



MAB-OS: Multi-Armed Bandits Metaheuristic Optimizer Selection

Kazem Meidani^a, Seyedali Mirjalili^{b,c}, Amir Barati Farimani^{a,d,e,*}

^a Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

^b Centre for Artificial Intelligence Research and Optimization, Torrens University Australia, Australia

^c Yonsei Frontier Lab, Yonsei University, Seoul, Republic of Korea

^d Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA

^e Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

ARTICLE INFO

Article history:

Received 13 October 2021

Received in revised form 22 July 2022

Accepted 1 August 2022

Available online 5 August 2022

Keywords:

Metaheuristic

Optimization

Multi-Armed Bandits

Reinforcement Learning

Algorithm selection

Adaptive algorithm

ABSTRACT

Metaheuristic algorithms are derivative-free optimizers designed to estimate the global optima for optimization problems. Keeping balance between exploitation and exploration and the performance complementarity between the algorithms have led to the introduction of quite a few metaheuristic methods. In this work, we propose a framework based on Multi-Armed Bandits (MAB) problem, which is a classical Reinforcement Learning (RL) method, to intelligently select a suitable optimizer for each optimization problem during the optimization process. This online algorithm selection technique leverages on the convergence behavior of the algorithms to find the right balance of exploration–exploitation by choosing the update rule of the algorithm with the most estimated improvement in the solution. By performing experiments with three armed-bandits being Harris Hawks Optimizer (HHO), Differential Evolution (DE), and Whale Optimization Algorithm (WOA), we show that the MAB Optimizer Selection (named as MAB-OS) framework has the best overall performance on different types of fitness landscapes in terms of both convergence rate and the final solution. The data and codes used for this work are available at: <https://github.com/BaratiLab/MAB-OS>.

© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

Metaheuristic Optimization algorithms have been developed to solve a wide range of optimization problems in engineering and science. Due to the intrinsic differences of optimization problems including the type of problem, conditions, and the dimensions, there is a large quantity of algorithms that cater such needs. The metaheuristic area, in particular, contains a vast variety of algorithms that are modeled from different sources of inspiration. A large group of them are nature-inspired algorithms that get their sources of mathematical models from the nature.

Nature-inspired algorithms cover a large number of methods that have been roughly categorized into several classes. Evolutionary Algorithms (EA) are based on natural selection, and contain mathematical models of mutation and crossover. Genetic Algorithms (GA) [1] and Differential Evolution (DE) [2] are of the most popular examples in this category. Human strategies and laws of physics have also been inspiration sources for some algorithms like Gravitational Search Algorithms (GSA) [3]. At last, swarm-based algorithms or Swarm Intelligence (SI) mimic the behavior of a group or swarm of animals. In fact, the optimization

search agents are observed as particles or members of a group. Some of the well-known SI algorithms include Particle Swarm Optimization (PSO) [4] and Artificial Bee Colony (ABC) [5].

All of the mentioned metaheuristic categories and sub-categories have common features that make them suitable for solving many types of problems [6–8]. These features include the simplicity and flexibility as well as the effectiveness of these methods to find global optima via avoiding local optima. More importantly, they are all derivative-free algorithms. Derivative-free or gradient-free algorithms are optimizers that do not depend on the gradient of the objective function as opposed to gradient-based methods such as gradient descent algorithms. Therefore, a derivative-free algorithm (also called black box optimizer) is an appropriate choice when the mathematical forms of objective function or its derivatives are either unknown or expensive to be extracted.

The extensive variety of problems has led to the development of a huge number of algorithms that perform well for different types of problems. Performance complementarity [9] of the algorithms, also known as No Free Lunch (NFL) theorem [10], provides a logical explanation for the existence of so many different algorithms. Researchers have been seeking comprehensive algorithms to cover the efficient solution of different types of problems in a single algorithm which has brought significant improvements in the field recently. So, metaheuristic algorithms are usually

* Corresponding author at: Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA.

E-mail address: barati@cmu.edu (A. Barati Farimani).

proposed after testing on various problems. Nevertheless, the performance complementarity of algorithms would still exist. Therefore, among a set of algorithms, the best optimizer would still depend on the type of problem. Moreover, there can be a combination of algorithms that performs even better than each of them individually. Hence, the best algorithm depends not only on the problem but also on the stage of optimization that we are currently in.

While there is an increasing number of optimization methods being introduced, issues such as the exploration-exploitation dilemma always remain difficult to be addressed. The proposed methods have different tendencies to perform exploitation, i.e. using the so far achieved information, or exploration, i.e. spending time to gather more information. Therefore, an algorithm with more exploitation properties can outperform another in relatively simpler problems such as unimodal objective functions, while has the inferior performance in more complex multimodal fitness landscapes.

Hybrid methods have been introduced as a possible solution to address the issue of exploration-exploitation balance by improving the abilities of all of the underlying methods. For example, an algorithm (A) with high exploitation capability can be hybridized by another algorithm (B) that explores the search space better. Such hybrid algorithms, however, are again limited to how they have been combined together. These hybrid methods can also be composed of some sub-parts of algorithms or external operators, such as chaotic maps [11] or lévy flight [12], to devise a new human-made algorithm.

As mentioned in the above discussion, the performance of algorithms significantly depends on the problem. However, the majority of the algorithms do not intend to rely on the feedback from the type of problem or its landscape. Adaptive algorithms have been known as another solution by adjusting the parameters in the algorithm based on the history of the data gathered throughout the optimization such as the features extracted from its fitness landscape [13,14]. Another view of employing multiple algorithms is to use algorithm selection and hyperheuristic techniques in order to select the suitable choice of optimizer based on the problem condition. Hyperheuristics are categorized into several classes based on their methodology and objectives [15–19]. For example, features from the landscape, such as its complexity or modality, can be extracted and used to select an algorithm update rule among a set of algorithms in a portfolio [9,20,21]. Selecting the favorable parameters or operators based on these features have been effective in reaching better solutions in GA [22,23] and Differential Evolution (DE) [24,25].

Besides landscape-aware (using features) algorithm selection techniques, there are other selection hyper-heuristic schemes [15] that can make use of search operators (low-level heuristics) from different metaheuristic (mid-level heuristic) algorithms in a hyper-heuristic selection framework to find a sequence of operators that builds the best metaheuristic for a specific problem [16]. In [16], simulated annealing (SA) [26] is used to tune the hyper-heuristic to find the best metaheuristic by assembling a set of extracted search operators from various baseline metaheuristics including direct search operators and more complicated operators in swarm algorithms. These hyper-heuristic frameworks can potentially find a suitable set of operators for a given problem by following an iterative search scheme such as SA. However, they cannot be used for an online search of the suitable optimizers when the problem is to be solved at once which is usually the case when a new black box search space has to be approached for optimization.

Such methods try to remedy the exploration balance issue, yet they are limited in some aspects. The hybrid methods do not use any feedback from the optimization process, and have to be

devised manually and independently from the problem. While the fitness adaptive methods generally provide better solutions to this issue, they are dependent on the definition of a meaningful feature and its extraction during the optimization. This is, however, not always readily accessible since the features may not always be good representatives for the problem. Therefore, making decision or planning strategy based on the extracted feature can be misleading in some cases. Another major limitation of the algorithm selection methods is that most of these methods select a single algorithm among several options to perform the whole optimization process. This makes the selected strategy to be at most as best as the current algorithms and never outperform them. *Online algorithm selection during the course of optimization, however, can provide us with the opportunity to use the merits of different algorithms and reach an outstanding performance that is better than all of the algorithm choices.* In this work, we aim to find the best algorithms in an online manner without using the landscape features (feature-agnostic), so we only rely on the convergence behavior and we do not compute any landscape features.

Reinforcement learning (RL) techniques have shown promising results in many applications such as optimization problems [27,28]. RL could be combined with metaheuristic optimization in different ways. For instance, the agent can seek the best strategy in a continuous or discrete space of strategies or parameters to adjust the exploration rate [29]. Multi-Armed Bandits (MAB) problem, as a classical RL algorithm, has also been considered in the context of optimization [30]. The algorithm selection problem can be viewed as an MAB problem [17,31] where the goal is to let the agent find the best algorithm via interactive trials based on the feedback from the optimization and its learning curve [32]. MABs have been used to search for the operators in the Evolutionary Algorithms (EA) [33,34] where different types of problems and reward definitions are analyzed to select the best operators [35]. MABs with Thompson Sampling (TS) have also been introduced [17] in a generic evolutionary algorithm form to improve the operator selection scheme by using TS over other selection mechanisms such as upper confidence bound (UCB).

The mentioned hyper-heuristics, including stationary or dynamic MABs with different selection mechanisms, have shown promising results on improving evolutionary algorithms (EA) to have outstanding performances on various types of problems. However, their span of control is usually limited to a set of defined operators in the EA algorithm. In other words, they seek tuning the defined parameters in an attempt to find the suitable sets of parameters or operators for a single algorithm.

