Annals of Operations Research
https://doi.org/10.1007/s10479-024-05992-9

ORIGINAL RESEARCH

Enhanced migrating birds optimization algorithm for optimization problems in different domains

Ramazan Algin[1] · **Ali Fuat Alkaya**[1,2] · **Mustafa Agaoglu**[1]

Received: 18 April 2023 / Accepted: 8 April 2024

Abstract
Migrating birds optimization algorithm is a promising metaheuristic algorithm recently introduced to the optimization community. In this study, we propose a superior version of the migrating birds optimization algorithm by hybridizing it with the simulated annealing algorithm which is one of the most popular metaheuristics. The new algorithm, called MBOx, is compared with the original migrating birds optimization and four well-known metaheuristics, including the simulated annealing, differential evolution, genetic algorithm and recently proposed harris hawks optimization algorithm. The extensive experiments are conducted on problem instances from both discrete and continuous domains; feature selection problem, obstacle neutralization problem, quadratic assignment problem and continuous functions. On problems from discrete domain, MBOx outperforms the original MBO and others by up to 20.99%. On the continuous functions, it is observed that MBOx does not lead the competition but takes the second position. As a result, MBOx provides a significant performance improvement and therefore, it is a promising solver for computational optimization problems.

Keywords Metaheuristics · Migrating birds optimization · Feature selection · Quadratic assignment problem · Obstacle neutralization problem · Continuous functions

1 Introduction

Metaheuristics are commonly used solution techniques for combinatorial optimization problems. They are preferred to exact algorithms when near optimum solutions are sought on large problem instances. However, for instances of high complexity or large-scale problems, heuristics or metaheuristics may not be sufficient to achieve satisfactory results. For this reason, especially during the last two decades, researchers have been trying to find new techniques that provide better performance. One of the fields that researchers are focusing on is

✉ Ramazan Algin
ramazan.algin@marmara.edu.tr

Ali Fuat Alkaya
alifuat.alkaya@nottingham.ac.uk

Mustafa Agaoglu
agaoglu@marmara.edu.tr

[1] Computer Engineering Department, Marmara University, Istanbul 34840, Turkey

[2] School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK

Published online: 21 May 2024

hybridizing metaheuristics. Hybrid metaheuristics are generally obtained by combining the power of two or more metaheuristics or by placing a local search heuristic within a metaheuristic. There are several studies in the literature that successfully hybridize metaheuristics. The most preferred technique is hybridizing genetic algorithms with a local search procedure or some other metaheuristics (Drezner, 2008; Gonçalves et al., 2005; Kao and Zahara, 2008). Apart from genetic algorithm-focused studies, hybridization of other meta-features such as simulated annealing with ant colony optimization (Behnamian et al., 2009), simulated annealing with differential evolution (Liao et al., 2014), harmony search algorithm with differential evolution (Duan et al., 2013) are observed in the literature.

In this study, we have studied on placing a different exploration strategy into the migrating birds optimization (MBO) algorithm which is proposed recently. The exploration strategy built into MBO is the stochastic motion tactic that the MBO algorithm is expected to benefit (exploit) from a wider range of solutions.

V formation of the migrating birds in real life is the inspiration of the MBO algorithm (Duman et al., 2012). The algorithm starts with randomly initializing the feasible solutions (represent birds in the analogy) in the solution space and by searching their neighborhood these feasible solutions try to move to better positions. Solutions are placed in a hypothetical V formation and throughout the algorithm they share their unused neighbors with the follower solutions. In the original MBO study, the authors test the performance of the MBO algorithm by comparing it with other metaheuristics on quadratic assignment problem instances. Results of the experiments show that MBO has competitive performance with the simulated annealing and better performance than the differential evolution algorithms, tabu search, scatter search, guided evolutionary simulated annealing, genetic algorithm, and particle swarm optimization (Duman et al., 2012).

MBO algorithm is proven to be a good performing algorithm and thanks to its swarm structure and benefit mechanism among the feasible solutions, it has the chance to find the global minima. Nevertheless, it has a drawback; solutions always move to better feasible solutions causing the algorithm to get stuck in local minima. In order to avoid getting stuck at local minima, embedding a new exploration strategy in MBO is a novel and promising idea. This superior version of MBO is called MBOx throughout the manuscript. In this study, we showed the superiority of MBOx over MBO and four other well-known metaheuristics; genetic algorithms (GA), differential evolution (DE), simulated annealing (SA) and harris hawks optimization (HHO) on four different problem sets (three of which are NP-Hard): (i) feature selection problem, (ii) quadratic assignment problem, (iii) obstacle neutralization problem, (iv) well-known continuous functions, through conducting extensive computational experiments. Therefore, we believe that MBOx will take its place as a promising problem solver for optimization problems.

The manuscript is arranged as follows. Brief information about the MBO algorithm, benchmark algorithms, tackled problems and previous work are given in Sect. 2. Details of the MBOx algorithm is discussed in Sect. 3. Implementation details of algorithms, computational experiments and discussion are given in Sect. 4. Section 5 concludes the paper with some future work.

2 MBO, benchmark algorithms and previous work

In this section, we give some information about MBO and benchmark algorithms with their summarized literature reviews.

2.1 MBO and related previous work

MBO algorithm is one of the recently proposed swarm intelligence techniques that is inspired from the migrating birds in real life and their V-shape formation. In this subsection, we prefer to stick to the metaphor based terminology used in the original MBO paper. However, rather than using specifics words belong to migrating birds, general optimization terminology will be used in the remaining of the manuscript to make MBOx easily comparable with other algorithms.

In the MBO algorithm, there is a leader bird which is chosen randomly among all birds, whereas the remaining birds are divided into two groups (both groups have same number of birds) behind the leader bird as in a V-shape formation. Every bird in the flock generates predetermined number of feasible solutions that determines the speed of the flock (in the remainder of the manuscript, instead of using "feasible solution" we prefer to use "solution"). The speed of flock determines the search area, where the flock can do search in a wider area if its speed is higher.

The working principles of the algorithm are as follows. Firstly, initial solutions are generated randomly by placing the birds to the search space in a hypothetical V-shape formation. After the initialization phase, leader bird generates its neighbours, selects the best of these and then replaces if this is better than itself. Then, the leader bird gives best unused solutions to the bird immediately behind. Remaining birds also share their neighbors in the same way as the leading bird does. Once all the birds share their unused neighbors with the following birds, one flapping process is completed. After m flappings, the leader bird is moved to the last position of one of the tails and the process starts over. Total number of iterations or number of neighbors generated is used as the stopping criteria. Figure 1 presents the structure of the MBO algorithm where the parameter definitions are as follows:

k = the number of neighbor solutions to be considered
m = number of tours
n = the number of initial solutions
x = the number of neighbor solutions to be shared with the next solution

Duman et al. (2012) introduced the MBO algorithm to the literature in where performance of the MBO algorithm is tested on QAP instances and also compared with other well-known metaheuristics. According to the results, MBO has competitive performance with the simulated annealing and outperforms differential evolution algorithms, tabu search, scatter search, guided evolutionary simulated annealing, genetic algorithm, and particle swarm optimization.

In the literature, there are several applications of MBO to several problems. In Alkaya and Algin (2015), ant system (AS), GA, SA and MBO algorithms are applied to the obstacle neutralization problem (ONP). Among these metaheuristic algorithms, MBO and SA give competitive results and outperform AS and GA. In that study, as a problem, the authors focused on just one problem (ONP). Whereas in our study, four different problems are tackled including ONP and for ONP, it is shown that MBOx algorithm outperforms all other metaheuristics mentioned in Alkaya and Algin (2015). Therefore, our study also provides a novel contribution to the ONP literature. In another interesting study, neighbor generation method of the MBO is modified and so MBO is applied to 30 different functions on continuous domain (Alkaya et al., 2014). The contribution of that study is developing an effective and adaptive neighbor generation function for the MBO. The tests are conducted on continuous functions with different dimension values (2, 10 and 30).

In another study, an improved version of MBO algorithm is introduced to the literature where it is used to minimize the total flow time for a hybrid flow shop scheduling problem,

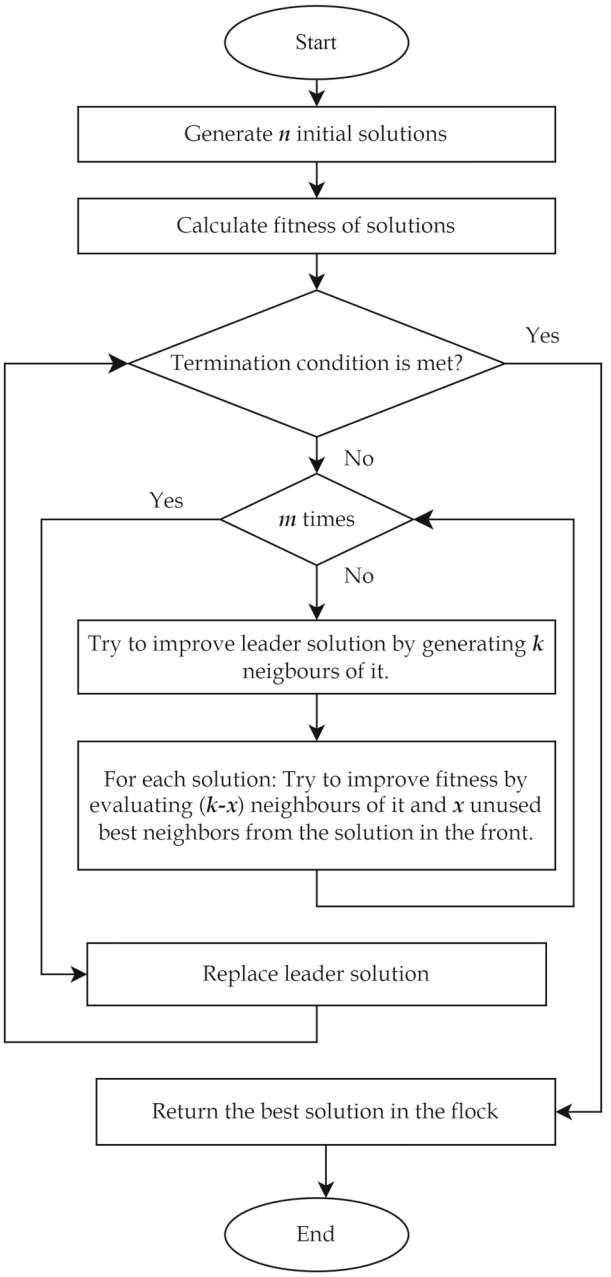


Fig. 1 The structure of the MBO

which has important practical applications in modern industry. In that study, authors proposed many effective and advanced technologies to improve the MBO algorithm, including a leaping mechanism, a diversified initialisation method, a local search procedure, a mixed neighbourhood structure and a problem specific heuristic (Pan and Dong, 2014). Similarly, in Benkalai et al. (2017), the authors use MBO for solving permutation flow shop problem with sequence-dependent set-up times. They modified the basic MBO by changing neighbourhood function and generating leader bird using ad-hoc heuristic.

MBO algorithm is also used to solve credit card fraud detection problem (Duman and Elikucuk, 2013). In that study, original MBO algorithm is improved by using some benefit mechanism. Improved MBO, genetic algorithm with scatter search (GASS) and MBO algorithms are compared. According to the experiment results GASS is outperformed by both improved MBO and MBO.

Recently, MBO is applied to the feature selection problem where it is compared with some metaheuristic approaches and it is shown that MBO outperforms others (Algin et al., 2020; Kalayci et al., 2019).

