Mushroom Reproduction Optimization (MRO): A Novel Nature-Inspired Evolutionary Algorithm

Conference Paper · September 2018

DOI: 10.1109/CEC.2018.8477837

4 authors:

Mahdi Bidar
University of Regina
15 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Hamidreza Rashidy Kanan
Shahid Rajaee University (SRTTU)
66 PUBLICATIONS **642** CITATIONS
SEE PROFILE

Malek Mouhoub
University of Regina
127 PUBLICATIONS **430** CITATIONS
SEE PROFILE

Samira Sadaoui
University of Regina
76 PUBLICATIONS **220** CITATIONS
SEE PROFILE

Mushroom Reproduction Optimization (MRO): A Novel Nature-Inspired Evolutionary Algorithm

Mahdi Bidar
Department of Computer Science
University of Regina
Regina, Canada
bidar2m@uregina.ca

Hamidreza Rashidy Kanan
Department of Computer Engineering
Shahid Rajaee Teacher Training University
Tehran, Iran
h.rashidykanan@srttu.edu

Malek Mouhoub and Samira Sadaoui
Department of Computer Science
University of Regina
Regina, Canada
{sadaouis,mouhoubm}@uregina.ca

***Abstract*—We introduce a new nature-inspired optimization algorithm namely Mushroom Reproduction Optimization (MRO) inspired and motivated by the reproduction and growth mechanisms of mushrooms in nature. MRO follows the process of discovering rich areas (containing good living conditions) by spores to grow and develop their own colonies. We thoroughly assess MRO performance based on numerous unimodal and multimodal benchmark functions as well as engineering problem instances. Moreover, to further investigate on the performance of the proposed MRO algorithm, we conduct a useful statistical evaluation and comparison with well known meta-heuristic algorithms. The experimental results confirm the high performance of MRO in dealing with complex optimization problems by discovering solutions with better quality.**

I. Introduction

Optimization is the process of finding a scenario that minimizes or maximizes an objective function while satisfying a given set of constraints. This scenario, called optimal solution, is often searched in an exponential set of candidate solutions, and therefore requires a very expensive execution time. To overcome this difficulty in practice, meta-heuristic approximation methods have been introduced. While these solving methods do not guarantee the solution optimality, they are however very successful in returning near-to-optimal solutions [1], [2], [3]. Two strategies are used by meta-heuristic algorithms to find the best solution, exploitation and exploration synonymous with intensification and diversification respectively. Generally speaking, exploitation looks for the best solution within the local scale whereas exploration searches globally for the solution in the problem space [4]. Exploration prevents meta-heuristic algorithms from being stuck in local optima. These algorithms have been widely applied to high-dimensional optimization problems, such as optimization of objective functions [5], operation and control of power systems [6], job scheduling and assignment [7], [8], [9], pattern recognition [10] and image processing [11]. In this paper, we focus on nature-inspired meta-heuristic algorithms that can be divided into three categories according to their source of inspiration [12]: a) Swarm intelligence algorithms, b) Non-swarm intelligence algorithms, c) Physical and chemical algorithms.

The swarm intelligence class refers to the collaborative behaviour of agents (often unintelligent) of decentralized and self-organizing swarm systems. By following simple governing rules to perform simple tasks for reaching the goals, the collective intelligence is attained. The most well-known algorithms belonging to this category are: Ant Colony Optimization (ACO), inspired by the foraging behaviour of real ants, Particle Swarm Optimization (PSO), inspired by the swarm intelligence of birds and fishes, Honey Bee Colony Optimization (HBCO), based on honey-bee behaviour in finding food sources, Firefly Algorithm (FA) which mimics the social behaviour of fireflies and their light emitting characteristics, Bat Algorithm (BA) inspired by the echolocation behaviour of bats, Cuckoo Optimization Algorithm (COA) inspired by the cuckoo characteristics in laying eggs and breeding, Krill Herd (KH), based on the herding behaviour of krill individuals in their environment, Grey Wolf Optimizer (GWO), inspired by the leadership hierarchy and hunting mechanisms of grey wolves, and Ant Lion Optimizer (ALO) which mimics the hunting mechanism of real ant lions. On the other hand, non-swarm intelligence algorithms are not endowed with the collaborative behaviour of their agents. The popular algorithms of this category include: Genetic Algorithms (GAs) based on the Darwinian evolution and theory of survival of the fittest, Differential Evolution (DE) which consists of improving a potential solution w. r. t. a given measure of the solution quality, Invasive Weed Optimization (IWO), simulating the ecological behaviour of weeds in finding places for growth and reproduction, and Biogeography-Based Optimization (BBO) which borrows ideas from island biogeography. The third category of algorithms is inspired by the physical or chemical processes, and the most famous ones are: Gravitational Search Algorithms (GSAs), based on the law of gravity and mass interactions, Interior Search Algorithms (ISAs), inspired by the interior design and decoration, and Central Force Optimization (CFO), inspired by metaphor of gravitational kinematics. Although the aforementioned nature-inspired algorithms provide satisfactory results, however an algorithm may be successful in certain optimization problems but not in others [13]. Thus, developing a new optimization method to deal effectively with a wider range of problems as well as to tackle efficiently more complex problems is still a recognized research objective [13] [14]. Our aim in this present study is to elaborate a novel optimization algorithm inspired by the real mushroom reproduction process in nature. This algorithm mimics the way mushrooms reproduce and migrate to rich areas with adequate nourishment, moisture and light. The reproduction model is twofold: a) producing spores, and b) distributing them randomly in the environment. This reproduction mechanism leads to bigger mushroom colonies located in different regions with suitable living conditions. We model this mechanism as an optimization strategy capable of finding the fittest regions containing high-quality solutions and probably the optimal solution. Based on this strategy, we devise a Mushroom Reproduction Optimization (MRO) algorithm, which belongs to the non-swarm intelligence class. Unlike some other meta-heuristic algorithms, such as Firefly, PSO and COA, the searching process in MRO does not consider the distance between the agents. Below we highlight the main characteristics of MRO:

1) From the view point of exploitation, MRO introduces a new strategy for searching the local areas. This consists of evaluating the neighbouring areas of the search agents (parent mushrooms), refining these areas by releasing spores, and pursuing the local search towards higher quality points. Thanks to this local searching strategy, if there is a better solution in the neighbouring area of a parent mushroom, MRO will most probably discover it.
2) Regarding the exploration, spores of parent mushrooms are distributed randomly in the problem space. The movement of spores is highly influenced by the external factor wind. The latter has a stochastic behaviour in terms of speed and direction. The fittest agents are selected as the new parent mushrooms. Another important distinct feature of MRO is its capability in locating richer areas containing better quality solutions and probably the optimal solution instead of looking only for qualified solutions.
3) MRO is able to record the searching process. In fact, MRO keeps track of the achievements of each colony, which are then used to make a decision about the next movement. For instance, decide about the colonies that should be eliminated from the problem space.

II. Mushroom Reproduction Optimization (MRO)

A. Mushroom Life Cycle in Nature

Most mushrooms are designed with the thread-like structure known as hyphae. The latter fuses with other compatible hyphae to form a complex structure, called a mycelium [15]. Mycelium develops into a pinhead and then grows into a mature mushroom. The suitable temperature for mushrooms to grow is in the range of 20 to 30 centigrade. Due to the fact that mushrooms do not have chlorophyll, therefore they require ready-made food for nourishing, like hydrocarbon, proteins and nitrates. Mushrooms grow on plants and animal corpses in jungles, grasslands and farms. These biological sources form appropriate living conditions by supplying rich substances (moisture, light and nutrition) to mushrooms to grow and make big colonies. The reproduction mechanism of mushrooms is twofold: 1) producing spores, and 2) distributing them randomly in the environment. A spore is a plant seed that germinates by settling in suitable growth lands and later develops hyphae. The germination of spores and development of hyphae are part of the vegetative stage.