In this work, multi-armed bandits problem is applied for algorithm selection to identify the best metaheuristic algorithm's update rule in an online manner during the optimization. In this framework, each metaheuristic algorithm's update rule is viewed as an armed bandit. Each armed bandit can be a stand-alone algorithm with its own sets of defined parameters and update rules. Since the convergence curves usually have dynamic behavior, the resulting MAB problem has a non-stationary setting and the RL agent looks for the best algorithm in the current status of optimization based on the achieved rewards.

The contribution of this work is to propose a general framework that can use different stand-alone competent algorithms as baseline and improve upon their performance overall by adaptively selecting the best optimizer among them for each specific condition using an RL agent. We showcase the effectiveness of the framework in finding better algorithms by evaluating its performance on obtaining the good solutions and having high convergence rate when three metaheuristic algorithms are considered as baselines. This framework then can be easily used in various applications to efficiently find and use the best algorithm

for optimization. Depending on the type of problem MAB-OS framework can be applied on different choices of base algorithms.

The structure of this work is as follows: We review the Multi-Armed Bandit (MAB) approach and its optimization methodology in the context of algorithm selection in Section 2. Then, the base algorithms and the experiments designed to show the performance of the framework are elaborated upon in Section 3. Section 4 discusses some of the observations from the experiments and makes conclusions about the current and future possible directions of the work.

2. Method

2.1. Multi-Armed Bandits problem

Multi-Armed Bandits (MAB) is a classical problem in Reinforcement Learning (RL) where the balance between exploitation and exploration plays a key role. In this problem, N armed bandits with different utility properties are considered. At the beginning, there is usually no prior information on these bandits. The resource, which can be observed in terms of computation time, is fixed and limited. The goal is to maximize the gain or equivalently minimize the regret when we allocate the resources to the bandits. More interaction with the bandits would reveal more information on their properties. In the classic static MAB problem, the underlying value of the bandits are constant over time, meaning that there is a single best bandit that can maximize the reward. In many problems such as optimization, however, the problem is dynamic, and the best operator can be variable based on the problem situation.

The information about the bandits is stored in an array of values Q that is updated as we get more feedback by interacting with bandits. We have one scalar value for each bandit at each time step, so we write the Q array as $Q(t, a)$ where a denotes action a . The feedback from the interaction with bandits appears in terms of a reward ($R(t, a)$) that is defined as the outcome of selecting a specific bandit. At each iteration, we select one bandit based on the Q values and by following some exploration strategy that is explained later. Selection of the best, i.e., highest, Q at each time step leads to a greedy, fully exploiting strategy. To allow other underestimated bandits to have more chance, exploration mechanisms like ϵ -greedy, optimistic initialization, Upper Confidence Bound (UCB), and Thompson sampling [17] can be integrated to the strategy. Dynamic systems, like selecting the best optimizer, require even more exploration since the suitable bandit can be changed over time.

Formally speaking, for a non-stationary multi-armed bandits problem where the reward (R), values (q), and therefore estimated values (Q) are functions of time, we can define the true action-value q as the expected reward achieved by selecting action a at time step t :

$$q(t, a) := \mathbb{E}[R(t)|A(t) = a] \quad (1)$$

where $A(t) = a$ denotes the selection of action a in the iteration t , and $R(t)$ is the corresponding reward as the consequence of this action. However, since $q(t, a)$ is unknown, we estimate it by $Q(t, a) \propto q(t, a)$, and the greedy action would be $A^*(t) = \operatorname{argmax}_a Q(t, a)$. If the optimal action at time t is $a^*(t)$, then the optimal value is defined as:

$$v^*(t) := q(a^*, t) = \max_{a \in A} q(a, t) \quad (2)$$

and the total regret is defined as the opportunity loss throughout the process as follows:

$$L_T = \mathbb{E} \left[\sum_{t=1}^T (v^*(t) - q(a(t), t)) \right] \quad (3)$$

The goal of MAB is to minimize the total regret, or to equivalently maximize the total gained rewards. Since we estimate the true values, the regret is inevitable as we cannot always select the optimal action. We expect the estimation to get more accurate over time if the changes in the system are smooth. The estimation of the value of action a is updated whenever the action is selected and a reward is collected. By setting $N_t(a)$ as the number of times that action or operator a is selected, we can write: $Q(t, a) = f(R_1, R_2, \dots, R_{N_t(a)})$. In a stationary problem where the rewards of different time steps contain the same amount of information on the current status of the system, Q can be calculated as a simple average of the achieved rewards, i.e. $Q(t, a) = \sum_{i=1}^{N_t(a)} R_i / N_t(a)$. In general, whether stationary or non-stationary, we can write the estimation with incremental updates as follows:

$$Q(t+1, a) = (\alpha)Q(t, a) + (1 - \alpha)R(t) \quad (4)$$

where α is a hyperparameter ($\alpha \in [0, 1]$) that we can choose. Lower values of α result in more concentration on the recent rewards and are more suitable for the dynamic systems. The choice of a too small α , on the contrary, leads to relying only on the current unstable reward. So, a balance is needed for this parameter selection. The value of α used for this study is mentioned in the experiments section.

2.2. Action selection

One of the major concerns of this problem that is known to be the most important aspect of MAB is to keep balance between exploitation of the current uncertain estimations and exploration to obtain better estimations with the possible cost of increased regret. It is shown that employing the upper confidence bound (UCB) [36] of the estimation for action selection can be a good strategy that alleviates the greedy exploitation issue. In this approach, in addition to estimating the expected value, the uncertainty of the actions are also taken into account. The UCB is defined as:

$$UCB(t, a) = Q(t, a) + cU(t, a) = Q(t, a) + c \sqrt{\frac{\ln(t)}{N_t(a)}} \quad (5)$$

where c is a hyperparameter that determines the effect of upper bound of confidence to the final estimation. c is a non-negative real number where its small values reduce the effect of uncertainties until it will have no effect in case of $c = 0$. Large values of c , on the other hand, can overemphasize the uncertainties that would fade the effect of Q values. Based on this, a balance is required for the selection of this value. The choice of c for our study is mentioned later in the experiments section. Also, note that $U(t, a)$ in the second term depends on both number of total iterations so far t and the number of trials of action a . The uncertainty shrinks over time, i.e. less uncertainty, as we have more trials ($N_t(a)$) with a specific action and increases when we have trials with other actions (increase in t).

Since at the beginning we have equal Q estimations for different actions and $N_t(a)$ is zero for every actions, we have to select the actions purely by exploration. This phase can be done by random selection of actions. We, however, try each action for only one duration in an arbitrary order of actions. After trying the actions once, we have updated Q values and non-zero $N_t(a)$, so we can continue the process based on UCBs as follows.

Based on the estimation of the value and the upper confidence bound, we expect that higher Q values correspond to better actions. To softly select the action based on the UCB value, we employ softmax sampling:

$$P(A = a) = \operatorname{softmax}(UCB(a)) = \frac{\exp[UCB(a)]}{\sum_{a \in A} \exp[UCB(a)]} \quad (6)$$

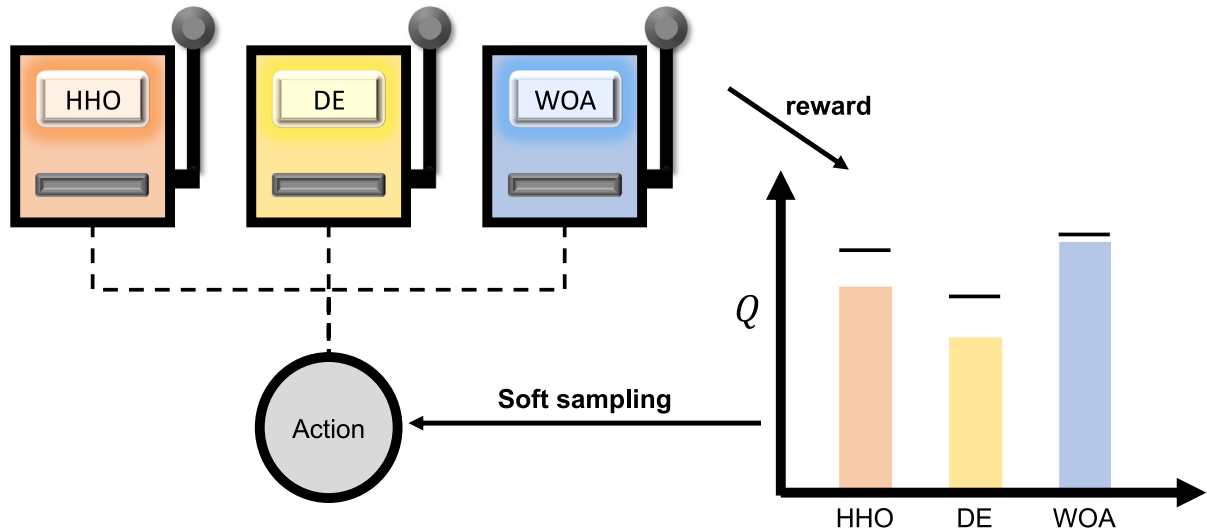


Fig. 1. The scheme of Multi-Armed Bandits (MAB) problem on the optimization with HHO, DE, and WOA as base algorithms viewed as armed-bandits. The collected rewards update the Q values, and the action is softly sampled based on the Upper Confidence Bound (UCB) of bandits.