In addition to these studies, there are many studies where the authors tried to enhance the MBO algorithm. In Oz (2017), MBO is improved by designing problem specific neighbor function for the multi-objective task allocation problem. This new neighboring function allows to perform both exploration and exploitation. In another study Sioud and Gagné (2018), MBO algorithm is enhanced by developing an adapted neighborhood search based on a tabu list, an original leader selection process, swap and forward insertion moves, and a restart mechanism. In some of studies (Segredo et al., 2018; Tongur & Ülker, 2018) the MBO is hybridized with some other metaheuristics like particle swarm optimization and differential evolution. It is shown in these studies that hybridization of MBO with other metaheuristics increase the performance of the MBO.

The results of a set of preliminary tests of MBOx are presented in Algin et al. (2018) where MBOx is tested on QAP and compared only with MBO algorithm. However, as given in the introduction section, in our study, we present the performance of MBOx on four well-known problems with extended datasets and compare it with greater number of algorithms. Therefore, our study contains a significant contribution and extension over (Algin et al., 2018).

2.2 Benchmark algorithms

As benchmark algorithms we use GA, SA, DE and HHO. In this subsection, we shortly describe these benchmark algorithms with their related literature review. Implementation details of these algorithms are given in Sect. 4.1.

2.2.1 Genetic algorithms (GA)

One of the most popular metaheuristic used in many problems from various application areas is GA which is inspired from the principles of natural evolution (Holland, 1986). GA is a population based algorithm and each solution (individual) is represented as a list of genes, therefore a solution is also referred to as chromosome. In GA, in order to produce better offsprings the individuals that have better fitness values are more likely to be chosen to undergo reproduction (Beasley et al., 1993).

GA is and old algorithm, therefore, there are lots of studies in the literature. Here we summarize some of the recently published studies related to the GA. In Sohail (2023), success of

GA when it is applied to the multi-dimensional problems in the fields of artificial intelligence and data sciences is discussed. It is also mentioned that GA improves the performance of the artificial intelligence tools such as classification, forecasting and optimization tools. In another study, non dominant sorting GA algorithm is proposed to solve a multi-objective problem called sustainable capacitated facility location/network design problem (Brahami et al., 2022). Similarly in Deng et al. (2022), enhanced version of non dominant sorting GA algorithm is proposed and applied to multi-objective problems.

Gen and Lin (2023) proposed a survey study related to GA and its applications. Firstly, hybrid genetic algorithms and adaptive genetic algorithms are mentioned in the study and then applications of GA on combinatorial optimization problems, network design problems, scheduling problems, reliability design problem, logistic problems, location and allocation problems are explained.

2.2.2 Simulated annealing (SA)

SA, proposed by Kirkpatrick et al. (1983) mimics the cooling and annealing of the metals and it can be said to be the oldest among the metaheuristics. When there are many local optima in the search space, SA can be used to find global optimum. SA has an explicit strategy to escape from local minima. The algorithm is similar to hill-climbing but with some randomness. If the selected move improves the quality of solution, then it is accepted. If the selected move is worse than current move, there is still chance to accept selected move with a probability which helps to escape from the local minima. During the search process, the probability is decreased.

Kosanoglu et al. (2022) developed a hybrid algorithm (DRLSA) by combining double deep q-network based deep reinforcement learning and SA algorithms. DRLSA applied to a joint maintenance planning problem where spare decisions of parts inventory management, workforce training, and workforce planning are considered simultaneously. Another hybrid algorithm is proposed by Liu et al. (2022) where SA algorithm is combined with shuffled frog leaping algorithm. It is applied to the continuous functions and feature selection problems. Fontes et al. (2023) proposed a hybrid particle swarm optimization and SA for the job shop scheduling problem with transport resources. In another study, SA is hybridized with the artificial algae algorithm to solve a location routing problem with twp dimensional loading constraints (Ferreira and de Queiroz, 2022). In order to increase the exploration and exploitation capacity of grasshopper optimization algorithm, it is combined with SA algorithm (Yu et al., 2022). It is applied to several engineering problems and parameter optimization of the kernel extreme learning machine problems.

2.2.3 Differential evolution (DE)

DE is one of the latest evolutionary optimization methods proposed by Storn and Price (1997). DE is a stochastic, population-based optimization algorithm. DE tries to optimize D dimensional parameter vectors, also called solutions, in a population through generations by using mutation and crossover operators.

Ahmad et al. (2022) proposed a survey study about the state-of-the-art works related to the DE algorithm. In that study, modifications on the DE to increase the efficiency and effectiveness of the algorithm, different DE variants, analysis of different parameter settings on the DE variants, hybridization of DE and recent applications of DE variants are explained in details. In Song et al. (2023), a cooperative co-evolutionary DE algorithm combined with

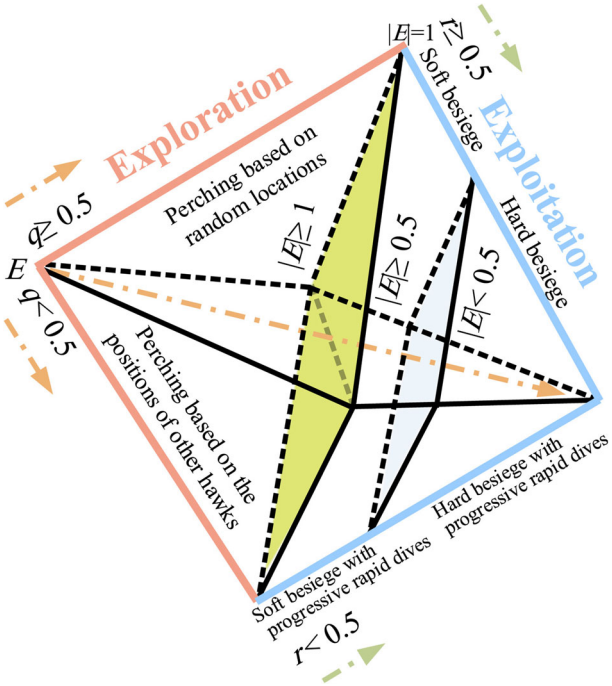


Fig. 2 Phases of HHO (Heidari et al., 2019)

GA and quantum evolutionary algorithm is designed and applied to train delay scheduling problem. In another study, a hybrid adaptive DE algorithm is used to solve multi-objective fuzzy job-shop scheduling problem (Wang et al., 2022). A survey related to the balancing the exploration and exploitation ability of DE algorithm is proposed recently by Zhang et al. (2023). In that study, recent works on DE from 2019 to 2023 are summarized and discussed about the exploration/exploitation trade-off in term of algorithm level, the operator level and the parameter level.

2.2.4 Harris hawks optimization (HHO)

HHO is one of the recently published population-based and gradient free optimization methods proposed by Heidari et al. (2019). HHO is inspired from the nature where swarm of hawks collaborate during the hunting and a prey tries to escape from hawk attacks. Different phases of HHO algorithm is shown in Fig. 2. There are three main phases of the HHO: (i) exploration phase, (ii) transition from exploration to exploitation and (iii) exploitation phase. The exploration phase consist of two strategies. In the first strategy, hawks perch based on the other hawks' positions in the swarm and the prey, whereas in the second strategy, hawks perch on random tall trees. Transition from exploration to exploitation is performed according to the escaping energy of the prey. When the energy$\geq$ 1 exploration happens when energy$<$ 1 exploitation happens. In the exploitation phase, according to the chasing strategies of hawks and escaping behaviors of the prey there are four strategies: (i) soft besiege, (ii) hard besiege, (iii) soft besiege with progressive rapid dives and iv) hard besiege with progressive rapid

dives. Performance of HHO is measured on continuous functions and compared with other metaheuristics.

Although HHO is recently published, due to its good performance there are plenty of studies about it in the literature. In Alabool et al. (2021), a comprehensive review of recent variants and applications of HHO is given. In another study, improved version of HHO with simulated annealing is proposed for feature selection problem (Elgamal et al., 2020). In that study, in order to enhance the population diversity chaotic maps are used at the initialization phase of HHO and in the exploitation phase of HHO SA algorithm is used to avoid stuck in local minima. Performance comparison of improved HHO with other metaheuristics are performed on the medical benchmark datasets. Similarly in Zhang et al. (2021), HHO is improved by embedding the salp swarm algorithm and applied to some continuous functions and the feature selection problem.

Multi objective version of HHO (MOHHO) is proposed in Zouache et al. (2023). MOHHO uses the strengthened dominance relation to select the solutions with better convergence and diversity balance. MOHHO uses the rabbit solutions to converge to better area of the search space. Five bi-objective and seven three objective test functions are used to measure the performance of the MOHHO and it is compared with three multi objective metaheuristics. According to the experiment results, MOHHO outperforms others. In order to overcome the low exploration of HHO, novice protection tournament based HHO is proposed in Li et al. (2023) and applied to 23 continuous functions and several engineering problems. In another similar study (Abualigah et al., 2023), two search strategies (sine and cosine functions) are added to the HHO. Converge speed of the HHO is improved by adding the sine function whereas cosine function is used to improve the ability of the exploration and exploitation searches of the HHO. Its performance is tested on 23 continuous functions and several engineering problems.

2.3 Tackled problems

In this study, we tackled four different problems; (i) feature selection problem (FS), (ii) quadratic assignment problem (QAP), (iii) obstacle neutralization problem (ONP), (iv) well-known continuous functions (CFs). The motivation of choosing these problem domains is to show the capability of MBOx in a cross domain test environment. Specifically, we chose the CF domain because MBO was originally proposed for discrete problems and its performance on CFs is unrevealed up to now. ONP is selected because MBO was the best performing algorithm in the introductory papers of ONP. FS and QAP are selected for being popular and challenging. In addition, they have different structures/natures in the sense that they all have different neighbor generation mechanisms. This section provides the details and a discussion of existing literature about the considered problems.

2.3.1 Feature selection (FS) problem

Feature Selection (FS) is a very important task in the machine learning area. It is used to reduce the size of the data by removing irrelevant or redundant features and increase the performance of algorithms by reducing the dimension (number of feature). In today's computing world, huge data is a reality and this causes many problems like high storage, low performance, etc. At that point, to solve these kinds of problems FS must be applied. By selecting the most important features in the dataset, FS reduces the dataset size and also improves the accuracy of algorithms.

In general, feature selection methods are divided into three categories: (i) filter methods, (ii) wrapper methods, and (iii) embedded methods. In filter method, evaluation of subsets is performed by subset evaluators, whereas in wrapper methods, it is done by using classifiers. Due to higher complexity of wrapper methods they are slower than filter methods where evaluation is performed faster. Embedded methods are similar to wrapper methods, however, feature selection is performed during training process. FS is in the NP-Hard problem domain. Therefore, heuristics and metaheuristics are mostly preferred to solve the FS problem.

In the literature, there are many studies where metaheuristics are applied to the FS problem, however here we summarized only a few of them. In Algin et al. (2020), four metaheuristics are applied to solve the FS problem where migrating birds optimization algorithm (MBO) is applied to the FS problem for the first time and shown that MBO algorithm outperforms other metaheuristics in term of accuracy values. Similarly, genetic algorithm (Oreski and Oreski, 2014), particle swarm optimization (Wang et al., 2007) and simulated annealing (Debuse and Rayward-Smith, 1997) algorithms have been proposed for the FS problem. In another study Diao and Shen (2015), comprehensive review of ten nature inspired metaheuristics for the FS problem is provided.