After completing the growth of mycelium, mushrooms enter the reproductive stage. The produced spores are released into the environment in several ways including animals, water and most importantly wind. Considering the speed and direction of the wind, spores travel to different places in the surrounding environment. Nevertheless, only a small percent of spores settles and grows while the rest is wasted. By landing in proper areas, spores grow and establish new colonies. In the situation when there is no wind but the living conditions are good, spores land in the local area of mature mushrooms, grow and proliferate. If a colony settles in a poor area, it will extinct eventually. Still very few spores may survive and will be moved by wind to other areas. The size of a colony is proportional to the richness of its regions. In a nutshell, a mushroom grows taller and wider and at the adult stage releases spores and distribute them in the environment, and the process restarts again. As we can see, spores are the survival factor of mushrooms.

B. Overview of MRO

The MRO algorithm is an agent-based technique inspired by the reproduction model of mushrooms in nature. The searching agents are parent mushrooms and spores. The transporter mechanism is the artificial wind that stochastically distributes spores in different locations of the problem space. Only a small percent of spores will be lucky enough to survive by landing in rich areas containing all the nourishment for growth. The probability of developing colonies in rich areas is higher than poor areas. In fact, rich areas are those containing high-quality solutions and probably the optimal one. In MRO, poor quality colonies are eliminated to be replaced by better colonies. In the initial phase, a mature mushroom (the parent), located in the problem space, distributes spores throughout the problem space. If no wind is blowing and the living conditions in the neighbouring area are good, then upon landing, spores germinate and grow. However, if the wind is blowing, spores are moved to different parts of the problem space and land in new locations. Next, spores grow and become mature mushrooms. This searching process of production and distribution is repeated, and some spores will ultimately find the richest area. In the final phase, the main operation, and consequently the colony expands locally and quickly. This will result in finding the optimal solution.

The strategy of finding the richest area by mushrooms is the basis for developing our MRO algorithm. The goal is to find the optimal solution (the best parent mushroom), which is probably in the richest area. The latter has the highest average of the fitness values of its individuals. In summary, parent mushrooms search the problem space by distributing spores locally and globally in the environment. They expand their colonies locally when the living conditions are good. But when a parent mushroom is in a poor area and the wind is blowing, its spores move to different places in the problem space in order to find better living areas. As we can see, MRO utilizes a combination of exploration and exploitation for searching the problem space alike the other meta-heuristic algorithms.

III. MRO Process

A. Initialization

In the initialization phase, *M* initial parent mushrooms are stochastically released in the search space. A colony is represented by the parent mushroom who is permitted to release exactly N spores in each reproduction iteration. Each mushroom and spore correspond to a potential solution denoted by *X*, and according to the richness of its area, gets its own fitness value indicating the quality of the found solution.

B. Local Reproduction as the Local Search

In each colony, one mushroom is selected as a parent who is allowed to reproduce. If a colony is already in a rich area, upon landing besides the parent mushroom, spores grow and become mature mushrooms with their own fitness values. This process causes the local extension of the colony, which results in discovering more quality solutions in the neighbouring area of the parent mushroom. Assume that the biggest mushroom owns the best solution. So, it is chosen as the new parent and will extend the colony in the defined radius. Actually, there are two strategies for selecting the new parent as the fittest: a) the best mushroom in the direct neighbourhood in terms of the highest fitness value for maximization problems and lowest fitness value for minimization problems; and b) by using the fitness proportionate selection strategy (also known as roulette wheel selection). The second strategy is determined by equation (1) where the function *F()* assigns fitness to the candidate solution X_j (current parent mushroom and spores) in colony *i* and P_j is the probability of selecting spore j as the new parent of colony *i*.

$$P_j = \frac{F(X_{ij})}{\sum_{j=1}^{N} F(X_{ij})} \tag{1}$$

While solutions with higher fitness values have a higher probability to be selected, there is still a chance for others to be chosen as well. The local search for those colonies that are living in appropriate circumstances will continue. As time passes, the richness of a region of a colony may increase or decrease. In MRO algorithm, the local extension of a colony is performed by placing *N* spores in the neighbouring area of the parent mushroom within the radius *r*. The position of a new spore is defined based on equation (2) where $i = 1, \ldots, M$, $j = 1, \ldots, N$ and *Rand()* generates random numbers (uniformly distributed) in the determined interval.

$$X_{ij} = X_i^{parent} + \overrightarrow{Rand(-r, r)} \tag{2}$$

C. Wind for the Global Search

If the living area does not contain proper moisture and nourishment (the indigence condition), spores land in the neighbouring area, stay inactive and wait for the wind to transport them. The wind blows successively in different directions and carry the light pieces to different places in the problem space. This provides the algorithm an opportunity to explore other areas of the search space and to produce diversified solutions. The spores produced by a parent mushroom stochastically travel by wind to new locations. In MRO, among the distributed spores, only the fittest one is permitted to live and the rest are eliminated. The movement model of the spores is shown in equation (3) where X_{ij} is the new position of the *spore j* of *colony i*, $X_i{}^{parent}$ the position of the parent mushroom and Mov_j^{wind} the stochastic movement of spore *j* influenced by the wind.

$$X_{ij} = X_i^{parent} + Mov_j^{wind} \tag{3}$$

When searching the problem space globally, spores should move through the entire space in an attempt to find appropriate locations to form new colonies. Such locations can be found by the successive search in the problem space. When a colony is located in rich area (in term of the quality of their solutions), its location can be considered as a clue for discovering new good locations. Thus, the agents' movements are influenced by the location of other colonies to ensure the progress of the algorithm. In fact among the distributed spores, those that move foreward in the surrounding area of already found rich regions have more chance to survive since such area is probably rich too. This effect is reflected in the movement equation :

$$Mov_j^{wind} \propto X_i^* - X_k^* \tag{4}$$

where X_i^* is the best solution of colony i (the parent's solution) and X_k^* (k = 1 ... M, $k \neq i$) the best solution of colony k.

To distribute spores within an appropriate distance in the problem space, we consider the inverse relationship between the average of the fitness values of the solutions in colony *i* (*Avg(i)*) and the total average of fitness values of all found solutions by all the colonies in the current iteration(T_{avg}). Thus, equation (4) is improved to equation (5):

$$Mov_j^{wind} \propto (X_i^* - X_k^*) \times (\frac{Avg(i)}{T_{ave}})^{-m} \tag{5}$$

where *m* is a user-defined parameter to set the intensity of the wind. To determine the direction of the wind, which is a stochastic feature, a random vector must be included in equation (5) where δ is the direction coefficient (a real number) of the wind.

$$Mov_j^{wind} \propto (X_i^* - X_k^*) \times (\frac{Avg(i)}{T_{avg}})^{-m} \times \overrightarrow{Rand(-\delta, \delta)} \tag{6}$$

To adjust the size of the random step, the controlling parameter (random step size) *rs* is also included in equation (6):

$$Mov_j^{wind} \propto (X_i^* - X_k^*) \times (\frac{Avg(i)}{T_{avg}})^{-m} \times \overrightarrow{Rand(-\delta, \delta)} \times rs \quad (7)$$

Finally, we include a movement to a random position within the radius r:

$$Mov_j^{wind} = (X_i^* - X_k^*) \times (\frac{Avg(i)}{T_{avg}})^{-m} \times \overrightarrow{Rand(-\delta, \delta)} \times rs + \overrightarrow{Rand(-r, r)} \quad (8)$$

In summary, since wind moves stochastically, spores move stochastically in the problem space. Considering the difference between Avg and T_{avg}, spores will be distributed with different distances. As this difference decreases, the dispersion distance of the spores decreases. Finally, decreasing the controlling parameter rs from 1 to 0 declines the size of the random step (i.e. decreases the exploration) and emphasizes the exploitation. So, spores land in a new position and begin to make their own new colonies.