2.3. Optimizer selection

In the context of metaheuristics, we apply the multi-armed bandits problem to the online selection of optimizers among a set of multiple options. In fact, operators are observed as bandits and the action is to use a specific update rule for the positions of the search agents. This problem is highly dynamic since the optimization process includes a dynamic landscape and the agents start from exploring the space toward convergence to the optima. Therefore, different optimizers can be the optimal choice at different stages of the optimization. Viewing these optimizers as the actions that should be selected over the course of optimization justifies the use of MAB for Optimizer Selection (MAB-OS).

One of the most important aspects of reinforcement learning and multi-armed bandits is the reward definition. The defined reward has to be meaningful and representative of the task that we are seeking to do. For metaheuristic optimization, the goodness of the optimization is determined by the function values. In the case that we want to train a model to find a single best algorithm from a portfolio to perform the whole optimization, the final obtained solution at the end of optimization can be considered as a sparse reward. It should be noted that such a stochastic value depends on a lot of parameters. Here, however, this is not the case since in this work we are looking for an online method that can select the update rules during the optimization, assuming that we have a limited computational resources.

The improvements of the fitness values over the course of optimization, for example after K iterations, can be a good representation of the performance of the algorithm. The reasoning for this choice is that better algorithms are able to achieve more improvements in the same amount of time compared to their alternatives. Therefore, we can consider this improvement as the reward during the optimization (Eq. (7a)). Moreover, since the problem is fully non-stationary, we do not want to rely on old Q values, and thus we increase the weight of recent rewards compared to the previous values. To this end, we choose $\alpha = 0.5$ to involve the most recent reward to high extent into the estimated values (Eq. (7b)).

$$R(t) = F_{t-K}^* - F_t^* \quad (7a)$$

$$Q(t+1, a) = 0.5Q(t, a) + 0.5R(t) \quad (7b)$$

Another important point about multi-armed bandits problem is the effect of number of armed-bandits, i.e. algorithms in this

work, on the solution. In a stationary open-ended trial scheme, adding a new armed-bandit may result in better asymptotic result since the new option can have better inherent value. In the optimization problem considered in this work, the time, or equivalently number of fitness evaluations, is limited. Hence, while considering more options can bring about better limiting behavior but also requires spending the time budget on the evaluation of the new options. Therefore, a balance should be kept for the number of algorithms in the framework. We argue that choosing competitive algorithms as the base armed-bandits helps the framework to exploit their merits compared to a case where one algorithm outperforms the other in the majority of the problems. Algorithm 1 elaborates upon the steps taken in the MAB framework using a set of base algorithms $\mathcal{A} = \{A_1, A_2, \dots, A_m\}$.

Algorithm 1: Multi-Armed Bandit Metaheuristic Algorithm Selection

Input: Update rules of Metaheuristic Algorithms

A_1, A_2, \dots, A_m

Initialize the population X_i ($i = 1, 2, \dots, N$) and maximum number of iterations T

Set interval (K), and evaluate initial fitness values

Take each action (algorithm) for a K -iteration period and update Q values using Eq. (7) for every iteration ($K = 1$ in Eq. (7a)).

while $t \leq T$ **do**

if $t \% K = 0$ **then**

 Select algorithm $A_i = a$ with soft sampling based on Q values using Eq. (6)

 Use algorithm A_i update rule

return X^*, F^*

To showcase the performance of framework, we select three successful metaheuristic algorithms as base armed-bandits. Based on a primary observation on the performances of a set of optimizers on the objective functions (convergence curves are provided in Appendix A), we select Harris Hawks Optimizer (HHO) [37], Differential Evolution (DE) [2], and Whale Optimization Algorithm (WOA) [38] as the options in the framework. Therefore, at each stage of the optimization, the positions of agents gets updated based on one of these algorithms that is selected based on UCB of Q values (Fig. 1).

As shown in Fig. 1 and explained in the algorithm 1, the algorithms are selected and used for some iterations during the

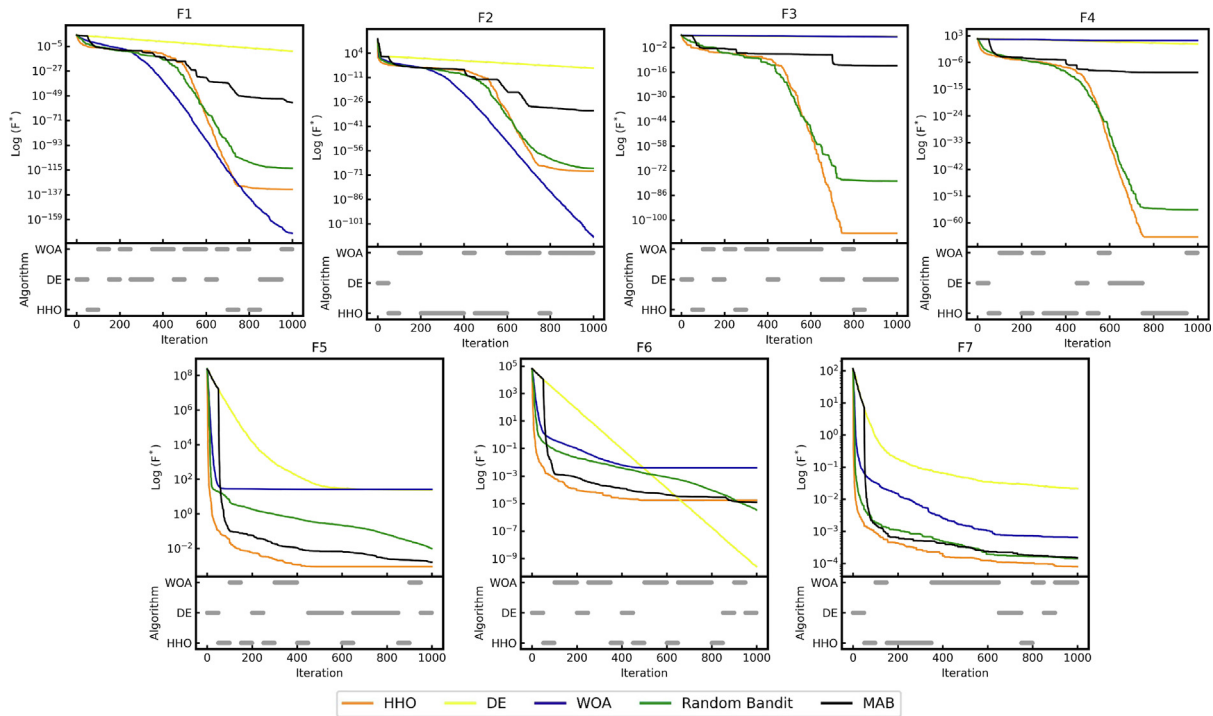


Fig. 2. The performance of multi-armed bandit (MAB) between Differential Evolution (DE), Harris Hawks Optimizer (HHO), and Whale Optimization Algorithm (WOA) compared to the vanilla DE, HHO, and WOA algorithms, and random bandit selection baseline on unimodal fitness landscapes.

optimization. Hence, there is a caveat for using the proposed framework that the selected base algorithms have to be able to communicate with each other in some way. By communication, we point to the information exchange between the algorithms. For example, the iteration, the best solution, and the position matrix of agents should be shared among all of the algorithms so that they can be updated using each of the update rules easily. This feature is not easily and readily available for every choices of metaheuristic algorithms. First, some algorithms require additional computation or features that may not be provided with other algorithms. The storage of personal best scores in Particle Swarm Optimization (PSO) [4] is a good example of this extra computation. To address this issue, we have to keep track of these parameters globally for all of the algorithms. Second, there is a need for the effective implementation of the main united optimization with different subroutines to perform the update rules.

3. Experiments

In this section, the experimental setup and environment are first discussed. The convergence analysis of the proposed method on several test cases are then covered in details. Next, the statistical analysis and comparison with other algorithms are provided. And finally, we show that MAB-OS can be generalized to other algorithms.