2.3.2 Quadratic assignment problem (QAP)

QAP is a well-known combinatorial optimization problem and also one of the most difficult problems to solve optimally. The interpretation of the QAP can be explained simply by assigning offices to people (Hanan and Kurtzberg, 1972). In this problem, the affinity between person i and person j is c_{ij} and their cost matrix is $CM = [c_{ij}]$. The people are assigned to N number of possible offices. The distance between office e and office g is shown as d_{eg} and their distance matrix is $DM = [d_{eg}]$. The assignment cost of person i assigned to office $p(i)$ and person j assigned to office $p(j)$ is shown as $c_{ij}d_{p(i)p(j)}$. The cost of all office assignments can be calculated by sum of each assignment costs over all people. The aim in this problem is minimizing the total assignment cost. A mathematical formulation of the QAP can be given as follows:

$$min \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{e=1}^{N}\sum_{g=1}^{N} c_{ij}d_{eg}y_{ie}y_{jg} \tag{1}$$

s.t.

$$\sum_{i=1}^{N} y_{ie} = 1, \qquad e = 1, ..., N \tag{2}$$

$$\sum_{e=1}^{N} y_{ie} = 1, \qquad i = 1, ..., N \tag{3}$$

$$y_{ie} \in \{0, 1\}, \qquad i, e = 1, ..., N \tag{4}$$

where y_{ie} is the binary variable that states the assignment of person i to office e.

There are so many studies related with the QAP in the literature. The QAPLIB website stores the latest studies on the QAP as well as the QAP instances that researchers are continuously working on QAPLIB (1997). In Drezner (2008), the author tried to solve the QAP using different variants of hybrid genetic algorithm. The author compares the simple tabu search and the modified robust tabu search as local optimization algorithms combined with a crossover operator. Seven modifications of the basic hybrid genetic algorithm are used on

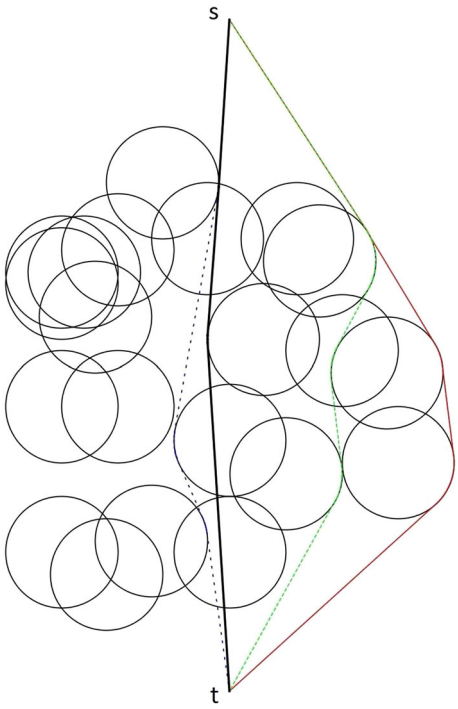


Fig. 3 An example to the ONP and optimal paths for K=0, 1, 2 and 3 Alkaya and Algin (2015)

the experiments. In another study, an improved hybrid genetic algorithm (IHGA) is used to solve the QAP. In the IHGA, iterated local search technique and tabu search are combined, called limited iterated tabu search (LITS), and used as local improvement procedure. For the comparison on the QAP, fast ant system, genetic hybrid algorithm, and robust tabu search are used. Among these algorithms, IHGA outperformed all algorithms and also the proposed algorithm gets better solutions than previous studies on the literature (Misevicius, 2004). Robust tabu search is an important contribution to solve the QAP with fewer parameters and less complexity and still it is being improved (Paul, 2011; Taillard, 1991, 1995).

2.3.3 Obstacle neutralization problem (ONP)

ONP is a kind of constrained shortest path problem where an agent tries to reach to a destination point swiftly and safely from a given source point through an arrangement of disc-shaped obstacles in the plane. Due to the agent's payload capacity, (s)he has limited number of neutralization capability, say by K. When the agent neutralizes a disc, (s)he can enter the disc and the neutralization cost is added to the traversal length of the path.

Mathematically, an ONP instance is a tuple $(\mathcal{A}, s, t, K, c,)$, where $\mathcal{A}$ represents a finite set of open discs in $\mathbb{R}^2$, s is the start point and t is the target (terminal point) in $\mathbb{R}^2$, K is a given constant in $\mathbb{N}$ and c is a cost function mapped to $\mathbb{R}_{\geq 0}$. In this problem, the goal is taking the agent from s to t safely through the shortest path.

An instance of ONP is provided in Fig. 3. In this instance, neutralization cost is 1 and radius of each disc is set to 5. When the agent has zero neutralization capability (K=0), then (s)he chooses red (solid) path. Similarly, for the values of K = 1, 2, 3; green (dotted), blue (dashed), and black (bold solid) paths are the optimum paths, respectively.

ONP is introduced to the literature in Alkaya et al. (2015) where a heuristic approach is proposed to solve the ONP. The proposed approach is called penalty search algorithm (PSA). In the PSA, largest penalty term α^* is found in the unconstrained shortest path where it has the highest number of neutralizations and also satisfies the neutralization limit. It is shown that PSA can find the optimum paths in special cases: all discs have equal neutralization cost and radii. However, in many cases this may not be realistic.

In another study related to ONP, an exact algorithm is developed (Alkaya and Oz, 2017). There are two phases in the exact algorithm. In the first phase, PSA is used to find an upper bound for the ONP. In the next phase, if the upper bound is not optimal solution, starting from upper bound a kth shortest path algorithm is used to obtain the optimal solution. Both grid and continuous graphs are used to test the performance of the exact algorithm. Experiment results show that the algorithm works very fast on moderate and small sized graphs. Since the exact algorithm is based on the PSA, the cases mentioned in the PSA are required.

In Algin et al. (2013), an ant system algorithm is developed where problem specific information is used in the state transition rule to guide the ants. In that study, in order to improve the performance of the algorithm, importance of parameter fine tuning of an algorithm is showed. In the experiments, a real world naval minefield dataset is used. However, in their study, without performing any metaheuristic comparison, they only present the algorithm developed for the ONP and the results of the computational experiments. In Algin and Alkaya (2015), MBO is compared with ant system and ant colony system algorithms on the ONP and it is shown that MBO outperforms other two algorithms. In a recent study, genetic algorithm, migrating birds optimization, simulated annealing and ant system algorithms are exploited to solve the ONP (Alkaya and Algin, 2015). Performance of metaheuristic algorithms are tested on the ONP instances and compared with the optimum solution obtained using an exact algorithm.

2.3.4 Continuous optimization problems

In addition to FS, ONP and QAP, we also tackled 30 well-known continuous functions including Rosenbrock's, Weierstrass, Ackley's and Schaffer's functions. These functions are well known in the sense that they are used as benchmark problems for assessing the performance of the optimization algorithms (Alkaya et al., 2014). Four of those functions can be seen in Fig. 4 and their equations are given below. Equations of 30 continuous functions are given in the Appendix A.

- Rosenbrock's function:

$$f(x) = \sum_{i=1}^{D-1} (100\,(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{5}$$

- Weierstrass function:

$$f(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{20} [0.5^k cos(2\pi.3^k(x_i + 0.5))] \right) - D \sum_{k=0}^{20} [0.5^k cos(2\pi.3^k.0.5)] \tag{6}$$

- Ackley's function:

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D}cos(2\pi x_i)\right) + 20 + e \tag{7}$$

- Schaffer's F7 function:

$$f(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}(x_i^2 + x_{i+1}^2)^{1/4}(x_i^2 + x_{i+1}^2)^{1/4} \sin^2(50(x_i^2 + x_{i+1}^2)^{0.1})\right) \tag{8}$$

In the next section, we provide the details of the proposed MBOx algorithm.

3 MBO with a new exploration strategy (MBOx)

In this section, we present the details of the proposed hybrid MBO, namely MBOx. MBOx algorithm is obtained by embedding a new exploration strategy and its related parameters into the MBO.

MBO is proven to be a good performing algorithm and thanks to its swarm structure and benefit mechanism among the solutions it has the chance to find the global minima. Nevertheless, it has a drawback; it always moves to better solutions causing the algorithm to get stuck in local minima.

We want to note that even though the original MBO algorithm is defined with a metaphor using migrating birds, in our study, we prefer to use a metaphor-free description of the MBOx algorithm written according to Sörensen (2015). Therefore, the word "solution" will be used throughout the manuscript.

Modification of MBO's exploration policy is a promising and novel idea which avoids getting stuck at local minima. Hence, in order to increase the probability of finding global optimum by moving to worse solutions, we embedded a stochastic move strategy in MBO algorithm. The strategy is; the best solution, z', in the neighbour set of a solution (z) is accepted as new solution depending on $f(z)$, $f(z')$ and T where f is the fitness evaluation function. z' replaces z if $f(z') < f(z)$ or, in case $f(z') \geq f(z)$, with a probability which is a function of T and $f(z') - f(z)$. We calculate the probability using the Boltzmann distribution. Mathematical formulation is given in Eq. 9.

$$z \Leftarrow \begin{cases} z', \text{ if } f(z') < f(z) \\ z', \text{ else if } random(0,1) < \exp(\dfrac{-\|f(z') - f(z)\|}{T}) \end{cases} \tag{9}$$

In this way, we expect to enhance the exploration capability of the MBO algorithm. In the implementation, the best solution found throughout the algorithm is traced and its fitness is reported as the output.

The structure of the MBOx algorithm is shown in Fig. 5. In addition to the parameters of MBO, MBOx has three more parameters:

- a: temperature decrease ratio

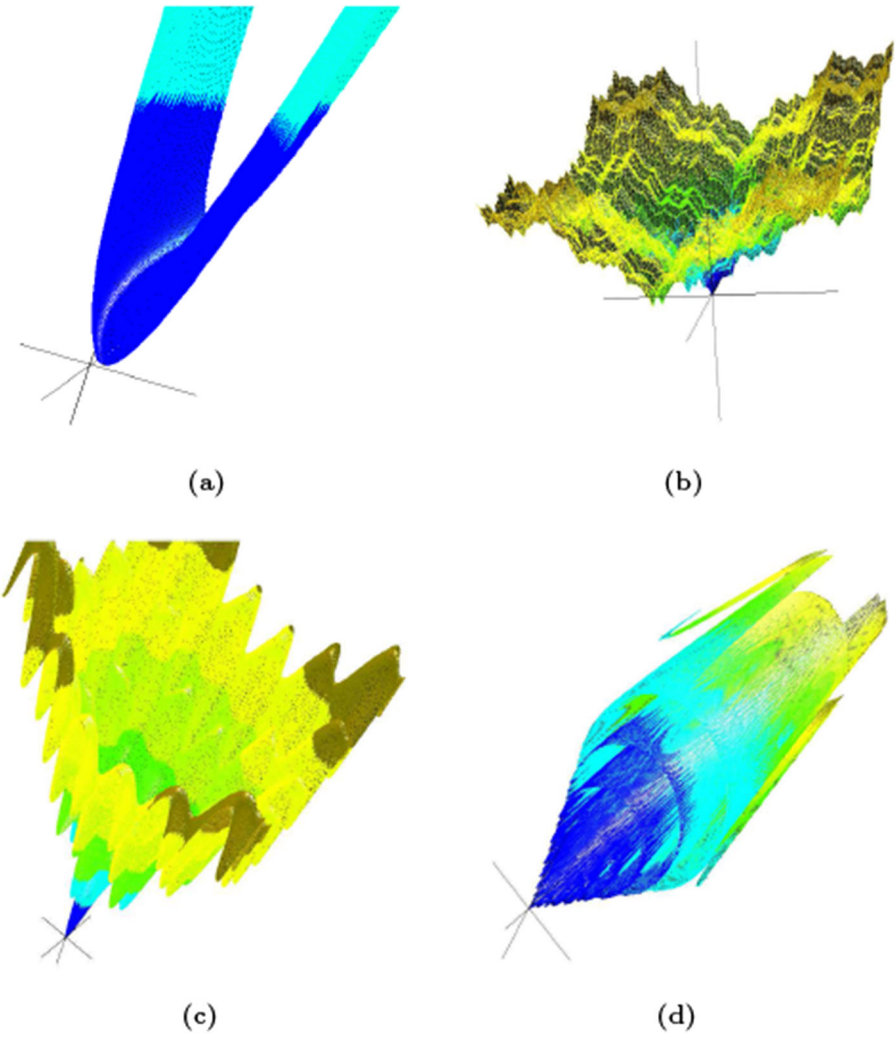


Fig. 4 (**a**) Rosenbrock's function (**b**) Weierstrass function (**c**) Ackley's function (**d**) Schaffer's F7 function

- dp: position of the temperature decrementation operation
- T: initial temperature

Values of a and T that present best performance are investigated in Sect. 4. On the other hand, for the decrementation operation (dp), we determined two possible locations; first one is just after the innermost for loop, and the second one is just before moving the solution.