D. An Illustrative Example

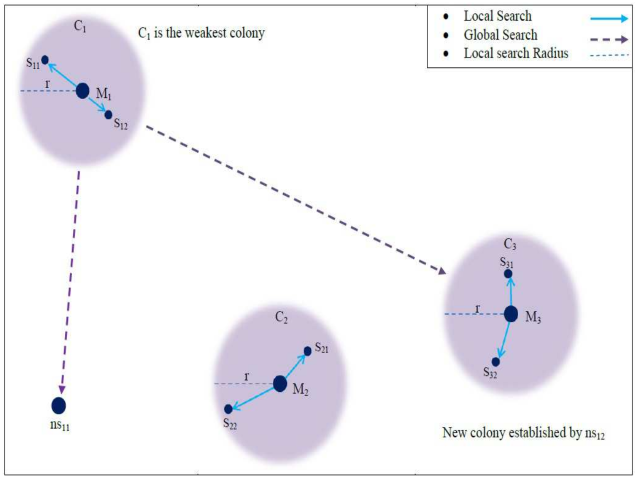


Figure 1. An Example of local and global search

To better understand the MRO algorithm, we illustrate its behaviour through an example. In Figure 1, we assume C_1 and C_2 are the initial colonies, M_1 and M_2 the parent mushrooms respectively, and only 2 spores can be released. The shadow zone around the parent mushrooms refers to the local random search performed in the neighbouring area with a certain radius. For each colony, the average fitness of its individuals/solutions ($avg(i)$) and its best solution X_i^* are calculated. The former is used to decide whether to keep or delete a colony based on its overall quality and the latter is to update the global best solution and determine the new parent mushroom. Suppose C_1 is the weakest colony (for example with the smallest average fitness for maximization problems), we then replace it with a better quality colony called C_3. This is done by conducting the global search that consists of distributing new spores ns_{11} and ns_{12} using the virtual wind (see equation 8) through the problem space. Since ns_{12} has a higher quality, it is chosen as the parent mushroom of the new colony. Assume now the algorithm is in its last iteration, then the best of the two best solutions of C1 and C3 becomes the global best solution that will be returned by MRO.

IV. MRO Algorithm

A. The algorithm

In Figure 2, we expose the steps of the MRO pseudocode and its parameters are given in Table I.

Table I
MRO Parameters

Parameter	Definition
M	Number of parent mushrooms (colonies)
N	Number of distributing spores of a parent mushroom
r	Radius of the local search/reproduction
c	To decide whether to delete a colony or not
$m \in [1, \infty)$	To adjust the wind intensity
$rs \in (0,1)$	To control the random step size
$\delta \in (0,1)$	To adjust the direction coefficient of wind

In the algorithm, the indigence condition $Avg(i) + T_{Avg}/c < T_{Avg}$ is used to estimate the quality of a colony. If the fitness is high or in an acceptable range, then the colony keeps developing in the local area; otherwise it must be deleted and MRO globally searches for a new location in the problem space by distributing spores using the movement equation. Here the threshold parameter c helps the algorithm to decide about the colonies that should be destroyed w. r. t. their qualities. The second condition $F(X_k^*) < F(X_i^*)$ recognizes better colonies to be used by the algorithm as clues of richer areas. By considering the locations of quality colonies, the MRO algorithm successively searches the problem space to find better locations. The global best solution and its fitness value are the outputs of MRO algorithm.

B. The Clarifier Example

We visually illustrate the execution process of MRO based on the peak function given in equation 9 [16]. The 3-D plot and contour of the peak function are respectively presented in the left and right charts of Figure 3.

$$Z = (|x| + |y|)e^{-0.0625\times(x^2+y^2)} \quad (9)$$

In this testing, we utilize 25 parents mushrooms. In each reproduction iteration, a mushroom is permitted to distribute exactly four spores, and only the fittest one is selected as the surviving spore to continue the lifecycle. The running steps of MRO are exposed in Figures 4(a)-(f). Figure 4 shows the convergence trend of the agents to the peaks. Figure 4(a) exposes the initialization of MRO in which the parent mushrooms are randomly distributed in the problem space. Figure 4(b) depicts how MRO extends the colonies by distributing spores in the neighbouring radius r of each parent mushroom. Afterwards, the algorithm ranks the colonies based on the averaged fitness values, and then enters the main loop of the searching process. Figure 4(c) shows that all the search agents are approaching the optimal points of the problem space. In each iteration, MRO identifies the poorest regions and if they satisfy the

```
Randomly generate M parent mushrooms
**For i = 1: M**
    **For j = 1: N**
        $X_{ij} = X_i^{parent} + \overrightarrow{Rand(-r, r)}$
        //generate spore $j$ for colony $i$
    **Endfor**
**Endfor**
**For i = 1: M**
    Calculate $F(X_i^{parent})$
    Calculate Avg(i)
    //average fitness of colony $i$
    Select best agent $X_i^*$ in colony $i$
    $X_i^{parent} = X_i^*$
**Endfor**
Calculate T$_{avg}$
//total average fitness of all colonies
**While** (termination condition is not met)
    **For i = 1: M**
        **If** $Avg(i) + \frac{T_{Avg}}{c} < T_{Avg}$ **then**
            **For k = 1: M**
                **If** $F(X_i^*) < F(X_k^*)$ **then**
                    **For j = 1: N**
                        $X_{ij} = X_i^{parent} + Mov_j^{wind}$
                        //move spore $X_{ij}$ with artificial wind
                    **Endfor**
                    Select best agent $X_i^*$ in colony $i$
                    $X_i^{parent} = X_i^*$
                    Update global best solution
                **Endif**
            **Endfor**
        **Endif**
        **For j = 1: N**
            $X_{ij} = X_i^{parent} + \overrightarrow{Rand(-r, r)}$
        **Endfor**
        Calculate Avg(i)
        Select best agent $X_i^*$ in colony $i$
        $X_i^{parent} = X_i^*$
        Update global best solution
    **Endfor**
    Calculate T$_{avg}$
**Endwhile**
Return global best solution
```

Figure 2. Mushroom Reproduction Optimization Algorithm

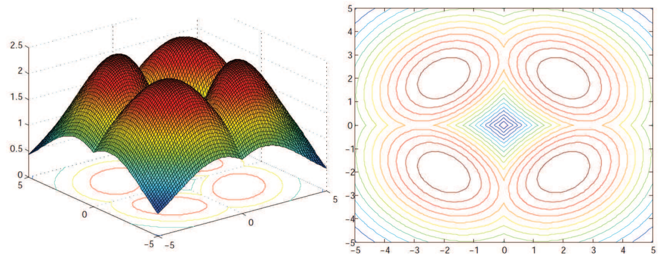


Figure 3. The 3-D plot of the peak function (left) and the contour map of the peak function (right)

indigence condition, the algorithm destroys their colonies and reallocates to new locations using the wind factor. Based on the fittest selection strategy of equation 1, the algorithm decides which spores can stay alive and grow. Figures 4(d)-(f) illustrate how the colonies are approaching the richest area containing the best solution. The search agents converge to the optimal solution after 10 iterations.

V. Experimentation

We report on the experiments conducted to assess the efficiency of the proposed MRO when compared with several known meta-heuristic techniques selected from each of the three categories mentioned in section 1. Indeed, we apply MRO and 9 other algorithms to 20 well-known uni and multi-model benchmark functions in addition to engineering problems and report their performance results. To facilitate the comparison, the best run are normalized between 0 and 1 (where 0 and 1 respectively correspond to the worst and best solution) using equation (10) [17].

$$f(X_i)_{normalized} = 1 - \frac{f(X_i) - f(X_{\min})}{f(X_{\max}) - f(X_{\min})} \tag{10}$$

where $f(X_i)$ is the fitness value of solution X_i, $f(X_{min})$ the minimum fitness value and $f(X_{max})$ the maximum fitness value achieved in our experiment.