3.1. Experiments environment

We test the performance of algorithms as well as the Multi-Armed Bandits Operator Selection (MAB-OS) on a set of benchmark functions containing unimodal, multimodal [39,40], and composite functions from CEC-2017 [41]. The details and two-dimensional landscapes (if applicable) of these 33 functions are tabulated and depicted in the tables and figures in Appendix B. The metaheuristic algorithms are based on stochastic calculations which affect the results of each run. Therefore, the optimizers and

algorithms are evaluated multiple times on each benchmark function and the average and standard deviation of the performances are reported. As a result, the obtained solution and convergence curve of each pair of algorithm-function is extracted over 30 independent runs.

As mentioned before, several metaheuristic algorithms, and especially recent swarm-based methods, have gone through a primary examination to select the base algorithms for the framework. We use Python and the implementation of the algorithms are based on EvoloPy package [42,43]. In addition to HHO, DE, and WOA that are selected, we also considered algorithms like Grey Wolf Optimizer (GWO) [40], Moth-Flame Optimization Algorithm (MFO) [44], Multi-verse Optimizer (MVO) [45], Salp Swarm Algorithm (SSA) [46], and Sine Cosine Algorithm (SCA) [47]. Note that the convergence curves of these methods on some benchmark functions are provided in Appendix A. It is especially observed that HHO performs better on unimodal functions than low dimensional multimodal or composite functions. On the other hand, DE has an outstanding performance on these types of landscapes. This observation is also aligned with the previous studies on these algorithms [48,49]. DE is equipped with exploration operators that make it a capable method for complex landscapes. On the other hand, HHO has excellent built-in local search strategies that result in its good performance in simpler unimodal functions. Table 1 contains the parameters that are used for the experiments in this work.

Before discussing the effectiveness of the framework in obtaining good final solutions for the optimization problems, we need to discuss its efficiency in terms of computation time and fitness evaluations. The time budget for the optimization problems is considered to be limited and the aim of these methods usually is to provide fast convergence to good enough solutions. In the context of algorithm selection, spending time on the examination and selection of algorithms would reduce the time for the optimization itself. Hence, we are usually in need for examination and selection methods that are not too time-consuming. An important remark here is that in MAB-OS, there is no need

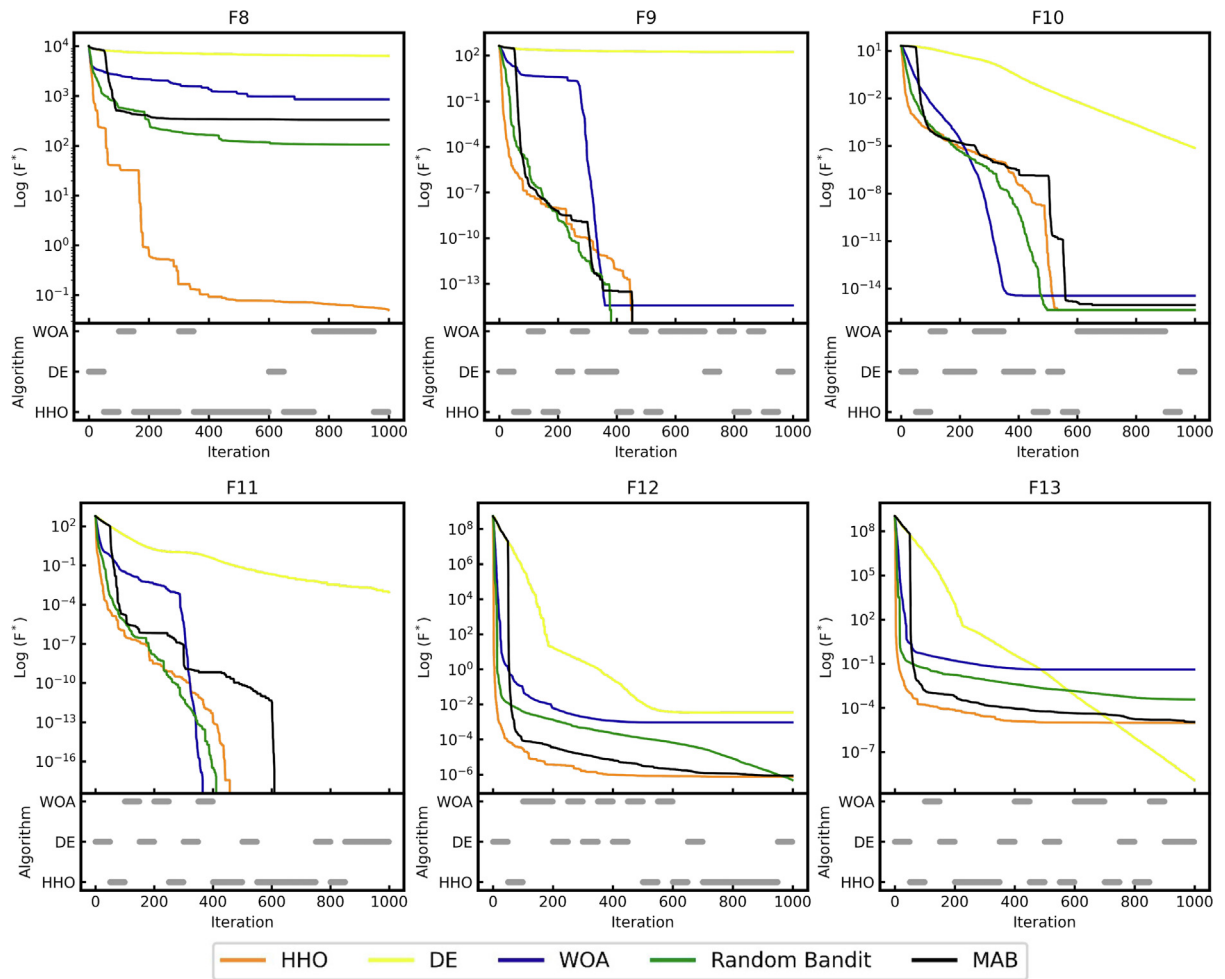


Fig. 3. The performance of multi-armed bandit (MAB) between Differential Evolution (DE), Harris Hawks Optimizer (HHO), and Whale Optimization Algorithm (WOA) compared to the vanilla DE, HHO, and WOA algorithms, and random bandit selection baseline on high-dimensional multimodal fitness landscapes.

Table 1

The details of the parameters used for the optimization of each of the algorithms in the experiments.

| Algorithm | Parameters | Value |
|----------------|----------------------------|--------------------------|
| HHO | E | Linear from 2 to 0 |
| | LF β | 1.5 |
| DE | Mutation Factor (F) | 0.5 |
| | Crossover Ratio (CR) | 0.7 |
| WOA | a | Linear from 2 to 0 |
| | a_2 | Linear from -1 to -2 |
| MAB | K | 50 |
| | c | 1 |
| | α | 0.5 |
| All algorithms | Max. Iteration (T) | 1000 |
| | No. search agents (N) | 50 |
| | Dimension ^a (D) | 30 |

^aFor the variable dimensional benchmark functions.

for additional iterations or multi-agent computations. Moreover, there is no change in the number of fitness evaluations with the MAB-OS framework as it does not require any extra evaluation.

We analyzed the wall-clock run time for the base algorithms and the proposed implementation of MAB-OS, and noticed that the extra operations including the computation of Q values and algorithm selection do not cause any significant computation time added to the normal run time of the algorithms. Please note

that the base algorithms do not have same average wall-clock time as they consist of different operations, so we cannot expect the MAB-OS to be more efficient than each of them, but it is interestingly efficient compared to the average time over the base algorithms.

3.2. Convergence analysis

The first evaluation of the performance of the framework is by analyzing the convergence curves on the introduced benchmark functions. Here, we compare the MAB-OS convergence behavior with those of base algorithms, i.e. HHO, DE, and WOA, as well as a random bandit selection baseline. In the random baseline, one of the algorithms are selected in a uniformly random manner and the positions are updated with that algorithm's update rule. This baseline would indicate the effect of non-intelligent and non-systematic online hybridization of algorithms.