The algorithm is exploring continuously with a higher T value by moving to worse neighbour solutions, whereas with a lower T value the algorithm is exploiting around the given initial solution (as in the original MBO). A high a value may decrease the temperature (the probability of moving to worse solutions) very fast, resulting an equivalent behaviour of the original MBO algorithm. Moreover, a low a value may decrease the temperature not as high as needed. On the other hand, we observe that for $dp = 1$, the temperature decrementation operation is performed m times more frequently than for $dp = 0$. Hence, a quick decrease

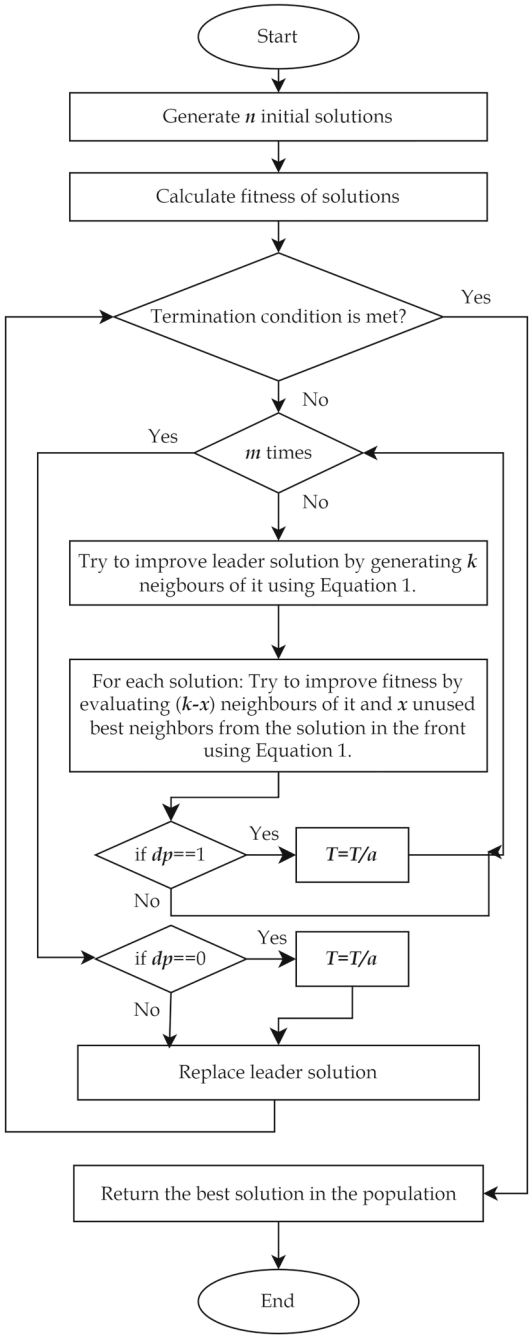


Fig. 5 The structure of the MBOx

Table 1 Description of UCI datasets

Dataset	Instances	Features	Classes	C4.5 (%)
Abalone	4177	8	29	20.49
Arrhythmia	452	275	16	60.40
Iris	150	4	3	96.00
Muskv1	476	166	2	84.87
Optdigits	5620	64	10	90.68
Ticdata2000	5822	85	41	94.45
Vehicle	846	18	4	72.58
Wine	178	13	3	93.82

occurs in T when $dp = 1$. The results of best performing values of these parameters are given and discussed in Sect. 4.

4 Computational experiments and results

In this section, we present the setup for our extensive computational experiments, their results and discussions of important findings. In the first part, we present the parameter fine tuning tests and in the second part we present the results of the detailed comparisons of the algorithms.

The experiments are run on an HP Z820 workstation with Intel Xeon E5 processor at 3.0 GHz with 128 GB RAM running Windows OS. All algorithms are implemented in the Java language. Different stopping criteria are set for the problems. Stopping criterion for the FS problem is creating a predefined number of solutions which is set to 50,000. Regarding the ONP, the stopping criterion is a predefined number of iterations (Alkaya and Algin, 2015). The stopping criterion for the QAP is a given number of solution instances generated (where each neighbor, child, mutant, trial or donor vector is counted). Specifically, the allowed number of solution instances generated is $N^3 * 100$ where N is the number of possible locations or facilities for QAP instances. On the other hand, the stopping criterion for the continuous optimization problems is $D * 10,000$ solution instances generated where D is the dimension.

4.1 Implementation details of algorithms

In the FS problem, we focus on the filter methods where metaheuristics are used as the search algorithms. In order to evaluate subsets we use correlation-based FS (CFS) (Hall, 1999) as a subset evaluator and performance of subset returned by the search algorithm is measured by using the decision tree (C4.5) classifier (Quinlan, 2014). CFS evaluates the subsets of features according to the correlation of features and class where subsets are uncorrelated with each other but highly correlated with the class. C4.5 classifier is a tree based method composed of leaf, root and branches. In the tree, each path from root to leaf represents the classification rules. Performance of algorithms is measured on the eight datasets taken from UCI machine learning repository (Lichman et al., 2013). Table 1 gives information about the datasets where the range of number of features and instances changes from 4 to 275 and 150 to 5822, respectively. Results belonging to C4.5 classifier are the accuracy values obtained by using full number of features.

In our six algorithms, solution for the FS problem is implemented as given in Algin et al. (2020). A solution is defined as a random number (between 0 and 1) vector where each element of the vector represents the weight of the features in the dataset. If the weight value of a feature is greater than or equal to 0.5, then the feature is selected in that subset, otherwise it is not selected. Mutation in GA and neighbor generation in MBO, MBOx and SA are performed by updating all elements of a solution with a small amount. Crossover in DE and GA and mutation in DE are implemented as given in Hancer et al. (2018). Exploration phase of HHO is performed by using crossover operator of DE. In the exploitation phase of HHO, neighbor generation function of MBOx is used one times with current solution for hard besiege, neighbor generation function of MBOx is used three times with current solution and the best one selected for soft besiege, neighbor generation function of MBOx is used three times with the best solution and the best one selected for soft besiege with progressive rapid dives, and for the hard besiege with progressive rapid dives neighbor generation function of MBOx is used one times with best solution.

In our QAP implementation, similar to the ones in Alkaya and Duman (2015), Duman et al. (2012), MBO and MBOx obtain a neighbour solution for the QAP by a pairwise exchange of any two locations. Regarding SA, GA and DE, implementation details are given in Davendra and Onwubolu (2007), Gambardella et al. (1999). PTL crossover is used for exploration phase of the HHO. For the exploitation phase, in a similar way as in the FS problem, neighbor generation of MBOx and mutate by insertion operator are used with current or best solution.

Remember that in ONP the agent finds shortest path between s-t points without exceeding the neutralization limits. So, we can think of the ONP as selecting at most K discs. In this study, a solution for the ONP is represented as a set of discs, S (see Fig. 6). While calculating the cost of the path, neutralization cost of all discs in the space is set to a maximum value. Then cost of discs which are in set S is set to original value. After that, Dijkstra's algorithm is used to calculate the path cost. Finally all discs' cost is set to their original value for finding new solution. An example is given in the Fig. 6 where there is a solution with five neutralization limits: $\{12, 8, 17, 2, 33\}$. Cost of this solution is calculated by maximizing the neutralization cost of all discs except $\{12, 8, 17, 2, 33\}$. Then, under this terms, Dijkstra's algorithm is used to find the shortest path. In this example only four of the discs are neutralized. Once shortest path is found, the neutralization cost of all discs is set back to their original values.

In order to apply MBOx algorithm to the ONP we need to use a well designed neighbor generation function. In our study, we used the neighbour generation function developed in Alkaya and Algin (2015). In this function, a neighbor of a solution is obtained by swapping one of the elements of S with one of its nearby discs by avoiding any replicates in S. Specifically, if the discs' centers are at most 3 * radius away from each other, a disc is closely placed to another. If there is no closely placed disc around a disc, then any disc from $\mathcal{A}$ is selected for replacement. Figure 7 depicts the neighbor generating method for the ONP. In Fig. 7a, a solution with five discs is given. One of the discs that belongs this solution is selected randomly (d_{13}). In Fig. 7b, new solution is generated by swapping d_{13} with one of its neighbors (d_{18}). SA and MBO also use the same neighbor function of the MBOx. On the other hand, for GA and DE, the crossover operator in our implementation is the one developed in Alkaya and Algin (2015), and the mutation operator is the neighbour generation function used by MBOx. In the HHO, crossover operator of DE is used in the exploration phase and for the exploitation phase, in a similar way as in the FS problem, neighbor generation function is used with different repetitions and different solutions (current or best).

In order to design a well performing MBOx algorithm an effective neighbor generating function is crucial. In a D dimensional solution space, we used D dimensional spheres (D-spheres) to have a more effective exploration plan. While generating a neighbor for a solution,

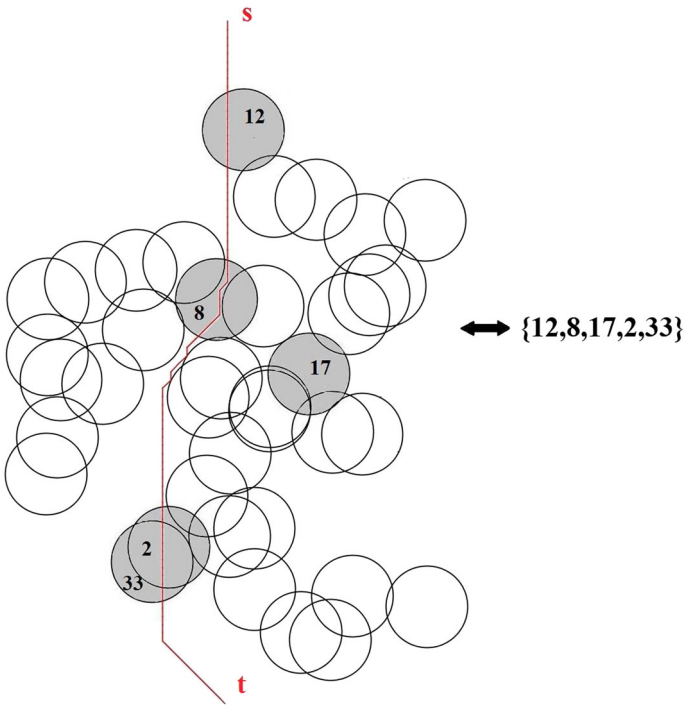


Fig. 6 A solution with five neutralization limits

its neighbor can be obtained only within the D-sphere around it. A neighbor of a solution can be at most r units away from the original solution where r is the radius of the D-sphere that surrounds it. A random number in $[0, r]$ is used to keep the distance that how far will the new solution be away from the original solution (l).