A. Uni and Multimodal Benchmark Functions

The benchmark set contains ten high dimensional (F1-F10) and ten low dimensional (F11-F20) functions (see Table II) taken from [18] [19] [20] [21]. The definition of the constants (c, a and p) of functions F15-20 can be found in [21]. The aim of the optimization task is to find the best solution that minimizes a function.

B. Parameter Setting

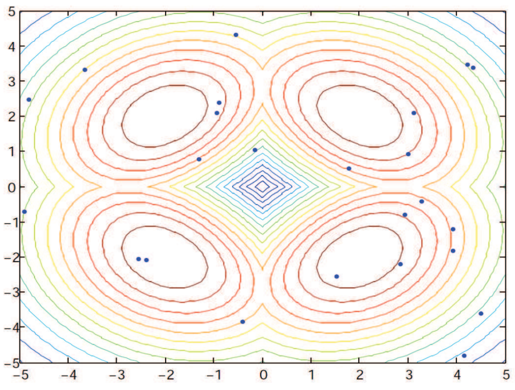
The population size of MRO is set to $M = 20$, each parent mushroom is allowed to produce $N = 4$ spores in each iteration, the radius r is determined w. r. t. the definition of the problem, and the maximum number of iterations is equal to 200 and rs is 0.8. For high dimension functions, the dimension n is set to 20, the threshold factor c has a real value between 2 and 10, and the search space is a subset of R^n. Since the experiments must be done under fair circumstances in terms of the number of evaluations, so if $N = a$, then the number of executions for MRO is determined as $200/a$.

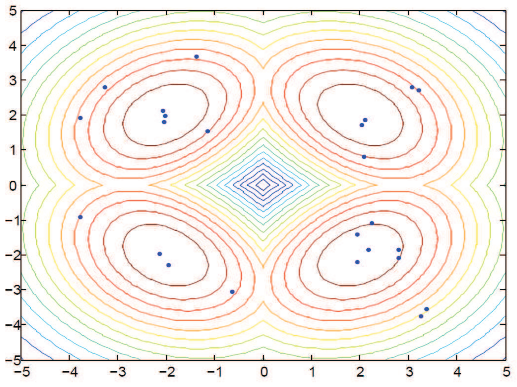
C. Performance Evaluation

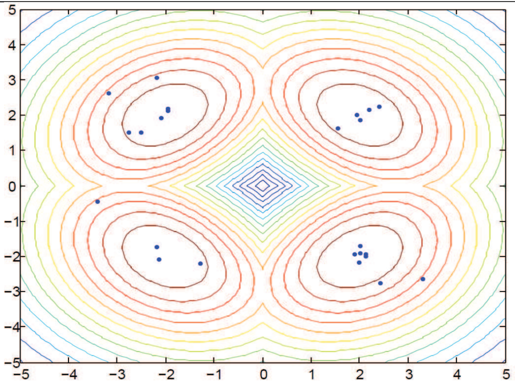
Due to the stochastic behaviour of MRO and the other algorithms, their real performance cannot be judged based on the result of a single run. Thus, the final results are obtained in 50 trials. The normalized statistical results of all the algorithms on high and low dimension functions are exposed respectively in Tables III and IV. In these tables, *Best* is the best solution ever found by the algorithms, *Mean* the average of the returned results in 50 runs, and *Std.dev* the standard deviation of the results (an useful criterion to show the amount of dispersion from the average value).

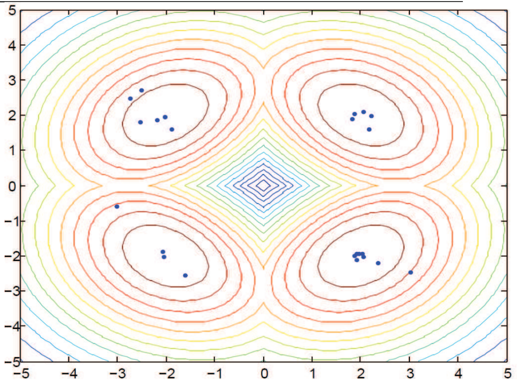
D. Statistical Comparison

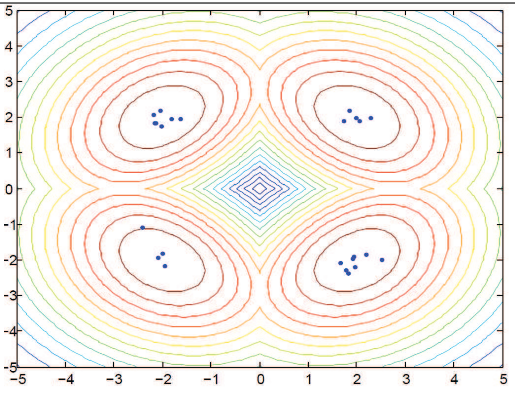
A useful statistical comparison to compare the efficiency of Meta-heuristic algorithms is the "Non-parametric Wilcoxon Rank Sum Tests" shown in Table VI. The results are presented with two parameters, *p-value* and *h*, which help find out


(a): Initialization


(b): After 4 iterations


(c): After 6 iterations


(d): After 8 iterations


(e): After 10 iterations

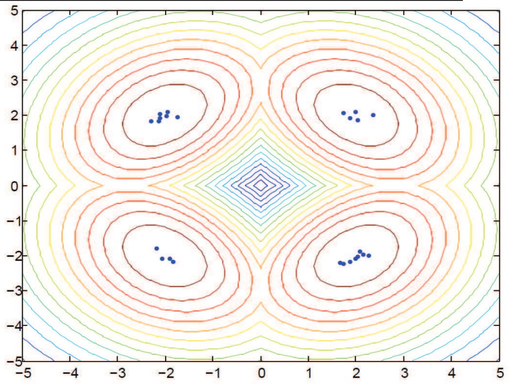

(f): After 12 iterations

Figure 4. Running steps of MRO to find the global best of the peak function

whether there is a significant difference between two algorithms or not. *h* can get three different values: 1 indicates that the performance of two algorithms is significantly different with 95% confidence, *0* that there is no statistical difference, and 1^+ that MRO has higher performance than the other algorithm (and 1^- vice versa). From Table VI, we provide the analytic results of 5 functions that we selected randomly.

- F1: MRO ranked forth. It has statistically significant performance over FA, DE, KH, PSO, BBO and COA. There is no significant difference between MRO and ISA. GSA has significant performance over MRO.
- F3: MRO ranked first. MRO has statistically significant performance over FA, DE, PSO and COA. There is no statistically significant performance between MRO, KH, BBO, GSA and ISA. BBO has statistically significant performance over MRO.
- F5: MRO ranked second. It has significant performance over FA, KH, DE, PSO, APSO and COA. There is no significant between MRO, GSA and ISA.
- F7: MRO ranked first. It has significant performance over FA, KH, DE, PSO, GSA, COA and ISA. There is no significant difference between MRO, APSO and BBO.
- F9: MRO ranked first. Statistically it has higher perfor-