We start with the unimodal functions, F1–F7, where Fig. 2 shows the convergence curves of the algorithms and MAB frameworks (top subplot) and the distribution of the active algorithm in an example trial for the MAB-OS (bottom subplot). The MAB-OS does not seem efficient in this type of functions compared to the best algorithm which is usually HHO, or WOA (e.g. on F1 and F2). The reason for this inferior performance is assumed to be due to the fact that in unimodal functions, exploitation is a suitable strategy that brings about fast convergence to the very small values around the only local minimum, which is also

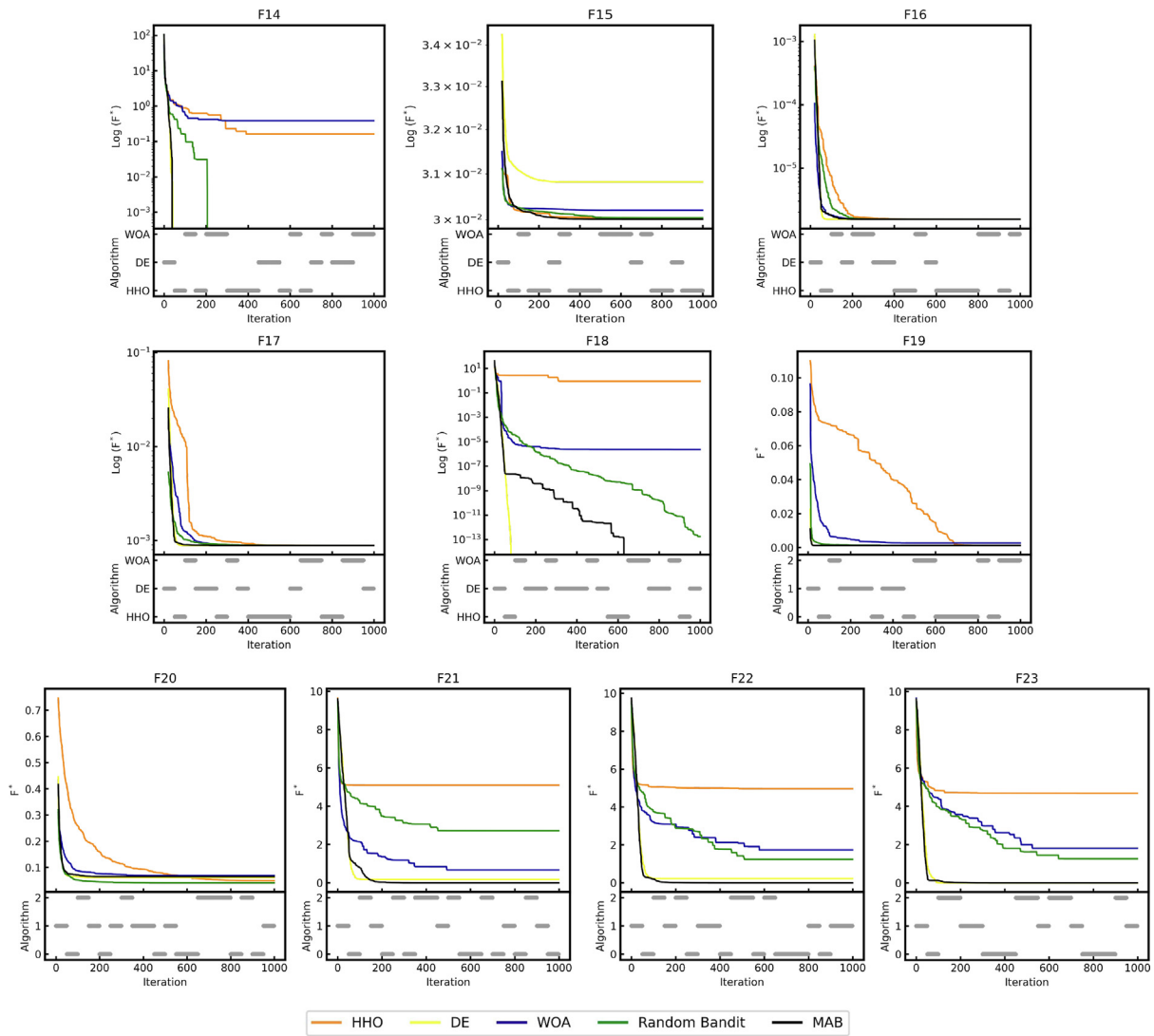


Fig. 4. The performance of multi-armed bandit (MAB) between Differential Evolution (DE), Harris Hawks Optimizer (HHO), and Whale Optimization Algorithm (WOA) compared to the vanilla DE, HHO, and WOA algorithms, and random bandit selection baseline on fixed-dimensional multimodal fitness landscapes.

the global optimum. As a result, the winning strategy focuses on one specific superior update rule and does not lose any chance for the improvement. This is probably the main reason for the linear (in logarithmic scale) decrease of the curves in many of these functions (e.g. WOA curve in F1). For MAB-OS, however, this cannot be the case since it has to spend time on the evaluation of all the constituting algorithms. Spending iterations on the sub-par algorithms, e.g. DE for some unimodal functions, would hinder the MAB-OS convergence rate to be as fast as the winning algorithm. In fact, we consider this as a good indication of not easily trapping in locally optimal solutions when using MAB-OS, which will be investigated on multi-modal and composite test functions. One might argue that such functions are more similar to challenging, real-world optimization problems.

The convergence behavior of algorithms on high dimensional multimodal functions are depicted in Fig. 3. There is no absolute winner on this type of landscape, however, HHO and WOA are still performing better in most cases. We can see that MAB and random bandit also usually reach the solutions that HHO achieves. In F8, HHO outperforms with high margin, and its reason is similar to the unimodal case. In fact, the consistency in the single update rule can sometimes lead to never-ceasing improvement. An interesting observation in F8 is that the MAB-OS also insists on using HHO in the majority of the iterations due

to its performance. Generally speaking, there is a relatively high randomness in the behavior of algorithms on these functions that strongly depends on a variety of parameters such as initialization and local minima conditions. This leads to the observation that while MAB algorithms are performing relatively good here, there is no meaningful difference between the random bandit and MAB in these functions.

The next type of functions are fixed dimensional multimodal functions which usually have much lower dimensions but with different modalities that can trap algorithms in locally optimal solutions (Fig. 4). In such functions, DE has a better performance compared to its performance in the previous functions. In fact, the exploration capability of DE makes it converge to better solutions in many of these multimodal landscapes. This is while HHO shows inferior behavior now compared to its performance on unimodal functions. Also, the superiority of intelligent MAB-OS compared to random hybridization is observed in most of the cases. Especially, in F21–F23, we can observe how MAB has a similar performance to the winning algorithm DE while random bandit and WOA are in the middle and HHO shows the worst behavior.

Fig. 5 illustrates the convergence curves for the composite functions that have challenging landscape combined from different types of landscapes. This feature complicates keeping balance

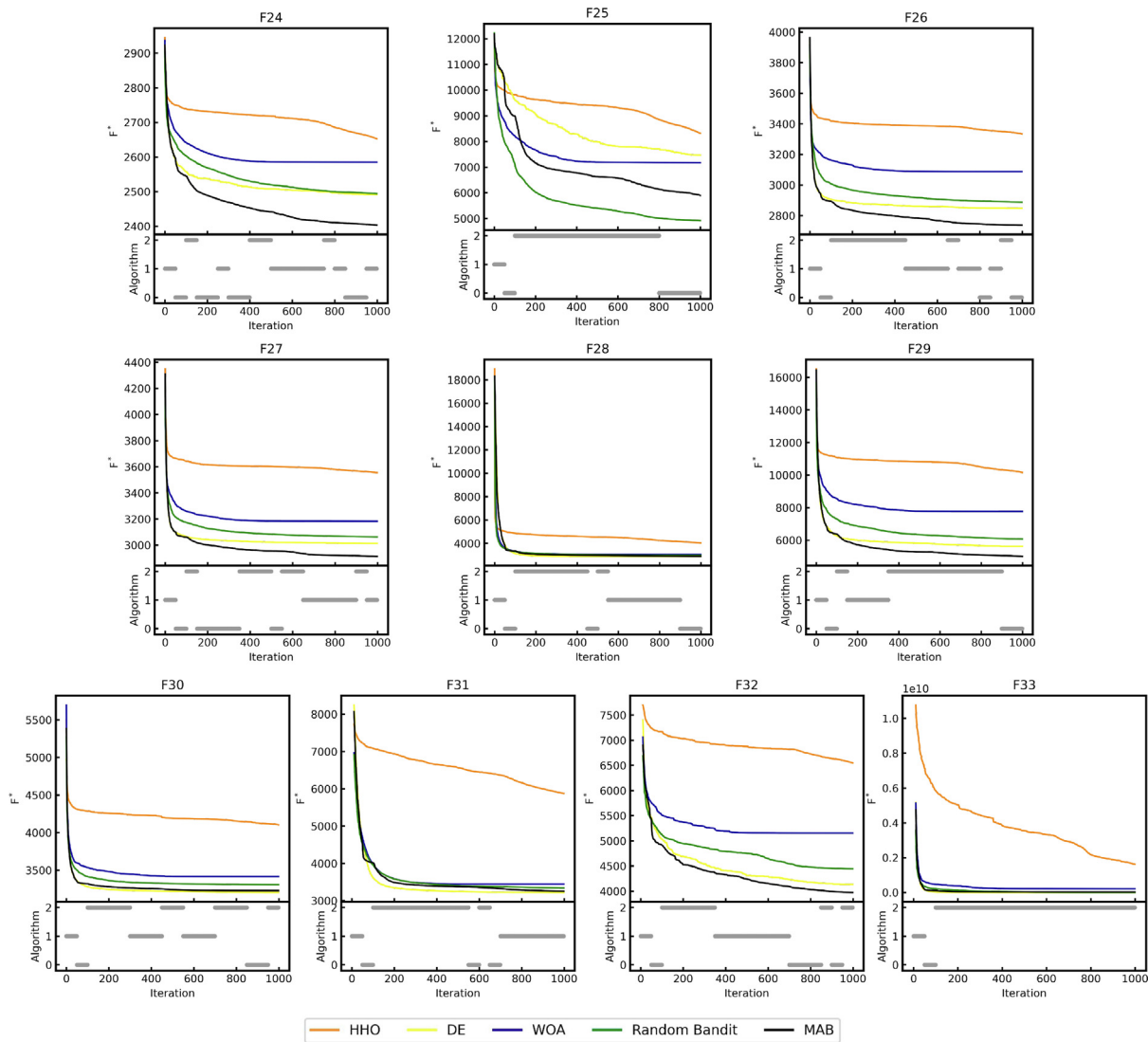


Fig. 5. The performance of multi-armed bandit (MAB) between Differential Evolution (DE), Harris Hawks Optimizer (HHO), and Whale Optimization Algorithm (WOA) compared to the vanilla DE, HHO, and WOA algorithms, and random bandit selection baseline on composite fitness landscapes.