Additionally, determining the location (coordinate in each axis) of the point in the D dimensional space is also very important. For this, we used the following set of trigonometric formula.

$$
\begin{aligned}
&x_D = l * cos(\theta_{D-1}) \\
&x_{D-1} = l * sin(\theta_{D-1}) * cos(\theta_{D-2}) \\
&x_{D-2} = l * sin(\theta_{D-1}) * sin(\theta_{D-2}) * cos(\theta_{D-3}) \\
&... \\
&x_2 = l * sin(\theta_{D-1}) * sin(\theta_{D-2}) * ... * sin(\theta_2) * cos(\theta_1) \\
&x_1 = l * sin(\theta_{D-1}) * sin(\theta_{D-2}) * ... * sin(\theta_2) * sin(\theta_1)
\end{aligned}
$$

where x_i is the coordinate of the point in the i^{th} axis and θ_i is the angle between i^{th} and $(i+1)^{th}$ axis. Before using this set of formula θ_i's must be obtained randomly such that $\theta_1 \in [0, 2\pi]$ and $\theta_i \in [0, \pi]$ for $i = 2, \ldots, D-1$. An example for the formulas given above is presented in Fig. 8 for $D = 3$.

SA and MBO use the same neighbor function of MBOx. On the other hand, for GA and DE, the crossover operator in our implementation is the well-known "one-cut crossover" in which the coordinates of the solutions are used as chromosomes. The mutation operator is the neighbour generation function explained above. HHO is implemented as in the original study (Heidari et al., 2019).

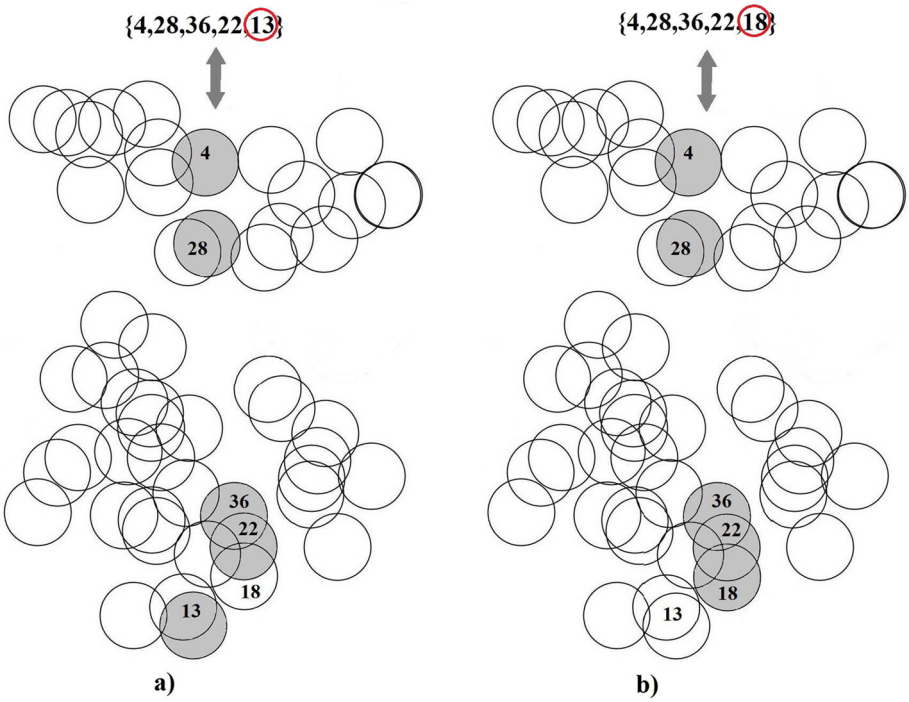


Fig. 7 A solution with five discs (**a**) and a neighbor solution with five discs (**b**)

4.2 Parameter fine tuning

As explained in the previous sections, all algorithms under focus have several parameters and they need to be fine tuned so that their best performing values are revealed for each problem. Therefore, we tried to determine the best performing parameter values of the algorithms in the first set of computational tests. In this subsection, unless stated otherwise, each reported figure is an average of 10 runs. This first set of tests are conducted on randomly selected 5 datasets taken from UCI Machine Learning Repository (Lichman et al., 2013), randomly selected 10 QAP instances (files) taken from QAPLIB (QAPLIB, 1997), 10 ONP instances given in Alkaya and Algin (2015) and 4 continuous optimization functions given in Sect. 2.3.4 for D=2 and r=1. To be consistent, if they exist in the literature, we got the best performing values of the parameters from previous studies. Otherwise we conducted parameter fine tuning tests. Table 2 presents the best performing values of the algorithms. Of those 20 applications (five algorithms each applied on four problems), ten are taken from the literature as footnoted in Table 2. HHO algorithm doesn't have any parameters to be fined tuned other than hawk number, therefore, we didn't put it to the table. As recommend in Heidari et al. (2019), we set the hawk number to 30.

Since this study is introducing the MBOx to the literature, we provide its parameter fine tuning analysis in detail. Firstly, we fine tuned the parameters peculiar to MBOx. We determined 19 values for the T and 11 values for the a parameters. On the other hand, we determined two possible locations for the decrementation operation (dp). Together with the a, T and dp parameters, best performing values of the parameters inherited from the original

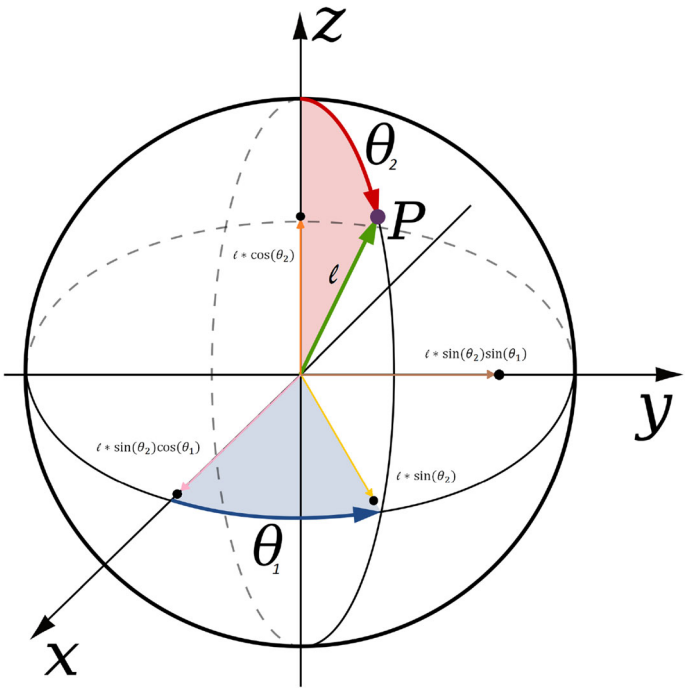


Fig. 8 Representation of a point (solution) and its vectors in three dimensions

MBO algorithm are given in Table 2. Regarding the values of parameters a and dp, 1.06 and 0 are the best performing values, respectively. On the other hand, we observed that 3000 is a better value for T. This is given in Fig. 9. In the figure, the performance of MBOx draws an U shape where low and high values of T present worse results. This is in line with the above discussion in Sect. 3 about exploration versus exploitation. In addition to the given parameters in Table 2, there is another parameter called $radius$ used in the neighbor generation phase of MBOx, MBO and SA algorithms' adapted versions for the FS problem. The values of the $radius$ parameter in the fine tune experiments are {0.01, 0.02, 0.05} and its best performing values for MBOx, MBO and SA are 0.02, 0.02 and 0.05, respectively.

4.3 Comparison of the algorithms

After fine tuning the parameters for all algorithms, we made an extensive set of tests for comparing the algorithms.

In the FS problem, 8 different datasets are used for each search algorithm. All experiments are repeated 31 times and the percentages of average accuracy values are given in Table 3.

According to Table 3, it is seen that all search algorithms have competitive results. Among six algorithms, in terms of average accuracy values, we can say that MBOx is the best performing search algorithm followed by MBO. In terms of winning cases among 8 dataset: MBOx has 5, MBO, SA and DE have 2 and GA and HHO have 1 winning case(s). When we check the feature number, all algorithms decrease the number of feature significantly which is about 50% on the average.

Table 2 Values of parameters used in the fine tune experiments

Alg.	Par.	FS	ONP	QAP	Cont. func.
MBOx	n	{5,21,**51**}	{**13**}	{3,5,11,25,**51**}	{3,5,11,25,**51**}
	k	{**3**,5,7}	{**3**,5,10}	{**3**,5,7,9}	{**3**,5,7,9}
	m	{**5**,10}	{3,**5**,7}	{**10**}	{**3**,5,7,10}
	x	{**1**,2,3}	{**1**,2,3}	{**1**,2,3,4}	{**1**,2,3,4}
	T	{1000,3000,**5000**}	{**1000**,3000}	{100, 200, 300, ..., 1000, 2000, **3000**, ..., 10, 000}	{**1000**,3000,5000}
	a	{1.1,**1.5**}	{1.05,1.1,**1.5**}	{1.01, 1.02, ..., 1.05, **1.06**, 1.07, ..., 1.1, 1.5}	{1.01,1.05,1.1,**1.5**}
	dp	{**0**,1}	{**0**,1}	{**0**,1}	{**0**,1}
MBO	n	{5,**21**,51}[1]	{**13**}[2]	{3,5,11,25,**51**}[3]	{3,5,11,25,**51**}
	k	{3,**5**,7}[1]	{**3**,5,10}[2]	{**3**,5,7,9}[3]	{**3**,5,7,9}
	m	{**5**,10}[1]	{3,**5**,7}[2]	{**10**}[3]	{3,5,**7**,10}
	x	{**1**,2,3}[1]	{**1**,2,3}[2]	{**1**,2,3,4}[3]	{**1**,2,3,4}
SA	T	{**100**,1000}[1]	{**100**,1000}[2]	Parameter settings	{**100**,1000,3000,5000}
	R	{5,**20**}[1]	{**5**,20}[2]	Proposed in	{**5**,20,50}
	a	{1.1,**1.5**}[1]	{1.1,**1.5**}[2]	Gambardella et al. (1999)	{1.01,1.1,**1.5**}
	b	{1.1,**1.5**}[1]	{1.1,**1.5**}[2]	are used	{1.01,1.1,**1.5**}

Table 2 continued

Alg.	Par.	FS	ONP	QAP	Cont. func.
DE	*nog*	{**50**,100,200}[1]	{50,**100**,200}	parameter settings	{50,100,**200**}
	nos	{**50**,100,200}[1]	{50,100,**200**}	proposed in	{50,100,**200**}
	cxp	{**0.1**,0.5,0.9}[1]	{**0.2**,0.5,0.8}	Davendra and Onwubolu (2007)	{0.2,0.5,**0.8**}
	mp	{**0.1**,0.5,0.9}[1]	{**0.2**,0.5,0.8}	are used	{0.2,0.5,**0.8**}
	nomr	—	{**1**,5,10}		{1,5,**10**}
GA	*nog*	{25,50,100,**200**}	{10,**25**,50,100}[2]	parameter settings	{25,50,100,**200**}
	nos	{25,50,100,**200**}	{10,20,**25**,50,100}[2]	proposed in	{25,50,100,**200**}
	cxp	{0.25,**0.5**,0.75,0.9}	{0.1, 0.2, 0.3, ..., 0.8, **0.9**, 1}[2]	Gambardella et al. (1999)	{**0.25**,0.50,0.75,0.9}
	mp	{**0.05**,0.25,0.75,0.9}	{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, **0.95**}[2]	are used	{0.05,0.25,**0.75**,0.9}

The best performing values are shown bold
[1] This parameter set is taken from Algin et al. (2020)
[2] This parameter set is taken from Alkaya and Algin (2015)
[3] This parameter set is taken from Duman et al. (2012)