Table II
High and low dimensional benchmark functions

ID	Name	Equation
		High Dimensional Function
F1	Ackley	$f(x) = -20 \times \exp\left(-0.02 \times \sqrt{n^{-1} \sum_{i=1}^{n} x_i^2}\right) - \exp\left(n^{-1} \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$
F2	Griewank	$f(x) = 1 + \frac{1}{4000} \times \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{i}\right)$
F3	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$
F4	Schwefel 2.26	$f(x) = \sum_{i=1}^{n} -x_i \times \sin(\sqrt{\lvert x_i \rvert})$
F5	Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F6	Schwefel 2.21	$f(x) = \max\{\lvert x_i \rvert, 1 \leq i \leq n\}$
F7	Schwefel 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_i\right)^2$
F8	Sphere	$f(x) = \lVert x \rVert = \sqrt{\sum_{i=1}^{n} x_i^2}$
F9	Rastrigin	$f(x) = An + \sum_{i=1}^{n} \left[x_i^2 - A \times \cos(2\pi x_i)\right], A = 10$
F10	Quatric	$f(x) = \sum_{i=1}^{n} i x_i^4 + rand$
		Low Dimensional Function
F11	Branin	$f(x) = a(x_2 - bx_1^2 + 2x_1 - d)g(1-h)\cos(x_1) + g$ $a = 1, b = 1.25\pi^{-1}, c = 5\pi^{-1}, d = 6, g = 10, h = 0.125\pi^{-1}$
F12	Camel Back-6 Hump	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1 x_2 - 4x_2^2 + 4x_2^4$
F13	De Joung (Shekel's Foxholes)	$f(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i-1}^{n} (x_i - a_{ji})^2}), where \begin{cases} a_{ij} = -32, -16, 0, 16, 32 \quad for \quad j = 1, \ldots, 5 \\ a_{1k} = a_{1j} \quad for \quad k = j+5, j+10, j+15, j+20 \\ a_{2j} = -32, -16, 0, 16, 32 \quad j = 1, 6, 11, 16, 21 \\ a_{2k} = a_{2j} \quad for \quad k = j+1, j+2, j+3, j+4 \end{cases}$
F14	Goldstein and Price	$f(x) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 - 6x_1x_2))\times$ $(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$
F15	Hartman 1	$f(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{4} a_{ij}(x_i - p_{ij})^2)$
F16	Hartman 2	$f(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_i - p_{ij})^2)$
F17	Kowalik	$f(x) = \sum_{i=1}^{11} (a_i - \frac{x_1(1 + x_2 b_i)}{(1 + x_3 b_i + x_4 b_i^2)})$
F18	Shekel 1	$f(x) = -\sum_{i=1}^{5} ((x - a_i)(x - a_i)^T)^{-1}$
F19	Shekel 2	$f(x) = -\sum_{i=1}^{7} ((x - a_i)(x - a_i)^T)^{-1}$
F20	Shekel 3	$f(x) = -\sum_{i=1}^{10} ((x - a_i)(x - a_i)^T)^{-1}$

Table III
Statistical results of MRO and nature-inspired algorithms on high dimension functions

ID	Criterion	MRO	FA	KH	DE	PSO	APSO	BBO	GSA	COA	ISA
F1	Best	0.934511	0.654211	0.999147	0.127942	0.856125	0.999999	0.992158	0.985471	0.506378	0.947621
	Mean	**0.950194**^{4}	0.395789^{8}	0.964644^{3}	0.000000^{10}	0.735129^{7}	0.971862^{2}	0.935451^{5}	0.978427^{1}	0.390429^{9}	0.871948^{6}
	Std.dev.	0.654781	0.754863	0.579421	0.987452	0.774162	0.548742	0.725797	0.245812	0.412536	0.546217
F2	Best	0.901457	0.881627	1.000000	0.000124	0.801594	1.000000	0.875429	1.000000	0.884012	0.992149
	Mean	**0.891737**^{4}	0.668126^{8}	0.998250^{1}	0.000000^{10}	0.684218^{7}	0.884371^{5}	0.859450^{6}	0.895712^{3}	0.484633^{9}	0.973418^{2}
	Std.dev.	0.754123	0.198541	0.221547	0.101268	0.558468	0.847624	0.451281	0.321751	0.501124	0.427912
F3	Best	1.000000	0.879512	1.000000	0.254128	0.995421	1.000000	1.000000	0.992147	0.000000	1.000000
	Mean	**1.000000**^{1}	0.784179^{6}	0.998696^{2}	0.243509^{7}	0.951247^{4}	1.000000^{1}	0.894202^{5}	0.978654^{3}	0.102458^{8}	1.000000^{1}
	Std.dev.	1.000000	0.612487	0.246514	0.351247	0.764851	1.000000	1.000000	0.457612	0.547891	0.945871
F4	Best	1.000000	0.887411	0.945217	0.621444	0.901247	1.000000	1.000000	0.902547	1.000000	1.000000
	Mean	**0.985989**^{4}	0.842713^{9}	0.992304^{3}	0.423967^{10}	0.890274^{7}	0.937512^{5}	1.000000^{1}	0.884288^{8}	0.901445^{6}	0.998967^{2}
	Std.dev.	0.894751	0.684527	0.754123	0.847612	0.554128	0.884571	0.642188	0.951244	0.984122	0.652874
F5	Best	0.954812	0.481225	0.945195	0.625874	0.754921	0.897452	0.658741	0.986541	0.945271	0.842711
	Mean	**0.961439**^{2}	0.778996^{7}	0.874231^{6}	0.368184^{10}	0.715843^{9}	0.882184^{4}	0.876328^{5}	0.983956^{1}	0.845712^{7}	0.945218^{3}
	Std.dev.	0.985421	0.879521	0.254158	0.984512	0.554127	0.845271	0.265841	0.985412	0.654781	0.296547
F6	Best	0.995745	0.845717	1.000000	0.754680	0.812765	0.871462	0.988473	0.751248	0.887549	0.845121
	Mean	**0.992789**^{3}	0.754812^{7}	1.000000^{1}	0.678919^{10}	0.741953^{9}	0.832814^{5}	0.901784^{4}	0.998193^{2}	0.766301^{8}	0.789129^{6}
	Std.dev.	0.351242	0.435574	1.000000	0.845219	0.351795	0.512473	0.475196	0.542189	0.536522	0.845124
F7	Best	0.855743	0.751243	0.421597	0.000000	0.674158	0.912842	0.845712	0.512887	0.451273	0.748566
	Mean	**0.801456**^{1}	0.612557^{6}	0.320151^{8}	0.000000^{10}	0.631542^{5}	0.799514^{2}	0.758749^{3}	0.541263^{7}	0.323591^{9}	0.694256^{4}
	Std.dev.	0.658741	0.847125	0.736841	0.000000	0.684219	0.421762	0.975552	0.658414	0.335478	0.845176
F8	Best	0.912482	0.884714	1.000000	0.602478	0.715482	0.932157	0.741548	0.999999	0.873274	0.924951
	Mean	**0.895124**^{4}	0.779141^{7}	1.000000^{1}	0.594219^{9}	0.687912^{8}	0.892347^{5}	0.555140^{10}	0.999843^{2}	0.809949^{6}	0.895473^{3}
	Std.dev.	0.542198	0.856412	0.254198	0.955142	0.665812	0.354199	0.863715	0.954761	0.352471	0.551241
F9	Best	1.000000	0.000000	0.855541	0.674519	0.972674	1.000000	0.921473	0.896544	0.548733	0.962438
	Mean	**1.000000**^{1}	0.000000^{9}	0.712949^{6}	0.642448^{7}	0.952347^{2}	1.000000^{1}	0.905918^{3}	0.728941^{5}	0.441889^{8}	0.818856^{4}
	Std.dev.	0.945183	0.678125	0.995124	0.884512	0.513478	0.651987	0.254961	0.555144	0.836158	0.551473
F10	Best	0.999919	1.000000	1.000000	0.899163	0.902147	1.000000	0.995618	1.000000	0.965715	1.000000
	Mean	**0.987865**^{3}	0.999998^{2}	1.000000^{1}	0.842163^{8}	0.899919^{6}	1.000000^{1}	0.969895^{5}	1.000000^{1}	0.982957^{4}	0.892471^{7}
	Std.dev.	0.876324	1.000000	0.842555	0.658411	0.351278	0.731945	0.689427	1.000000	0.668457	0.962751

mance over FA, KH, DE, GSA, COA and ISA. There is no significant performance between MRO, PSO, APSO and BBO.