between exploration and exploitation. As a result, algorithm selection techniques that rely on landscape features would possibly have difficulties to select the right choice of algorithm for these functions. However, MAB-OS only depends on the rewards collected from the function values which makes it independent from the landscape features. The inferior performance of HHO is again observable in such functions. Similar to fixed dimension multimodal functions, DE has a good performance on the majority of these problems.

It is evident that MAB follows the outstanding behavior of DE and WOA in these functions and relies less on HHO. An interesting observation is that generally MAB uses HHO in much less number of iterations compared to previous functions. Another surprising observation is that MAB can outperform all of the base algorithms in some functions. For example in F24, F27, and F32, MAB shows the best convergence rate and final solution. This indicates that MAB is able to combine the merits of different algorithms in a synergistic manner that leads to performing beyond the underlying base algorithms.

In the previous convergence plots, an example of selected algorithms were shown in the subplots. To extend on this analysis, we measure the percentage of times each algorithm was selected during the optimization. As it can be shown in the subplots of

Fig. 5, HHO algorithm (shown as algorithm '0') is selected less than the other two algorithms. The average percentage of using HHO drops from around 32% to 27% when moving from unimodal functions to the composite functions where HHO does not show a satisfactory performance. A remark here is that this percentage is not a sufficient metric to evaluate the framework since the order by which these algorithms are selected is also very important as discussed before.

3.3. Performance evaluation and statistical analysis

Now that the convergence rates, on average, are observed and discussed, we provide the tabular results on the final obtained solutions by the proposed framework compared to the base algorithms. We evaluate the algorithms with different initialization conditions and random seeds to have a fair comparison. The quantity of interest is the best solution that the population has achieved at each point of the optimization (F^*). We provide the average and standard deviation of this value over 30 runs. Also, the best performance of these runs is reported as a measure of the potential capability of the algorithms. The reason is that some algorithms have more consistent, i.e. less variance, behaviors while others have more stochastic, i.e. high variance, performances.

Table 2

Performance table of unimodal test functions. Average, standard deviation, and the best obtained solutions are reported on each test function for all the baseline algorithms as well as random selection and multi-armed bandit algorithm selection.

| Objective function | | DE | HHO | WOA | RANDOM | MAB |
|--------------------|------|-------------------|--------------------|--------------------|-------------|-------------------|
| F1 | Ave | 3.8485E-10 | 5.4443E-133 | 6.2894E-172 | 2.6749E-114 | 6.1063E-56 |
| | Std | 1.6871E-10 | 2.9109E-132 | 0.00 | 1.4404E-113 | 3.2883E-55 |
| | Best | 1.1296E-10 | 2.2541E-147 | 2.1945E-187 | 1.8336E-156 | 2.9331E-192 |
| F2 | Ave | 5.9057E-06 | 3.1109E-69 | 9.4168E-110 | 1.4553E-67 | 5.0403E-32 |
| | Std | 2.1499E-06 | 1.3889E-68 | 3.2983E-109 | 5.4537E-67 | 2.7143E-31 |
| | Best | 2.0261E-06 | 3.7224E-78 | 2.7212E-117 | 2.2937E-84 | 1.5031E-107 |
| F3 | Ave | 1.6513E+04 | 3.3200E-108 | 1.1920E+04 | 1.2984E-78 | 5.0697E-13 |
| | Std | 2.8884E+03 | 1.4293E-107 | 7.3915E+03 | 6.9920E-78 | 2.2117E-12 |
| | Best | 8.5027E+03 | 7.6186E-130 | 1.5087E+03 | 9.8246E-109 | 1.2515E-143 |
| F4 | Ave | 1.4803E+00 | 2.0502E-65 | 2.3074E+01 | 2.8650E-56 | 4.6356E-10 |
| | Std | 7.4467E-01 | 8.9544E-65 | 2.3223E+01 | 1.5337E-55 | 2.0187E-09 |
| | Best | 5.9087E-01 | 5.8329E-77 | 4.5818E-04 | 8.6694E-71 | 1.7562E-70 |
| F5 | Ave | 2.4287E+01 | 9.0637E-04 | 2.6587E+01 | 9.9512E-03 | 1.6498E-03 |
| | Std | 6.7448E-01 | 1.2677E-03 | 3.0543E-01 | 7.8235E-03 | 2.2999E-03 |
| | Best | 2.2817E+01 | 9.7848E-07 | 2.6023E+01 | 7.5730E-04 | 1.4117E-06 |
| F6 | Ave | 2.6214E-10 | 1.7254E-05 | 3.9657E-03 | 3.4340E-06 | 1.2326E-05 |
| | Std | 1.4904E-10 | 2.5979E-05 | 2.4786E-03 | 3.1424E-06 | 2.7365E-05 |
| | Best | 9.1883E-11 | 2.8510E-10 | 1.7987E-03 | 3.5143E-07 | 5.1433E-11 |
| F7 | Ave | 2.1427E-02 | 7.9972E-05 | 6.4471E-04 | 1.4265E-04 | 1.4964E-04 |
| | Std | 5.9699E-03 | 9.6293E-05 | 5.9913E-04 | 1.6949E-04 | 1.8282E-04 |
| | Best | 1.1012E-02 | 3.8186E-06 | 5.1377E-05 | 1.5354E-05 | 6.1500E-06 |

Table 3

Performance table of high-dimensional multimodal test functions. Average, standard deviation, and the best obtained solutions are reported on each test function for all the baseline algorithms as well as random selection and multi-armed bandit algorithm selection.

| Objective Function | | DE | HHO | WOA | RANDOM | MAB |
|--------------------|------|-------------------|-------------------|-------------------|-------------------|-------------------|
| F8 | Ave | 6.3708E+03 | 5.0348E-02 | 8.5920E+02 | 1.0601E+02 | 3.3269E+02 |
| | Std | 2.9597E+02 | 7.9861E-02 | 1.0619E+03 | 3.1223E+02 | 1.0699E+03 |
| | Best | 5.5537E+03 | 3.8183E-04 | 6.6231E-02 | 1.0721E-03 | 6.1516E-04 |
| F9 | Ave | 1.7049E+02 | 0.00 | 3.7896E-15 | 0.00 | 0.00 |
| | Std | 1.2061E+01 | 0.00 | 2.0407E-14 | 0.00 | 0.00 |
| | Best | 1.2340E+02 | 0.00 | 0.00 | 0.00 | 0.00 |
| F10 | Ave | 7.6018E-06 | 4.4409E-16 | 3.5231E-15 | 4.4409E-16 | 9.1778E-16 |
| | Std | 3.2677E-06 | 9.8608E-32 | 2.1964E-15 | 9.8608E-32 | 1.5166E-15 |
| | Best | 3.2440E-06 | 4.4409E-16 | 4.4409E-16 | 4.4409E-16 | 4.4409E-16 |
| F11 | Ave | 8.9137E-04 | 0.00 | 0.00 | 0.00 | 0.00 |
| | Std | 3.3461E-03 | 0.00 | 0.00 | 0.00 | 0.00 |
| | Best | 1.9976E-10 | 0.00 | 0.00 | 0.00 | 0.00 |
| F12 | Ave | 3.4556E-03 | 7.6125E-07 | 9.4544E-04 | 4.8148E-07 | 8.6700E-07 |
| | Std | 1.8609E-02 | 7.4942E-07 | 1.4927E-03 | 2.9504E-07 | 2.1891E-06 |
| | Best | 4.8539E-11 | 1.6914E-10 | 1.6506E-04 | 4.6654E-08 | 3.8436E-11 |
| F13 | Ave | 1.2293E-09 | 9.6779E-06 | 4.0308E-02 | 3.7319E-04 | 1.0974E-05 |
| | Std | 8.5030E-10 | 1.6950E-05 | 4.9275E-02 | 1.9740E-03 | 3.9282E-05 |
| | Best | 1.1923E-10 | 2.1571E-08 | 3.3570E-03 | 1.4183E-06 | 1.4592E-09 |

We employ two metrics to compare the algorithms in this section. The first metric is based on the comparison of the statistical behavior of the algorithms. This metric considers all of the trials and examines if there is a winning algorithm for each pair of algorithms. To this end, we use paired t-test with significance level of $\alpha = 0.05$ and rank them based on the pair results of this test. This metric evaluates whether an algorithm's better performance is statistically meaningful or not and shows the consistency of the performance. The second metric, focuses only on the best performance, i.e. one trial, of the algorithms which can represent the potential outcome of using a specific algorithm. We can then rank the algorithms for each function based on this best obtained solution.