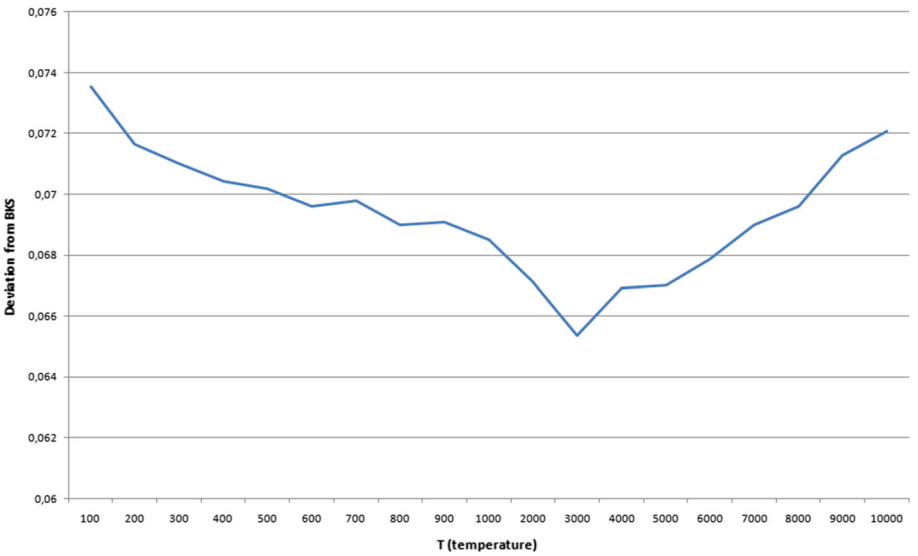


Fig. 9 Performance of MBOx with various T parameter values

Table 3 Accuracy values (%) of algorithms

Dataset	MBOx	MBO	SA	DE	GA	HHO
Abalone	**21.05**	19.34	19.34	20.21	20.02	19.34
Arrhythmia	60.43	60.64	**60.73**	60.09	60.56	60.57
Iris	**96.00**	**96.00**	**96.00**	**96.00**	**96.00**	**96.00**
Muskv1	81.92	81.99	81.52	**84.87**	81.57	82.38
Optdigits	90.13	**90.64**	90.26	89.54	88.15	90.60
Ticdata2000	**94.38**	94.21	94.20	81.47	94.14	93.25
Vehicle	**72.58**	69.30	69.82	71.39	69.04	68.32
Wine	**95.69**	93.96	93.82	91.96	94.00	93.82
Average	**76.52**	75.76	75.71	74.44	75.44	75.54

In the literature, QAP is the mostly tackled problem when compared to the other three problems. Therefore, rather than using our implementations for SA, DE and GA, we preferred to obtain and use results from the literature. The results belonging to SA and GA are obtained from Gambardella et al. (1999) and for the DE algorithm the results are taken from Davendra and Onwubolu (2007). Note that the DE algorithm used for QAP is an enhanced version of the classical DE algorithm (called EDE) and GA is hybrid GA (called HGA) where the details can be found in Davendra and Onwubolu (2007) and Gambardella et al. (1999), respectively.

In QAP experiments, we conducted tests on 38 small/medium-size and 5 large-size QAP instances with 31 runs in each test where we compared the performance of the algorithms with respect to the distance to the best known solutions (BKS) given in the literature in percentages (QAPLIB, 1997). Results are given in Table 4 as an average of 31 runs. The values given in this table are measured in per cent above the BKS. Out of 38 instances, MBOx outperformed other metaheuristics on 11 instances, MBO got best results on 9 instances, SA on 0 instance, HGA on 10 instances, EDE on 8 instances and HHO is 0 instance. According to

Table 4 Performance comparison of MBOx with other metaheuristics on the QAP instances (% deviation from BKS)

File	MBOx	MBO	SA	EDE	HGA	HHO
bur26a	0.00377	**0.00092**	0.14110	0.00600	0.01200	0.01504
bur26b	0.00472	0.00136	0.18280	**0.00020**	0.02190	0.01566
bur26c	0.00373	0.00006	0.07420	0.00005	**0.00000**	0.01787
bur26d	0.00189	**0.00004**	0.00560	0.00010	0.00020	0.01893
bur26e	0.00664	0.00006	0.12380	0.00020	**0.00000**	0.01734
bur26f	0.00218	0.00009	0.15790	0.00001	**0.00000**	0.01967
bur26g	0.02182	0.00006	0.16880	0.00010	**0.00000**	0.01697
bur26h	0.01089	**0.00004**	0.12680	0.00010	0.00030	0.01960
chr25a	**0.15510**	0.15957	12.49730	0.22700	2.69230	0.59612
els19	0.49728	0.02817	18.53850	0.00070	**0.00000**	0.31335
kra30a	**0.01550**	0.01868	1.46570	0.03280	0.13380	0.16573
kra30b	**0.00512**	0.00691	0.19470	0.02530	0.05360	0.15189
nug20	0.00303	0.00614	0.07000	0.01800	**0.00000**	0.10576
nug30	**0.00437**	0.00576	0.12100	0.00500	0.00700	0.11002
sko42	0.00457	0.00680	0.11400	0.00900	**0.00300**	0.09951
sko49	**0.00471**	0.00626	0.13300	0.00900	0.04000	0.08915
sko56	**0.00501**	0.00706	0.11000	0.01200	0.06000	0.09352
sko64	**0.00505**	0.00678	0.09500	NA	0.09200	0.08902
sko72	**0.00563**	0.00736	0.17800	NA	0.14300	0.08918
sko81	**0.00571**	0.00712	0.20600	NA	0.13600	0.08608
sko90	**0.00594**	0.00747	0.22700	NA	0.19600	0.08598
tai20a	0.02210	0.01475	0.71600	**0.01300**	0.26800	0.10616
tai20b	0.34687	0.00457	6.72980	0.00590	**0.00000**	0.11307
tai25a	0.09122	0.02113	1.00200	**0.01100**	0.62900	0.09403
tai25b	0.45325	0.00759	1.12150	0.00300	**0.00000**	0.20518
tai30a	0.02778	0.01871	0.90700	**0.01100**	0.43900	0.08950
tai30b	0.39632	0.01552	4.40750	0.02390	**0.00030**	0.15609
tai35a	0.02866	**0.01986**	1.34500	0.03700	0.69800	0.08985
tai35b	0.35272	0.01494	3.17460	**0.01010**	0.10670	0.14464
tai40a	0.02825	**0.02145**	1.30700	0.02600	0.88400	0.08853
tai40b	0.36362	0.02770	4.56460	**0.02700**	0.21090	0.17284
tai50a	0.02951	0.02392	1.53900	**0.01800**	1.04900	0.08964
tai50b	0.34902	0.01545	0.81070	**0.00100**	0.21420	0.14811
tai60a	0.05782	**0.02285**	1.39500	NA	1.15900	0.08888
tai60b	0.33741	**0.02198**	2.13730	NA	0.29050	0.16771
tai80a	0.09482	**0.02341**	0.99500	NA	0.79600	0.08483
tai80b	0.30112	**0.02588**	1.43860	NA	0.82860	0.17217
wil50	**0.00158**	0.00293	0.06100	0.03800	0.03200	0.05471

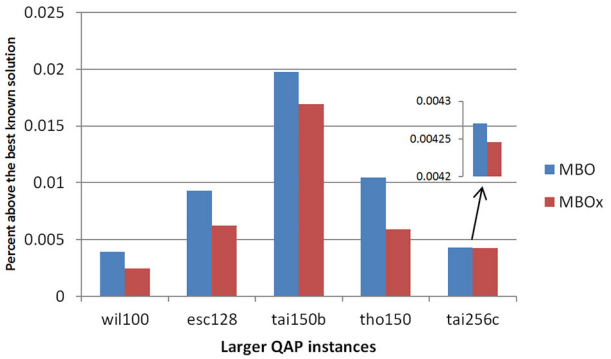


Fig. 10 Performance comparison of MBOx and MBO on large-size QAP instances

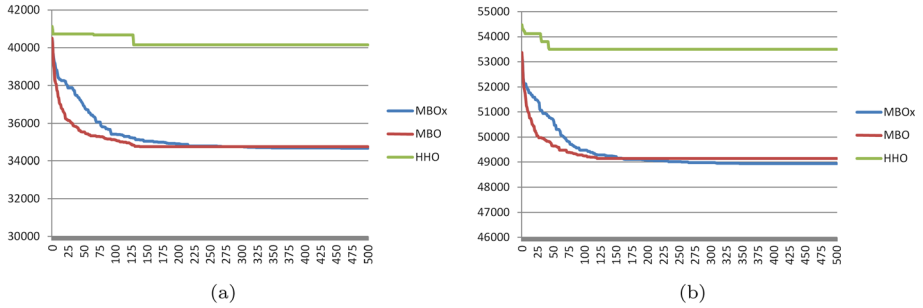
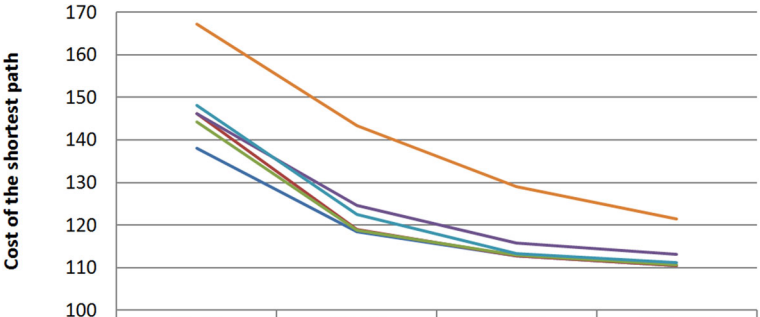


Fig. 11 Convergence graphs for the QAP instances. (**a**) sko56 (**b**) wil50

these results, it is shown that MBOx performed best among these metaheuristics. Obviously this is due to the improved exploration capability of MBOx. Following the MBOx algorithm, HGA and MBO algorithms have competitive results. On large-size QAP instances, MBOx and MBO are compared. Average of 10 runs can be seen in Fig. 10. According to the results, thanks to the new exploration strategy of MBOx, it outperforms MBO algorithm in all large-size instances.

In order to see convergence of the algorithms, we added convergence graphs for two instances (see Fig. 11). Since results of SA, EDE and HGA algorithms were taken from the literature, we put MBOx, MBO and HHO algorithms into the graphs convergence graphs. As seen in Fig. 11, MBO converges faster at the beginning, but in later iterations it gets stuck and falls behind the MBOx. HHO is originally developed for the continuous problems, therefore, it doesn't perform well at all for the QAP instances.

Tests on ONP are conducted on 10 ONP instances with 50 runs in each test. Each ONP instance includes 100 disks and the allowed maximum neutralization for the agent is set to 5. In this experiment, four types of different graph resolution are used (10×10, 20×20, 50×50 and 100×100). Results (costs of the shortest path) are shown in Fig. 12 as an average of 50 runs and 10 instances. As seen in this figure, MBOx outperforms other metaheuristics in all resolution settings. Performance of MBOx is better than others up to 20.99%, 20.94%, 14.49% and 10.02% for the resolutions 10x10, 20x20, 50x50 and 100x100, respectively. With these results we can say that MBOx is again the best performing metaheuristic on the ONP.



	10x10	20x20	50x50	100x100
MBOx	138.12	118.51	112.75	110.44
MBO	146.21	119.05	112.82	110.44
SA	144.31	118.83	113	110.61
DE	146.2	124.62	115.8	113.12
GA	148.05	122.51	113.38	111.18
HHO	167.11	143.33	129.08	121.51

Fig. 12 Performance comparison of MBOx with other metaheuristics on the ONP

Regarding the convergence graphs, one of the ONP instances is selected and its convergence graph is presented in Fig. 13 for different graph resolutions. In the figure, x axis is the number of iterations and y axis is the cost of the shortest path.

