 of the 4 benchmark functions in 5 and 10 dimensions; which increase the complexity of the experiments. The results of the experiments on almost all of the selected functions in 2, 5 and 10 dimensions showed that, FOA needs less number of evaluations to reach to a specific level of accuracy in comparison with GA and PSO; which is a good improvement. Also in another experiment, we compared the behavior of FOA in a smooth function with both GA and PSO; the results showed the superiority of FOA to both GA and PSO, because FOA needs less number of evaluations and it has high level of accuracy in comparison with GA and PSO.

After observing the acceptable results of FOA in benchmark functions, we have also attempted to solve feature weighting as a real challenging optimization problem and we used FOA to guide the feature weights as a preprocess step of data mining. We selected some data sets from UCI repository and the application of FOA on feature weighting problem (FWFOA) showed a good performance of FOA; because in some data sets FOA succeed to find optimal weights for features to improve the performance of KNN (K-Nearest-Neighbor) as a learning algorithm.

In this article we have tried to simulate the most common procedures among the trees of the forest, which is known as seed dispersal; but different trees use interestingly different methods in order to disperse their seeds in the forest. So, considering specific kinds of plants with their own way of survival and dispersal could lead us to better results and this consideration is one of our future attempts. Also, studying the behavior of FOA in multi-objective problems seems encouraging and it will help us to better study the importance of each parameter of FOA. In addition, in the future we will also apply FOA in more challenging optimization problems to further study the behavior of this algorithm.

FOA is proposed to solve continuous search problems and applying it to feature weighting showed its good performance in solving real continuous problems. Our ongoing research is to adjust FOA to solve discrete problems like feature selection; because feature selection is a special case of feature weighting with the weights limited to just 0 and 1 and due to promising results in feature weighting, we expect good results in feature selection too. The nature's procedure in the forests is more complex than what we have considered in this article and any more study on each procedure could lead us to better results and this is what we will invest on that in the future.

References

Aha David, W. (1998). Feature weighting for lazy learning algorithms. In H. Liu & H. Motoda (Eds.), *Feature extraction construction and selection: A data mining perspective* (pp. 13–32). Massachussetts: Kluwer Academic Publishers.

Blake, C., Keogh, E., & Merz, C. J. 1998. *UCI Repository of machine learning databases*, University of California, Irvine. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Cain, M. L., Milligan, B. G., & Strand, A. E. (2000). Long-distance seed dispersal in plant populations. *American Journal of Botany, 87*(9), 1217–1227.

Das, A., Battles, J., Stephenson, N. L., & van Mantgem, P. J. (2011). The contribution of competition to tree mortality in old-growth coniferous forests. *Forest Ecology and Management, 261*(7), 1203–1213.

Foster, N. F., & Dulikravich, G. S. (1997). Three-dimensional aerodynamic shape optimization using genetic and gradient search algorithms. *Journal of Spacecraft and Rockets, 34*(1), 36–42.

Gosling, P. (2007). Raising trees and shrubs from seed: Practice guide. Forestry Commission. 231 Corstorphine Road, Edinburgh EH12 7AT.

Green, D. S. (1983). The efficacy of dispersal in relation to safe site density. *Oecologia, 56*(2-3), 356–358.

Howe, H. F., & Judith, S. (1982). Ecology of seed dispersal. *Annual Review of Ecology and Systematics*, 201–228.

Hubbell, S. P. (1979). Tree dispersion, abundance, and diversity in a tropical dry forest. *Science, 203*(4387), 1299–1309.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling, 18*(11), 29–57.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks, 4*, 1942–1948.

LepS, J., & Kindlmann, P. (1987). Models of the development of spatial pattern of an even-aged plant population over time. *Ecological Modelling, 39*(1), 45–57.

Marcin, M., & Smutnicki, C. (2005). Test functions for optimization needs Retrieved June 2013, from <http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf>.

Murrell, D. J. (2009). On the emergent spatial structure of size-structured populations: When does self-thinning lead to a reduction in clustering? *Journal of Ecology, 97*(2), 256–266.

Ozsen, S., & Gunes, S. (2009). Attribute weighting via genetic algorithms for attribute weighted artificial immune system (AWAIS) and its application to heart disease and liver disorders problems. *Expert systems with applications* (Vol. 36, pp. 386–392). Elsevier.

Rajabioun, R. (2011). Cuckoo optimization algorithm. *Applied Soft Computing, 11*, 5508–5518.

Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms*. Berlin, Heidelberg: Springer-Verlag.

Snyman, J. (2005). *Practical mathematical optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms*. Springer.

Tahir, M. A., Bouridane, A., & Kurugollu, F. (2007). Simultaneous feature selection and feature weighting using hybrid tabu search/K-nearest neighbor classifier. *Pattern recognition letters* (Vol. 28, pp. 438–446). Elsevier.

Tosun, A., Turhan, B., & Bener, A. B. (2009). Feature weighting heuristics for analogy-based effort estimation models. *Expert systems with applications* (Vol. 36, pp. 10325–10333). Elsevier.

Triguero, I., Derrac, J., Garcia, S., & Herrera, F. (2012). Integrating a differential evolution feature weighting scheme into prototype generation. *Neurocomputing* (Vol. 97, pp. 332–343). Elsevier.

Triguero, I., Garcia, S., & Herrera, F. (2011). Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. *Pattern recognition* (Vol. 44, pp. 901–916). Elsevier.