The final solutions for the unimodal functions F1–F7 are tabulated in Table 2. The results indicate that HHO has the best overall performance. The random online hybridization and bandit also have relatively good results that is much better than the worst algorithm but not as well as the best algorithm. This is in alignment with the observations from the convergence curves.

Tables 3 and 4 report the results for the high dimensional and fixed dimensional multimodal functions, respectively. As discussed in the previous section, we can see that DE has a better performance on low dimensional multimodal landscapes and the MAB-OS is also good enough in most of the cases and performs almost as well as the winning algorithm for both average and the best performances.

At last, Table 5 shows the obtained solutions for the composition benchmark functions. Here, we can specifically observe that MAB-OS can not only perform as well as the winning algorithm which is DE in majority of the cases but also outperforms it and achieves the first rank among the algorithms.

To show the overall performance of the MAB-OS, we compute the relative ranking of the algorithms for different types of functions based on the aforementioned metrics (Fig. 6). The results capture the discussed results about the performance of HHO and DE in different types of landscapes where each of them are good only in specific types of landscapes. MAB, however, have a good rank on all types of functions and on average achieves the first rank. The order of the rankings of the algorithms over all of the

Table 4

Performance table of fixed-dimensional multimodal test functions. Average, standard deviation, and the best obtained solutions are reported on each test function for all the baseline algorithms as well as random selection and multi-armed bandit algorithm selection.

| Objective Function | | DE | HHO | WOA | RANDOM | MAB |
|--------------------|------|--------------------|--------------------|--------------------|--------------------|--------------------|
| F14 | Ave | -1.9962E-03 | 1.6237E-01 | 3.8965E-01 | -1.9962E-03 | -1.9962E-03 |
| | Std | 4.3368E-19 | 8.8511E-01 | 1.7766E+00 | 4.3368E-19 | 4.3368E-19 |
| | Best | -1.9962E-03 | -1.9962E-03 | -1.9962E-03 | -1.9962E-03 | -1.9962E-03 |
| F15 | Ave | 3.0823E-02 | 3.0008E-02 | 3.0201E-02 | 3.0038E-02 | 3.0007E-02 |
| | Std | 3.5890E-03 | 3.7100E-07 | 2.4901E-04 | 1.6437E-04 | 4.8341E-09 |
| | Best | 3.0007E-02 | 3.0007E-02 | 3.0009E-02 | 3.0007E-02 | 3.0007E-02 |
| F16 | Ave | 1.5465E-06 | 1.5465E-06 | 1.5465E-06 | 1.5465E-06 | 1.5465E-06 |
| | Std | 4.2352E-22 | 1.1248E-11 | 6.7713E-11 | 1.0378E-15 | 8.2753E-17 |
| | Best | 1.5465E-06 | 1.5465E-06 | 1.5465E-06 | 1.5465E-06 | 1.5465E-06 |
| F17 | Ave | 8.8736E-04 | 8.8743E-04 | 8.8755E-04 | 8.8736E-04 | 8.8736E-04 |
| | Std | 2.1684E-19 | 9.8602E-08 | 3.6067E-07 | 2.1684E-19 | 2.6926E-10 |
| | Best | 8.8736E-04 | 8.8736E-04 | 8.8736E-04 | 8.8736E-04 | 8.8736E-04 |
| F18 | Ave | -7.8426E-14 | 9.0000E-01 | 2.3582E-06 | 1.7599E-13 | -7.7360E-14 |
| | Std | 7.9936E-16 | 4.8466E+00 | 4.4223E-06 | 4.3999E-13 | 4.0917E-15 |
| | Best | -8.0824E-14 | -7.4163E-14 | 4.7904E-12 | -7.4163E-14 | -8.0380E-14 |
| F19 | Ave | 1.2179E-03 | 1.2199E-03 | 2.6664E-03 | 1.2179E-03 | 1.2179E-03 |
| | Std | 2.1684E-19 | 2.5256E-06 | 2.0834E-03 | 2.1684E-19 | 2.1684E-19 |
| | Best | 1.2179E-03 | 1.2179E-03 | 1.2185E-03 | 1.2179E-03 | 1.2179E-03 |
| F20 | Ave | 6.1414E-02 | 4.9699E-02 | 6.9192E-02 | 4.1599E-02 | 6.5378E-02 |
| | Std | 5.9314E-02 | 5.8990E-02 | 1.0629E-01 | 5.7294E-02 | 5.8916E-02 |
| | Best | -1.9952E-03 | -1.9786E-03 | -1.9930E-03 | -1.9952E-03 | -1.9952E-03 |
| F21 | Ave | 1.6841E-01 | 5.0980E+00 | 6.7321E-01 | 2.7189E+00 | 4.3476E-07 |
| | Std | 9.0694E-01 | 2.1346E-05 | 2.0460E+00 | 2.5433E+00 | 6.1290E-07 |
| | Best | 3.2094E-07 | 5.0980E+00 | 7.1072E-06 | 3.2094E-07 | 3.2094E-07 |
| F22 | Ave | 2.2248E-01 | 4.9608E+00 | 1.7331E+00 | 1.2401E+00 | -1.4054E-04 |
| | Std | 1.1989E+00 | 1.3258E+00 | 2.9301E+00 | 2.2481E+00 | 1.6090E-07 |
| | Best | -1.4057E-04 | 1.2712E-04 | -1.2147E-04 | -1.4057E-04 | -1.4057E-04 |
| F23 | Ave | -1.0982E-04 | 4.6869E+00 | 1.8077E+00 | 1.2617E+00 | -1.0967E-04 |
| | Std | 4.0658E-20 | 1.8382E+00 | 3.0929E+00 | 2.2873E+00 | 8.0408E-07 |
| | Best | -1.0982E-04 | 8.2292E-05 | -6.3864E-05 | -1.0982E-04 | -1.0982E-04 |

Table 5

Performance table of composite test functions. Average, standard deviation, and the best obtained solutions are reported on each test function for all the baseline algorithms as well as random selection and multi-armed bandit algorithm selection.