We can clearly see that the results achieved by MRO are comparable with those of other nature-inspired techniques in most experiments. The high performance of MRO is mainly due to the fact that it looks for several rich areas instead of only the optimal solution. Thus, it is able to produce high quality solutions. Moreover, the interesting strategy of MRO is searching the neighbouring area of the parent mushroom in each colony. This guarantees that if there is a qualified solution in the neighbourhood of a parent, it will be found with a high probability.

VI. Constrained Optimization Problem

In this section, we deploy MRO to a well-known constrained engineering problem, namely "welded beam design" [21], used to check the capability of different meta-heuristics algorithms. This problem has a set of constrained variables that needs to be set up to minimize the fabrication cost (see Figure 5) [21]: x_1 (weld thickness), x_2 (length of the weld), x_3 (width of the

Table IV
Statistical results of MRO and nature-inspired algorithms on low dimension functions

ID	Criterion	MRO	FA	KH	DE	PSO	APSO	BBO	GSA	COA	ISA
F11	Best	0.999141	0.907451	0.892641	0.800214	0.845768	0.974197	0.748354	0.967451	0.833416	0.777124
	Mean	**0.845779**^{3}	0.666128^{8}	0.847122^{2}	0.651244^{10}	0.817945^{5}	0.821244^{4}	0.666625^{7}	0.928427^{1}	0.741556^{6}	0.657154^{9}
	Std.dev.	0.354811	0.571246	0.355471	0.913514	0.215846	0.842756	0.884754	0.711458	0.774581	0.817354
F12	Best	1.000000	1.000000	1.000000	0.200214	0.947389	0.998427	0.998999	1.000000	1.000000	1.000000
	Mean	**1.000000**^{1}	1.000000^{1}	0.993245^{3}	0.000000^{6}	0.931795^{5}	0.973419^{4}	0.997847^{2}	1.000000^{1}	1.000000^{1}	1.000000^{1}
	Std.dev.	1.000000	0.999599	0.999454	0.658412	0.517942	0.332541	0.666541	0.778415	0.998412	0.558124
F13	Best	0.992147	0.879512	0.754120	0.254128	0.928741	0.998725	1.000000	1.000000	0.000000	0.879912
	Mean	**1.000000**^{1}	1.000000^{1}	0.912455^{4}	0.249573^{8}	0.900174^{5}	0.934817^{3}	0.722495^{7}	0.956841^{2}	1.000000^{1}	0.797554^{6}
	Std.dev.	0.655144	0.844712	0.665811	0.588841	0.847362	0.537469	0.911478	0.668421	0.884921	0.745591
F14	Best	0.973145	0.941274	1.000000	0.000000	0.791846	0.937184	0.884571	1.000000	0.967314	0.977845
	Mean	**0.921427**^{4}	0.910045^{6}	0.986214^{1}	0.000000^{10}	0.837426^{7}	0.917459^{5}	0.744259^{9}	0.985989^{2}	0.801224^{8}	0.974841^{3}
	Std.dev.	0.554812	0.845752	0.657143	0.354982	0.476581	0.843712	0.542816	0.816342	0.964827	0.665417
F15	Best	0.972154	1.000000	1.000000	0.465725	0.724891	1.000000	0.785134	1.000000	1.000000	0.914753
	Mean	**0.931204**^{2}	1.000000^{1}	1.000000^{1}	0.182414^{6}	0.710048^{5}	1.000000^{1}	0.524761^{6}	0.916557^{3}	1.000000^{1}	0.887476^{4}
	Std.dev.	0.842716	0.665417	0.384965	0.756182	0.665841	0.684127	0.897561	0.826571	1.000000	0.366472
F16	Best	1.000000	0.999999	1.000000	0.779954	0.951247	0.989271	0.899754	0.903247	0.888814	0.974561
	Mean	**1.000000**^{1}	0.990479^{2}	1.000000^{1}	0.671647^{7}	0.900145^{4}	0.951473^{3}	0.799014^{7}	0.846754^{6}	0.756876^{8}	0.894721^{5}
	Std.dev.	1.000000	0.900145	0.782459	0.458163	0.652184	0.798214	0.784267	0.345812	0.751468	0.421968
F17	Best	0.951495	0.999849	0.745281	0.664791	0.890216	0.962178	0.856731	0.755142	1.000000	0.937881
	Mean	**0.928451**^{4}	0.966847^{2}	0.315287^{8}	0.421788^{9}	0.881243^{5}	0.954712^{3}	0.712538^{7}	0.457961^{8}	1.000000^{1}	0.855142^{6}
	Std.dev.	0.667812	0.882845	0.942763	0.649782	0.251974	0.358941	0.235716	0.336541	1.000000	0.316942
F18	Best	0.994512	0.888891	1.000000	0.000000	0.854732	0.945271	0.862458	0.893614	0.901754	0.941157
	Mean	**0.893548**^{3}	0.824511^{5}	0.884174^{4}	0.000000^{10}	0.792481^{6}	0.934812^{1}	0.822174^{6}	0.723541^{7}	0.684122^{9}	0.901456^{2}
	Std.dev.	0.551478	0.254761	0.631578	0.759124	0.221458	0.512798	0.458721	0.932541	0.885472	0.914527
F19	Best	0.999999	1.000000	0.996124	0.902458	0.988741	0.982154	0.893247	1.000000	1.000000	1.000000
	Mean	**0.999012**^{2}	0.999999^{1}	0.913942^{7}	0.821105^{9}	0.941278^{5}	0.921785^{6}	0.875122^{8}	0.997120^{3}	0.981342^{4}	0.999999^{1}
	Std.dev.	1.000000	0.554812	0.881451	0.658219	0.214952	0.679154	0.329731	0.731543	0.665176	1.000000
F20	Best	1.000000	1.000000	1.000000	0.000000	0.988167	1.000000	0.999999	1.000000	0.999712	1.000000
	Mean	**0.998416**^{3}	1.000000^{1}	0.991487^{6}	0.000000^{8}	0.931845^{7}	0.992745^{5}	0.998425^{2}	1.000000^{1}	0.993014^{4}	1.000000^{1}
	Std.dev.	0.362474	0.995421	0.526411	0.854122	0.479512	0.584312	0.847122	0.362474	0.892166	0.457231

Table V
Performance comparison of MRO and nature inspired algorithms on benchmark functions

	MRO	FA	KH	DE	PSO	APSO	BBO	GSA	COA	ISA
High dimension	**35**	53	27	72	64	31	31	28	63	32
Low dimension	**24**	24	32	67	54	35	47	23	35	28
$\sum$	**59**	77	59	141	118	66	76	52	95	60
Total Rank	**2**	5	2	8	7	4	4	1	6	3