The results of the tests on continuous functions can be summarized as follows. There are 30 optimization functions used in the literature, mostly in parts, but we use all of them to assess a broad and fair comparison of the algorithms. Peculiar to the optimization functions, we can list the dimension (D) and radius (r) parameters. Dimension refers to the size of the dimension that the function lies in and radius refers to the area limit where a neighbor can be sought in. We considered 11 different dimensions and 3 different radii values. Hence, a total of 990 cases arise (algorithms work with 3 different radii values on 30 functions created in 11 different dimensions). When an algorithm is asked to find the global optimum for a function, it runs 31 times for a given setting and its average is recorded (that is each test is repeated 31 times). In order to compare the performance of the algorithms, we counted the number of cases that each algorithm outperforms others. According to the our results, it is seen that HHO algorithm, which is originally developed for the continuous domains, outperforms other algorithms on most of the cases and takes the first place in continuous function problem domain. Number of winning cases out of 30 functions for HHO are given in Table 5.

In order to see the comparison of other algorithms we present another table where HHO is not included (See Table 6). While comparing MBOx, MBO, SA, DE and GA we set the radius value as r={1, 5, 10}.

The comparison results of five algorithms are presented in Table 6. In the table, we observe that as the radius value increases, MBOx outperforms MBO and the others. This is due to the exploration capability embedded to the MBOx. Another interesting point observed in the table is the good performance of MBOx in higher dimensions. Even though the search space enlarges exponentially with increasing D values, MBOx find better results more easily than the others. Therefore, we can conclude that MBOx performs better on larger solution spaces by taking advantage of the new exploration capability.

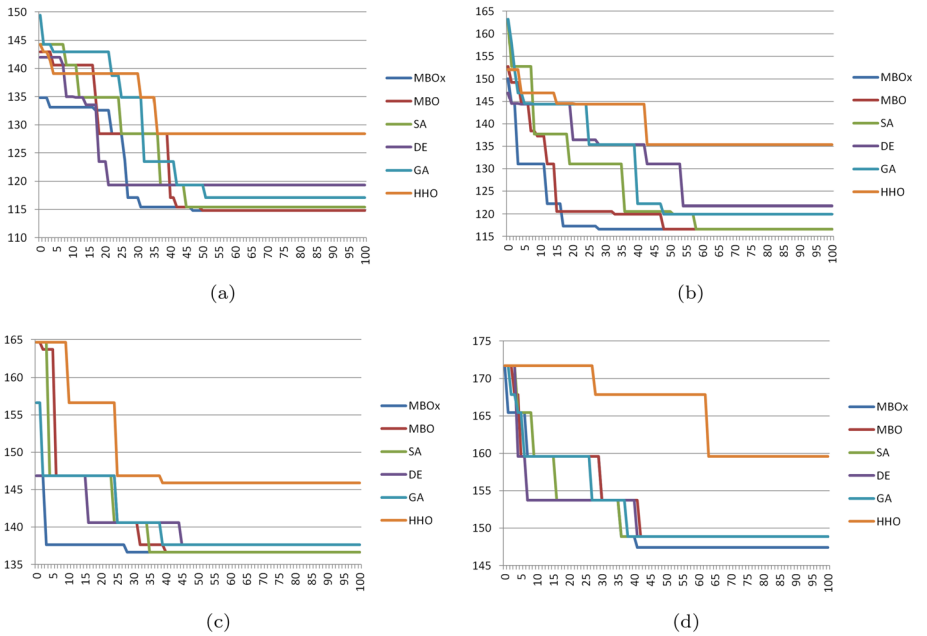


Fig. 13 Captions of subfigures should be in parentheses like (**a**) 100 × 100, (**b**) 50 × 50, (**c**) 20 × 20, (**d**) 10 × 10.

Table 5 Number of winning cases of HHO

D	HHO
2	27
5	24
10	24
15	29
20	27
25	28
30	28
35	28
40	28
45	28
50	28

Convergence graphs of the algorithms on randomly chosen two continuous functions are given in Fig. 14 where x axis is the iteration number and y axis is F(x) value.

5 Conclusion

In this study, we studied on embedding a different exploration strategy to the migrating birds optimization (MBO) algorithm. Proposed algorithm is called MBOx and its performance is tested on 8 well-known feature selection (FS) problem instances taken from UCI repository,

Table 6 Results on 30 different continuous functions with various radius (r) and dimension (D) values

	r=10					r=5					r=1				
D	*MBOx*	*MBO*	*SA*	*DE*	*GA*	*MBOx*	*MBO*	*SA*	*DE*	*GA*	*MBOx*	*MBO*	*SA*	*DE*	*GA*
2	6	**12**	6	6	0	9	**13**	4	4	0	**11**	3	6	10	0
5	**11**	4	**11**	4	0	**11**	3	9	7	0	6	2	**13**	9	0
10	10	3	**12**	5	0	10	2	**11**	7	0	7	3	**13**	7	0
15	10	3	**14**	6	1	8	3	**14**	8	1	9	2	**12**	10	1
20	**16**	3	7	7	1	**13**	4	9	7	1	10	2	7	**14**	1
25	**21**	2	2	7	2	**18**	2	3	9	2	**14**	3	5	10	2
30	**17**	5	2	9	1	**16**	6	2	7	3	**15**	6	3	9	1
35	**20**	3	2	8	1	**14**	9	2	7	2	**14**	3	3	13	1
40	**15**	8	2	8	1	**14**	9	2	8	1	**14**	3	3	13	1
45	**18**	4	2	8	2	**19**	3	2	9	1	**15**	3	3	12	1
50	**17**	5	2	9	1	**19**	4	2	7	2	**13**	4	3	**13**	1

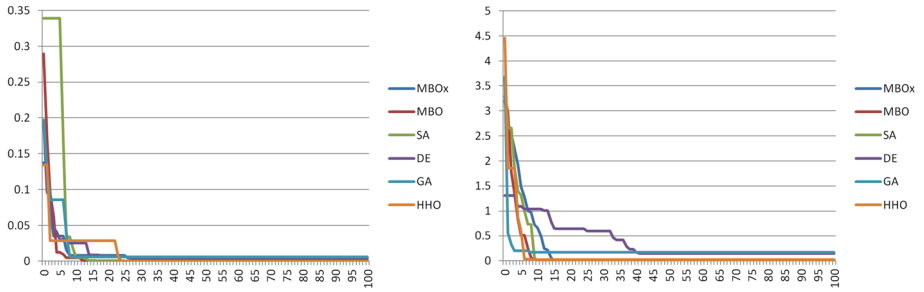


Fig. 14 Convergence graphs for the continuous functions. (**a**) f7 (**b**) f19

43 quadratic assignment problem (QAP) instances (including 5 large-size instances) taken from the QAPLIB, 10 obstacle neutralization problem (ONP) instances with four resolution settings and 30 well-known continuous optimization functions with 11 different dimensions. Results regarding the FS show that MBOx has higher accuracy value than the other algorithms. Results regarding the ONP show that again MBOx outperforms others by up to 20.99% and therefore becomes the best metaheuristic applied on the ONP, to our best. On the continuous functions, it is observed that MBOx does not lead the competition but takes the second position. On QAP, again MBOx algorithm gives solutions better than others in terms of number of winning case. As a result, MBOx is definitely showing better performance than the original MBO and other well-known metaheuristics on problems from discrete domain and therefore it is a promising problem solver for computational optimization problems. As a future research, other behavior patterns can be used to improve the MBOx. Specifically, keeping the personal best approach can be used to improve MBOx after T reaches 0 so that a better exploitation capability will be embedded. Another direction for future work might be improving MBOx by adding adaptive or self-adaptive exploration and exploitation capabilities.

Appendix A: 30 continuous functions

All continuous functions used in this study are listed below. Details of all results for the continuous function problem are given in the website (https://mimoza.marmara.edu.tr/~falkaya/files/cont_function_results.pdf).

Basic functions

1. Bent cigar function:

$$f_1(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$$

2. High conditioned elliptic function:

$$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$$

3. Neumaire 3 function:

$$x = D^2 x/100$$
$$f_3(x) = \sum_{i=1}^{D}(x_i - 1)^2 + \sum_{i=1}^{D} x_i x_{i-1} + \frac{D(D+1)(D-1)}{6}$$

4. Discus function:

$$f_4(x) = 10^6 x_i^2 + \sum_{i=2}^{D} x_i^2$$

5. Different powers function:

$$f_5(x) = \sqrt{\sum_{i=1}^{D} |x_i|^{2+4\frac{i-1}{D-1}}}$$

6. Rosenbrock's function:

$$x = 30x/100$$
$$f_6(x) = \sum_{i=1}^{D-1}(100\,(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

7. Alpine function:

$$x = 10x/100$$
$$f_7(x) = \sum_{i=1}^{D} |x_i \sin(x_i) + 0.1x_i|$$

8. Ackley's function:

$$f_8(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)) + 20 + e$$

9. Weierstrass function:

$$x = x/100$$
$$f_9(x) = \sum_{i=1}^{D}(\sum_{k=0}^{20}[0.5^k cos(2\pi.3^k(x_i + 0.5))]) - D\sum_{k=0}^{20}[0.5^k cos(2\pi.3^k.0.5)]$$

10. Griewank's function:

$$x = 600x/100$$
$$f_{10}(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$$

11. Rastrigin's function:

$$x = 5.12x/100$$
$$f_{11}(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$$

12. Katsuura function:

$$x = 5x/100$$
$$f_{12}(x) = \frac{10}{D^2}\prod_{i=1}^{D}(1 + i\sum_{j=1}^{32})\frac{|2^j x_i - |2^j x_i||}{2^j})\frac{10}{D^{1.2}} - \frac{10}{D^2}$$

13. Expanded Scaffer's F6 function:

$$g(x, y) = 0.5 + \frac{(sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$$
$$f_{13}(x) = \sum_{i=1}^{D-1} g(x_i, x_{i+1}) + g(x_D, x_1)$$

14. HappyCat function:

$$f_{14}(x) = |\sum_{i=1}^{D} x_i^2 - D|^{1/4} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$$

15. HGBat function:

$$f_{15}(x) = |(\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2|^{1/2} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$$

16. Schwefel's problem 2.22:

$$x = 10x/100$$
$$f_{16}(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|$$

17. Schwefel's problem 1.2:

$$f_{17}(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$$

18. Schwefel's problem 2.26:

$$x = 500x/100$$
$$f_{18}(x) = \sum_{i=1}^{D}(x_i \sin\sqrt{|x_i|})$$

19. Penalized function:

$$x = 50x/100$$

$$\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$$

$$f_{19}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]\}$$

$$+(x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D}\mu(x_i, 5, 100, 4)$$

20. Schaffer's F7 function:

$$f_{20}(x) = (\frac{1}{D-1}\sum_{i=1}^{D-1}(x_i^2 + x_{i+1}^2)^{1/4} + (x_i^2 + x_{i+1}^2)^{1/4}\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}))$$

21. Salomon function:

$$f_{21}(x) = 1 - \cos(2\pi\sum_{i=1}^{D}x_i) + 0.1\sum_{i=1}^{D}x_i^2$$

The following 7 functions are newly generated composition functions.