Table VI
Statistical comparison of MRO and other meta-heuristic algorithms

ID	FA		KH		DE		PSO		APSO		BBO		GSA		COA		ISA	
	p.value	h	p.value	h	p.value	h	p.value	h	p.value	h	p.value	h	p.value	h	p.value	h	p.value	h
F1	6.5203 E-004	1^{+}	0.0163	1^{+}	8.0848 E-009	1^{+}	9.7194 E-008	1^{+}	0.0054	0	7.0661 E-018	1^{+}	7.0661 E-018	1^{-}	1.4357 E-004	1^{+}	0.0842	0
F2	0.0087	1^{+}	0.0403	0	2.4225 E-009	1^{+}	7.3427 E-004	1^{+}	2.4021 E-004	1^{+}	0.0872	0	0.0012	1^{+}	2.4824 E-004	1^{+}	0.0072	1^{-}
F3	4.3270 E-004	1^{+}	0.5649	0	2.3044 E-008	1^{+}	4.9247 E-003	1^{+}	0.042	0	0.0817	0	0.0867	0	3.5360 E-009	1^{+}	0.2109	0
F4	0.0061	1^{+}	0.1962	0	6.1039 E-005	1^{-}	7.8497 E-008	1^{+}	3.0842 E-005	1^{+}	0.0177	0	0.0013	1^{+}	0.0186	1^{+}	0.0068	1^{+}
F5	2.2281 E-005	1^{+}	4.6852 E-004	1^{+}	6.0170 E-006	1^{+}	4.5378 E-005	1^{+}	7.6274 E-006	1^{+}	0.0016	1^{-}	0.01251	0	0.0268	1^{+}	0.0662	0
F6	4.0016 E-004	1^{+}	0.0029	1^{-}	7.3943 E-004	1^{+}	6.9914 E-006	1^{+}	4.2437 E-003	1^{+}	0.0584	1^{+}	3.4184 E-004	1^{-}	0.0531	0	0.0385	1^{+}
F7	1.1474 E-004	1^{+}	7.6982 E-005	1^{+}	1.7588 E-014	1^{+}	1.8167 E-008	1^{+}	0.0070	0	0.0946	0	1.2485 E-004	1^{+}	1.5745 E-006	1^{+}	0.0096	1^{+}
F8	2.2281 E-004	1^{+}	9.0223 E-007	1^{-}	2.2893 E-004	1^{+}	0.5412	1^{+}	3.2198 E-006	1^{-}	0.0693	0	1.4196 E-006	1^{-}	6.3574 E-004	1^{+}	5.2794 E-006	1^{-}
F9	9.5813 E-019	1^{+}	9.0219 E-004	1^{+}	1.3578 E-005	1^{+}	0.0102	0	0.0150	0	0.0428	0	5.4572 E-004	1^{+}	2.5383 E-006	1^{+}	0.0042	1^{+}
F10	0.0460	1^{+}	4.6852 E-005	1^{+}	0.0390	1^{+}	3.1002 E-003	1^{+}	7.5713 E-005	1^{-}	0.0793	0	0.0221	0	0.0725	0	0.0041	1^{-}
F11	7.2113 E-004	1^{+}	0.2963	0	3.8984 E-004	1^{+}	0.0942	0	0.0041	0	9.6808 E-005	1^{+}	0.0244	1^{-}	0.0262	1^{+}	2.4692 E-007	1^{+}
F12	0.1962	0	0.0257	1^{+}	1.4532 E-011	1^{+}	4.8164 E-005	1^{+}	4.1051 E-004	1^{+}	0.0151	1^{+}	0.0540	0	0.0364	1^{+}	0.2482	0
F13	0.1800	0	0.0026	1^{+}	8.1706 E-011	1^{+}	7.0014 E-003	1^{+}	2.6714	1^{+}	5.1192 E-005	1^{+}	0.0782	0	0.0725	0	9.9601 E-005	1^{+}
F14	0.0919	0	0.1356	0	3.2829 E-014	1^{+}	2.8134 E-006	1^{+}	0.1032	0	9.6808 E-005	1^{+}	0.0793	0	8.6359 E-005	1^{+}	0.0854	1^{-}
F15	9.0219 E-004	1^{+}	3.3293 E-004	1^{-}	3.2739 E-005	1^{-}	9.3045 E-007	1^{+}	4.5342 E-004	1^{-}	0.0059	1^{+}	5.1845 E-004	1^{+}	5.1192 E-005	1^{-}	5.5985 E-004	1^{+}
F16	0.0044	1^{+}	0.0747	0	5.5985 E-004	1^{+}	5.4872 E-006	1^{+}	9.4176 E-003	1^{+}	4.5574 E-004	1^{+}	9.9549 E-004	1^{+}	3.6993 E-004	1^{+}	0.0018	1^{+}
F17	0.0060	1^{+}	1.756E-011	1^{+}	1.1133 E-006	1^{+}	1.8519 E-003	1^{+}	7.9174 E-003	1^{+}	8.5865 E-004	1^{+}	4.6723 E-008	1^{+}	0.0157	1^{-}	0.0064	1^{+}
F18	4.9246 E-004	1^{+}	0.0438	1^{+}	2.0246 E-011	1^{+}	1.0293 E-006	1^{+}	0.3741	0	0.0032	1^{+}	1.9984 E-004	1^{+}	7.0590 E-005	1^{-}	0.0034	1^{+}
F19	0.0049	1^{+}	0.3397	0	5.7432 E-004	1^{+}	2.7154 E-002	1^{+}	1.0154 E-003	1^{+}	3.4184 E-004	1^{+}	0.0037	1^{+}	0.0987	0	0.1060	0
F20	0.0145	1^{+}	0.0475	1^{+}	1.1018 E-013	1^{+}	6.5124 E-004	1^{+}	0.0031	0	0.1868	0	0.0053	1^{-}	0.0089	1^{-}	0.0854	0

beam) and x_4 (thickness of the beam). The objective function to be minimized is given below:

$$Cost(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \tag{11}$$

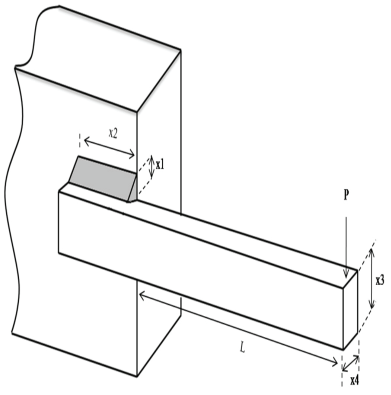


Figure 5. Schematic of the welded beam problem [21]

Table VII
The best solution of the welded beam design problem obtained by the MRO algorithm

Parameter	Best Solution
x_1	0.2447
x_2	6.2094
x_3	8.2891
x_4	0.2447
g_1(X)	-2.1814
g_2(X)	-23.4389
g_3(X)	0.0000
g_4(X)	-3.0216
g_5(X)	-0.1197
g_6(X)	-0.2342
g_7(X)	-3.5235E+003
Cost(X)	2.3828

where a solution $X = (x_1, ..., x_4)$ is subject to seven inequality constraints:

$$g_1(X) = \tau(X) - \tau_{\max} \leq 0$$
$$g_2(X) = \sigma(X) - \sigma_{\max} \leq 0$$
$$g_3(X) = x_1 - x_4 \leq 0$$
$$g_4(X) = 0.10471(x_1^2) + 0.4811 x_1 x_4 (14 + x_2) - 5.0 \leq 0$$
$$g_5(X) = 0.125 - x_1 \leq 0$$
$$g_6(X) = \delta(X) - \delta_{\max} \leq 0$$
$$g_7(X) = P - P_c(X) \leq 0$$

The constants of the design problem and their values are presented in [21]. The simple bounds of this problem are: *0.1* $\leq x_1$, $x_4 \leq$ *2.0 and 0.1* $\leq x_2$, $x_3 \leq$ *10.0*. Also, the exact optimal solution is X^*= *(0.205730 3.470489, 9.036624, 0.205729)* with the cost of *1.724852* [22]. The best solution obtained by the proposed MRO algorithm is given in Table VII together with the values of the problem constraints.

For the comparison purpose, the statistical results obtained by MRO and 14 meta-heuristic algorithms are tabulated in Table VI. The results of the 14 algorithms were taken from [21]. As we can see, MRO outperforms 7 methods and is comparable to the other 7, such as TCA and AATM.

VII. A Conceptual Comparaison

This section provides a conceptual comparison between MRO and four famous nature-inspired techniques chosen from each of the three categories: GSA (Gravitational Search Algorithm), KH (Krill Herd) and IWS (Invasive Weed Optimization).