22. Well function:

$$f_{22}(x) = \begin{cases} \sum_{i=1}^{D} x_i^2 & max(x) < 20 \\ 400 * D & otherwise \end{cases}$$

Composition functions

23. '8' + '13' + '21':

$$f_{23}(x) = f_8(x) + f_{13}(x) * 10 + f_{21}(x) * 1e - 2$$

24. '2' + '9' + '15' + '16':

$$f_{24}(x) = f_2(x) * 1e - 9 + f_9(x) * 2 + f_{15}(x) * 1e - 1 + f_{16}(x) * 5e - 2$$

25. '3' + '4' + '7' + '18':

$$f_{25}(x) = f_3(x) * 0.25 + f_4(x) * 1e - 9 + f_7(x) + f_{18}(x) * 1e - 2$$

26. '5' + '6' + '12':

$$f_{26}(x) = f_5(x) * 1e - 5 + f_6(x) * 1e - 7 + f_{12}(x) * 1e - 2$$

27. ('10' + '14' + '20')*'18':

$$f_{27}(x) = f_{18}(f_{10}(x), f_{14}(x), f_{20}(x))$$

28. ('19' + '17' + '1')*'9':

$$f_{28}(x) = f_9(f_{19}(x), f_{17}(x), f_1(x))$$

29. ('3' + '12' + '15')*'8':

$$f_{29}(x) = f_8(f_3(x), f_{12}(x), f_{15}(x))$$

30. ('6' + '21' + '14')*'13':

$$f_{30}(x) = f_{13}(f_6(x), f_{21}(x), f_{14}(x))$$

Acknowledgements The continuous function data has been also used in Alkaya et al. (2014) and results of initial studies related to hybridization of MBO with SA on the QAP are presented in Algin et al. (2018).

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK).

Declarations

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

Consent to participate The authors all consent to participate in the paper editing.

Consent for publication The authors all consent to the publication of the paper.

Availability of data and materials Not applicable.

References

Abualigah, L., Diabat, A., Altalhi, M., et al. (2023). Improved gradual change-based harris hawks optimization for real-world engineering design problems. *Engineering with Computers, 39*(3), 1843–1883.

Ahmad, M. F., Isa, N. A. M., Lim, W. H., et al. (2022). Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal, 61*(5), 3831–3872.

Alabool, H. M., Alarabiat, D., Abualigah, L., et al. (2021). Harris hawks optimization: A comprehensive review of recent variants and applications. *Neural Computing and Applications, 33*, 8939–8980.

Algin, R., & Alkaya, A.F. (2015). Solving the obstacle neutralization problem using swarm intelligence algorithms. In *2015 7th international conference of soft computing and pattern recognition (SoCPaR), IEEE* (pp. 187–192).

Algin, R., Alkaya, A.F., Aksakalli, V., et al (2013). An ant system algorithm for the neutralization problem. In *Advances in computational intelligence*, (pp 53–61).

Algin, R., Alkaya, A.F., & Aksakalli, V. (2018). Hybridization of migrating birds optimization with simulated annealing. In *International conference on hybrid intelligent systems, Springer* (pp. 189–197).

Algin, R., Alkaya, A. F., & Agaoglu, M. (2020). Feature selection via computational intelligence techniques. *Journal of Intelligent & Fuzzy Systems, 39*, 6205–6216.

Alkaya, A. F., & Algin, R. (2015). Metaheuristic based solution approaches for the obstacle neutralization problem. *Expert Systems with Applications, 42*(3), 1094–1105.

Alkaya, A.F., & Duman, E. (2015). Combining and solving sequence dependent traveling salesman and quadratic assignment problems in pcb assembly. *Discrete Applied Mathematics*.

Alkaya, A. F., & Oz, D. (2017). An optimal algorithm for the obstacle neutralization problem. *Journal of Industrial & Management Optimization, 13*(2), 835–856.

Alkaya, A.F., Algin, R., Sahin, Y., et al (2014). Performance of migrating birds optimization algorithm on continuous functions. In *Advances in swarm intelligence* (pp. 452–459).

Alkaya, A. F., Aksakalli, V., & Priebe, C. E. (2015). A penalty search algorithm for the obstacle neutralization problem. *Computers & Operations Research, 53*, 165–175.

Beasley, D., Bull, D., & Martin, R. (1993). An overview of genetic algorithms: Part i, fundamentals. *University Computing, 15*, 58–69.

Behnamian, J., Zandieh, M., & Ghomi, S. F. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an aco, sa and vns hybrid algorithm. *Expert Systems with Applications, 36*(6), 9637–9644.

Benkalai, I., Rebaine, D., Gagné, C., et al. (2017). Improving the migrating birds optimization metaheuristic for the permutation flow shop with sequence-dependent set-up times. *International Journal of Production Research, 55*(20), 6145–6157.

Brahami, M. A., Dahane, M., Souier, M., et al. (2022). Sustainable capacitated facility location/network design problem: A non-dominated sorting genetic algorithm based multiobjective approach. *Annals of Operations Research, 311*(2), 821–852.

Davendra, D., & Onwubolu, G. (2007). Enhanced differential evolution hybrid scatter search for discrete optimization. In *2007 IEEE congress on evolutionary computation, IEEE* (pp. 1156–1162).

Debuse, J. C., & Rayward-Smith, V. J. (1997). Feature subset selection within a simulated annealing data mining algorithm. *Journal of Intelligent Information Systems, 9*(1), 57–81.

Deng, W., Zhang, X., Zhou, Y., et al. (2022). An enhanced fast non-dominated solution sorting genetic algorithm for multi-objective problems. *Information Sciences, 585*, 441–453.

Diao, R., & Shen, Q. (2015). Nature inspired feature selection meta-heuristics. *Artificial Intelligence Review, 44*(3), 311–340.

Drezner, Z. (2008). Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Computers & Operations Research, 35*(3), 717–736.

Duan, Q., Liao, T., & Yi, H. (2013). A comparative study of different local search application strategies in hybrid metaheuristics. *Applied Soft Computing, 13*(3), 1464–1477.

Duman, E., & Elikucuk, I. (2013). Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization. In *Advances in computational intelligence* (pp. 62–71).

Duman, E., Uysal, M., & Alkaya, A. F. (2012). Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences, 217*, 65–77.

Elgamal, Z.M., Yasin, N.B.M., Tubishat, M., et al (2020). An improved harris hawks optimization algorithm with simulated annealing for feature selection in the medical field. IEEE Access 8:186,638–186,652

Ferreira, K. M., & de Queiroz, T. A. (2022). A simulated annealing based heuristic for a location-routing problem with two-dimensional loading constraints. *Applied Soft Computing, 118*(108), 443.

Fontes, D. B., Homayouni, S. M., & Gonçalves, J. F. (2023). A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *European Journal of Operational Research, 306*(3), 1140–1157.

Gambardella, L. M., Taillard, É. D., & Dorigo, M. (1999). Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society, 50*(2), 167–176.

Gen, M., & Lin, L. (2023). Genetic algorithms and their applications. In *Springer handbook of engineering statistics. Springer* (pp. 635–674).

Gonçalves, J. F., de Magalhães Mendes, J. J., & Resende, M. G. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research, 167*(1), 77–95.

Hall, M.A. (1999). Correlation-based feature selection for machine learning.

Hanan, M., & Kurtzberg, J. M. (1972). A review of the placement and quadratic assignment problems. *Siam Review, 14*(2), 324–342.

Hancer, E., Xue, B., & Zhang, M. (2018). Differential evolution for filter feature selection based on information theory and feature ranking. *Knowledge-Based Systems, 140*, 103–119.

Heidari, A. A., Mirjalili, S., Faris, H., et al. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97*, 849–872.

Holland, J. (1986). Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based system. *Machine Learning* pp. 593–623.

Kalayci, G.T., Alkaya, A.F., & Algin, R. (2019). Exploitation and comparison of computational intelligence techniques on the feature selection problem. In *International conference on intelligent and fuzzy systems* (pp. 1243–1249). Springer.

Kao, Y. T., & Zahara, E. (2008). A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Applied Soft Computing, 8*(2), 849–857.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

Kosanoglu, F., Atmis, M., & Turan, H. H. (2022). A deep reinforcement learning assisted simulated annealing algorithm for a maintenance planning problem. *Annals of Operations Research* (pp. 1–32).

Li, W., Shi, R., & Dong, J. (2023). Harris hawks optimizer based on the novice protection tournament for numerical and engineering optimization problems. *Applied Intelligence, 53*(6), 6133–6158.

Liao, T., Chang, P., Kuo, R., et al. (2014). A comparison of five hybrid metaheuristic algorithms for unrelated parallel-machine scheduling and inbound trucks sequencing in multi-door cross docking systems. *Applied Soft Computing, 21*, 180–193.
Lichman, M., et al (2013) Uci machine learning repository.
Liu, Y., Heidari, A. A., Cai, Z., et al. (2022). Simulated annealing-based dynamic step shuffled frog leaping algorithm: Optimal performance design and feature selection. *Neurocomputing, 503*, 325–362.
Misevicius, A. (2004). An improved hybrid genetic algorithm: new results for the quadratic assignment problem. *Knowledge-Based Systems, 17*(2), 65–73.
Oreski, S., & Oreski, G. (2014). Genetic algorithm-based heuristic for feature selection in credit risk assessment. *Expert Systems with Applications, 41*(4), 2052–2064.
Oz, D. (2017). An improvement on the migrating birds optimization with a problem-specific neighboring function for the multi-objective task allocation problem. *Expert Systems with Applications, 67*, 304–311.
Pan, Q. K., & Dong, Y. (2014). An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation. *Information Sciences, 277*, 643–655.
Paul, G. (2011). An efficient implementation of the robust tabu search heuristic for sparse quadratic assignment problems. *European Journal of Operational Research, 209*(3), 215–218.
QAPLIB (1997) Quadratic assignment problem library. http://www.opt.math.tugraz.at/qaplib/
Quinlan, J. (2014) C4. 5: programs for machine learning.
Segredo, E., Lalla-Ruiz, E., Hart, E., et al. (2018). On the performance of the hybridisation between migrating birds optimisation variants and differential evolution for large scale continuous problems. *Expert Systems with Applications, 102*, 126–142.
Sioud, A., & Gagné, C. (2018). Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times. *European Journal of Operational Research, 264*(1), 66–73.
Sohail, A. (2023). Genetic algorithms in the fields of artificial intelligence and data sciences. *Annals of Data Science, 10*(4), 1007–1018.
Song, Y., Cai, X., Zhou, X., et al. (2023). Dynamic hybrid mechanism-based differential evolution algorithm and its application. *Expert Systems with Applications, 213*(118), 834.
Sörensen, K. (2015). Metaheuristics-the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3–18.
Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.
Taillard, E. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing, 17*(4), 443–455.
Taillard, E. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location Science, 3*(2), 87–105.
Tongur, V., &Ülker, E. (2018) Pso-based improved multi-flocks migrating birds optimization (imfmbo) algorithm for solution of discrete problems. *Soft Computing* pp. 1–16.
Wang, G. G., Gao, D., & Pedrycz, W. (2022). Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. *IEEE Transactions on Industrial Informatics, 18*(12), 8519–8528.
Wang, X., Yang, J., Teng, X., et al. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters, 28*(4), 459–471.
Yu, C., Chen, M., Cheng, K., et al. (2022). Sgoa: annealing-behaved grasshopper optimizer for global tasks. *Engineering with Computers, 38*(Suppl 5), 3761–3788.
Zhang, Y., Liu, R., Wang, X., et al. (2021). Boosted binary harris hawks optimizer and feature selection. *Engineering with Computers, 37*, 3741–3770.
Zhang, Y., Chen, G., Cheng, L., et al. (2023). Methods to balance the exploration and exploitation in differential evolution from different scales: A survey. *Neurocomputing, 561*(126), 899.
Zouache, D., Got, A., & Drias, H. (2023). An external archive guided harris hawks optimization using strengthened dominance relation for multi-objective optimization problems. *Artificial Intelligence Review, 56*(3), 2607–2638.