Table VIII
Statistical results of the welded beam design problem obtained by MRO and nature-inspired algorithms

Method	Best	Mean	Std.dev	No. Evaluation
SBS	2.4426	N/A	N/A	20,000
TCA	2.3811	2.7104	9.31E-02	320,000
GA-AP	2.3814	5.9480	9.09E-01	320,000
GA-AIS	2.3812	2.3899	N/A	320,000
GA-AIS	2.3834	4.0560	2.02E-01	320,000
GA	2.4331	N/A	N/A	40,080
GA-PLP	2.3812	N/A	N/A	320,080
GA-AP	2.3816	2.4172	N/A	320,000
MOEA	2.3829	2.4209	2.56E-02	80,000
BFO	2.3868	2.4040	1.60E-02	48,000
SCA	2.3854	3.2550	9.60E-01	33,095
GA-SR	2.5961	10.1833	1.29E+0	320,000
AATM	2.3823	2.3870	2.20E-3	30,000
ISA	2.3812	2.4973	1.02E-01	30,000
MRO	2.3828	4.4817	2.68E-01	188,000

A. MRO versus GSA

MRO and GSA use different strategies in searching the problem space. They can be compared as follows:

1) MRO is a nature-inspired algorithm whereas GSA is inspired by a physical phenomenon.
2) The movement direction of an search agent in GSA is affected by other agents and is determined based on the overall force obtained by all the agents. But the movement of the search agents in MRO is highly influenced by the blowing wind and richness of the living areas (including other colonies' areas).
3) In MRO, the searching process is mainly performed in the rich areas. Similarly, search agents in GSA consider all the potential solutions in their neighbouring areas. This is due to the fact that the force equation of GSA is proportional to the quality of the found solutions.
4) The searching process in MRO is not impacted by the distance between the solutions, which is in contrast to GSA wherein the force is reversely proportional to the distance between the solutions.

B. MRO versus KH

Due to the difference of the inspiration sources (with different governing rules), MRO and KH follow different optimization strategies. Some of the similarities and differences are listed below:

1) The inspiration source of KH algorithm is the herding behaviour of krill individuals while MRO is inspired by the mushroom reproduction model.
2) The movement direction of the agents in KH is influenced by two main goals that are obtaining food and increasing the density of krill individuals whereas in MRO, the movement direction of the agents is affected by the direction of the wind and richness of the neighbouring areas of the agents.
3) The updating process in KH is performed by considering the distance between the individuals and the food source.

A krill individual tries to keep itself within a minimum distance to the highest krill density and food. So, the closer the distance to the high density and food, the higher fitness value it possesses. In contrast, the distance between individuals in MRO is not needed in the updating process.

C. *MRO versus IWO*

The similarities between mushrooms and weeds are twofold: a) MRO produces and releases spores and IWO seeds, and b) both are not affected by the distance between agents. Although they may look similar, they differ in several ways as explained below:

1) MRO mimics the way mushrooms reproduce and migrate to rich areas with nourishing sources and adequate light. IWO on the other hand simulates the invasive behaviour of weeds in nature.
2) MRO and IWO are quite different in the way they search the problem space. They perform different movement strategies through different equations. MRO movement is carried out based on the virtual wind and other parameters. IWO distributes seeds randomly through the problem space by generating normally distributed random numbers with mean equal to zero and varying variance.
3) The direction of movement in IWO completely depends on certain numbers produced randomly while in MRO the direction of spore migration is determined by the direction of the wind and richness of the areas in the surrounding environment.
4) Different factors are updated in MRO and IWO processes. MRO updates information about the best colony, average of each colony and total average of all colonies in each iteration. In each step of IWO, the information about the cost of the best and worst solutions and the standard deviation are updated.

VIII. Conclusion and Future Work

Mushrooms reproduce by making spores and distributing them randomly to different places in the environment. In our proposed MRO, the direction of spore migration is determined by the artificial mechanism wind as well as the richness of the surrounding areas. The process of finding the richest area by the search agents, which contains high-quality solutions and probably the optimal solution, is the basis for developing our MRO algorithm. The most significant contributions of the MRO algorithm are the strategy for searching the local solutions and the strategy for identifying the richest area that contains the optimal solution. Through uni and multimodal benchmark functions as well as engineering problem instances, we have examined the performance of MRO by comparing it with several known meta-heuristic algorithms using the Null Hypothesis Significance Testing. According to the experimental results, MRO was superior to 9 nature-inspired techniques, and was comparable to the Krill Herd algorithm. It is very encouraging to find out that MRO performs so effectively with different optimization problems and is superior to most of the well-known meta-heuristic algorithms.

References

[1] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[2] R. Abbasian, M. Mouhoub, and A. Jula, “Solving graph coloring problems using cultural algorithms,” in *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, 2011, pp. 3–8.

[3] S. K. Shil, M. Mouhoub, and S. Sadaoui, “An approach to solve winner determination in combinatorial reverse auctions using genetic algorithms,” in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. ACM, 2013, pp. 75–76.

[4] X.-S. Yang, *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[5] W. Du and B. Li, “Multi-strategy ensemble particle swarm optimization for dynamic optimization,” *Information sciences*, vol. 178, no. 15, pp. 3096–3109, 2008.

[6] Z. Baojiang and L. Shiyong, “Ant colony optimization algorithm and its application to neuro-fuzzy controller design,” *Journal of Systems Engineering and Electronics*, vol. 18, no. 3, pp. 603–610, 2007.

[7] C.-C. Wu, K.-C. Lai, and R.-Y. Sun, “Ga-based job scheduling strategies for fault tolerant grid systems,” in *Asia-Pacific Services Computing Conference, 2008. APSCC’08. IEEE*. IEEE, 2008, pp. 27–32.

[8] M. Mouhoub and Z. Wang, “Improving the ant colony optimization algorithm for the quadratic assignment problem,” in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 250–257.

[9] ——, “Ant colony with stochastic local search for the quadratic assignment problem,” in *Tools with Artificial Intelligence, 2006. ICTAI’06. 18th IEEE International Conference on*. IEEE, 2006, pp. 127–131.

[10] H. R. Kanan and K. Faez, “An improved feature selection method based on ant colony optimization (aco) evaluated on face recognition system,” *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 716–725, 2008.

[11] H. R. Kanan and B. Nazeri, “A novel image steganography scheme with high embedding capacity and tunable visual image quality based on a genetic algorithm,” *Expert Systems with Applications*, vol. 41, no. 14, pp. 6123–6130, 2014.

[12] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister, “A brief review of nature-inspired algorithms for optimization,” *arXiv preprint arXiv:1307.4186*, 2013.

[13] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[14] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[15] P. Stamets, *Growing gourmet and medicinal mushrooms*. Ten Speed Press, 2011.

[16] X.-S. Yang, “Firefly algorithm, levy flights and global optimization,” in *Research and development in intelligent systems XXVI*. Springer, 2010, pp. 209–218.

[17] C.-P. Fung, “Manufacturing process optimization for wear property of fiber-reinforced polybutylene terephthalate composites with grey relational analysis,” *Wear*, vol. 254, no. 3, pp. 298–306, 2003.

[18] I. Fister, X.-S. Yang, and J. Brest, “A comprehensive review of firefly algorithms,” *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[19] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, “A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems,” *Journal of Global Optimization*, vol. 31, no. 4, pp. 635–672, 2005.

[20] E. P. Adorio and U. Diliman, “Mvf-multivariate test functions library in c for unconstrained global optimization,” *Quezon City, Metro Manila, Philippines*, 2005.

[21] A. H. Gandomi, “Interior search algorithm (isa): a novel approach for global optimization,” *ISA transactions*, vol. 53, no. 4, pp. 1168–1183, 2014.

[22] R. V. Rao, V. J. Savsani, and D. Vakharia, “Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems,” *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.