| Objective Function | | DE | HHO | WOA | RANDOM | MAB |
|--------------------|------|-------------------|------------|------------|-------------------|-------------------|
| F24 | Ave | 2.4909E+03 | 2.6521E+03 | 2.5847E+03 | 2.4941E+03 | 2.4033E+03 |
| | Std | 1.3890E+01 | 5.3335E+01 | 6.0695E+01 | 5.3647E+01 | 3.4108E+01 |
| | Best | 2.4687E+03 | 2.5139E+03 | 2.4869E+03 | 2.3826E+03 | 2.3262E+03 |
| F25 | Ave | 7.4671E+03 | 8.3213E+03 | 7.1753E+03 | 4.9134E+03 | 5.8990E+03 |
| | Std | 3.3703E+03 | 6.8934E+02 | 2.1715E+03 | 2.4121E+03 | 2.2190E+03 |
| | Best | 2.3000E+03 | 6.9565E+03 | 2.3889E+03 | 2.3277E+03 | 2.3001E+03 |
| F26 | Ave | 2.8474E+03 | 3.3338E+03 | 3.0875E+03 | 2.8889E+03 | 2.7384E+03 |
| | Std | 8.6431E+00 | 1.0928E+02 | 7.7636E+01 | 7.0576E+01 | 2.1240E+01 |
| | Best | 2.8300E+03 | 3.1648E+03 | 2.9733E+03 | 2.7853E+03 | 2.6996E+03 |
| F27 | Ave | 3.0123E+03 | 3.5547E+03 | 3.1828E+03 | 3.0644E+03 | 2.9148E+03 |
| | Std | 1.0713E+01 | 1.2507E+02 | 1.2140E+02 | 6.5126E+01 | 2.6482E+01 |
| | Best | 2.9871E+03 | 3.3059E+03 | 2.9996E+03 | 2.9404E+03 | 2.8499E+03 |
| F28 | Ave | 2.8871E+03 | 4.0346E+03 | 3.0518E+03 | 2.9627E+03 | 2.9028E+03 |
| | Std | 1.0804E-01 | 2.7317E+02 | 5.8246E+01 | 2.9819E+01 | 3.7931E+01 |
| | Best | 2.8868E+03 | 3.5207E+03 | 2.9566E+03 | 2.8974E+03 | 2.8835E+03 |
| F29 | Ave | 5.6242E+03 | 1.0164E+04 | 7.7713E+03 | 6.0750E+03 | 5.0106E+03 |
| | Std | 9.5219E+01 | 1.0697E+03 | 1.2860E+03 | 1.2786E+03 | 4.5631E+02 |
| | Best | 5.3998E+03 | 8.2942E+03 | 3.4927E+03 | 2.9841E+03 | 4.0595E+03 |
| F30 | Ave | 3.2033E+03 | 4.1035E+03 | 3.4124E+03 | 3.3068E+03 | 3.2267E+03 |
| | Std | 6.5243E+00 | 3.0647E+02 | 1.0373E+02 | 5.3156E+01 | 1.8775E+01 |
| | Best | 3.1908E+03 | 3.5440E+03 | 3.2589E+03 | 3.2377E+03 | 3.1992E+03 |
| F31 | Ave | 3.2214E+03 | 5.8755E+03 | 3.4482E+03 | 3.3439E+03 | 3.2561E+03 |
| | Std | 1.8212E+01 | 5.7905E+02 | 7.6608E+01 | 4.7560E+01 | 4.7881E+01 |
| | Best | 3.2087E+03 | 4.3236E+03 | 3.3101E+03 | 3.2623E+03 | 3.2102E+03 |
| F32 | Ave | 4.1334E+03 | 6.5483E+03 | 5.1554E+03 | 4.4442E+03 | 3.9744E+03 |
| | Std | 1.6838E+02 | 1.0399E+03 | 5.1367E+02 | 3.1625E+02 | 3.3804E+02 |
| | Best | 3.6610E+03 | 4.5833E+03 | 4.3048E+03 | 3.8496E+03 | 3.4411E+03 |
| F33 | Ave | 4.8839E+04 | 1.6160E+09 | 1.9993E+08 | 6.0830E+06 | 2.7951E+07 |
| | Std | 2.0401E+04 | 2.3742E+09 | 1.8260E+08 | 5.4657E+06 | 3.6157E+07 |
| | Best | 2.4048E+04 | 4.2898E+06 | 2.1828E+07 | 3.4244E+05 | 1.7539E+04 |

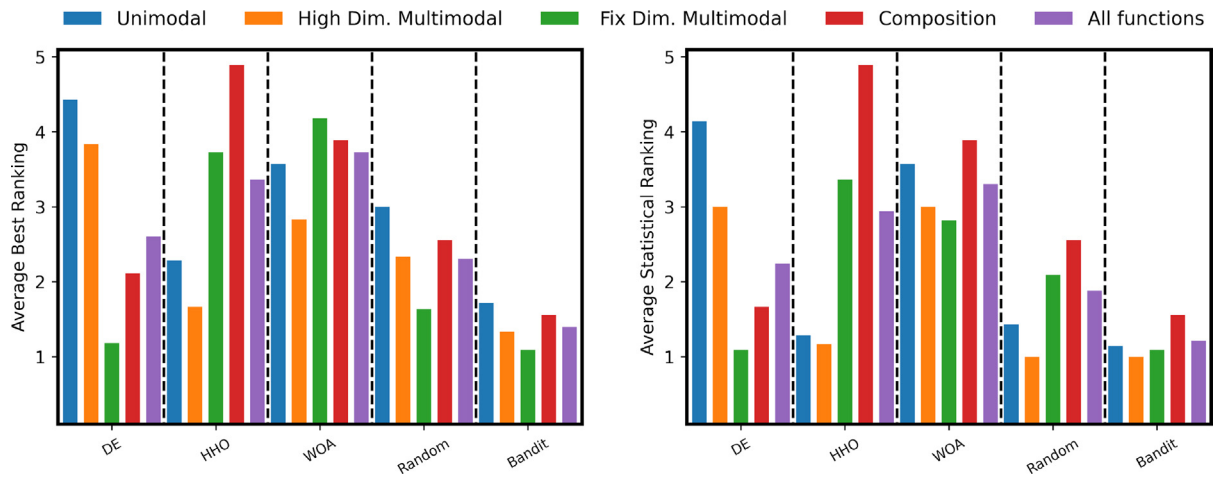


Fig. 6. The ranking of the base algorithms (HHO, DE, and WOA), random operator selection, and Multi-Armed Bandit Optimizer Selection (MAB-OS) for different types of landscapes: (a) Based on best performance and (b) Based on the statistical tests. The bar chart shows that Bandit, i.e. MAB, holds a good ranking in almost all functions while other base algorithms are only good in specific type of landscapes.

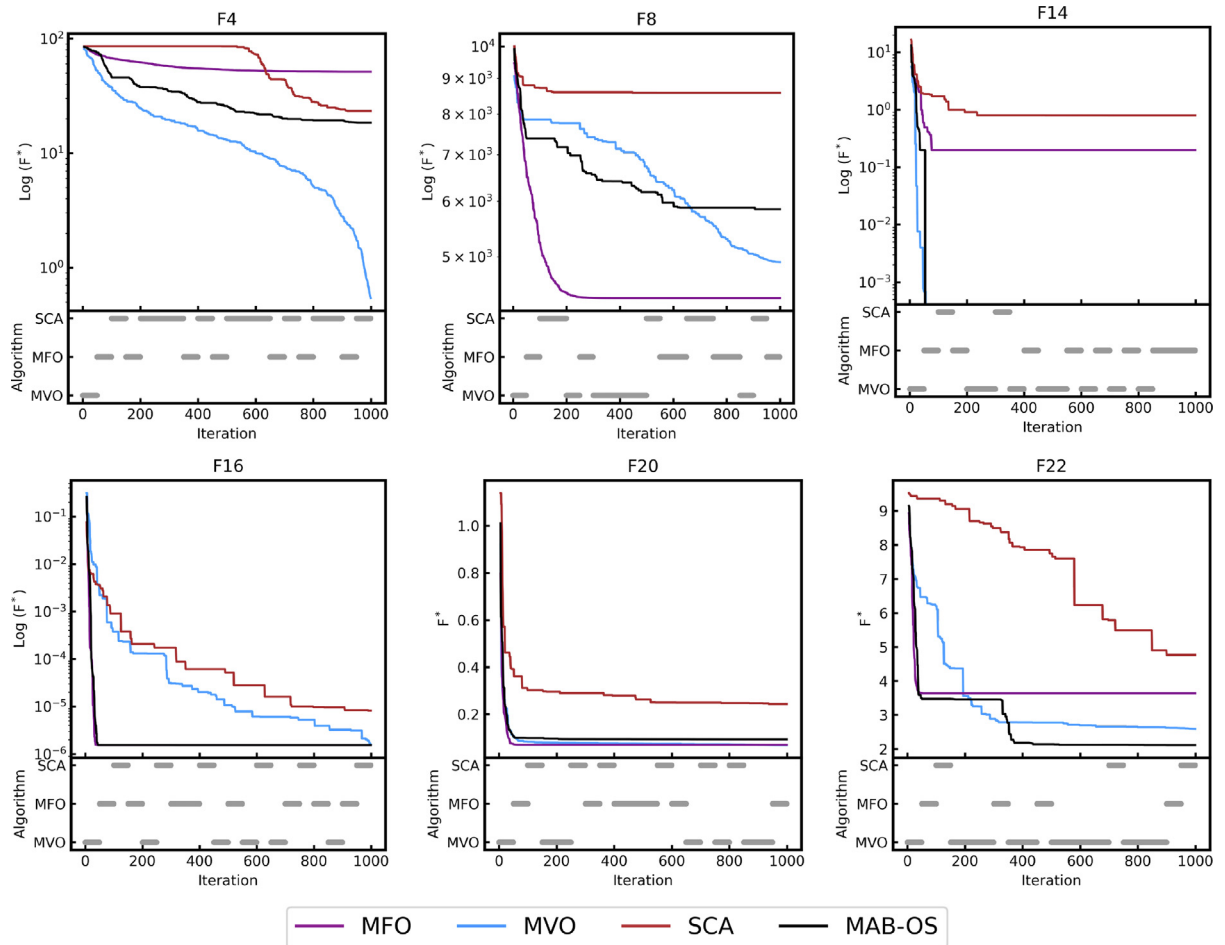


Fig. 7. The performance of multi-armed bandit (MAB) using moth-flame optimizer (MFO), multi-verse optimizer (MVO), and sine cosine algorithm (SCA) compared to the vanilla MFO, MVO, and SCA on some benchmark problems.

benchmark functions is as follows: MAB-OS < Random Bandits < DE < HHO < WOA.

Overall, the results of the convergence curves, the solution tables, and the ranking charts indicate that MAB-OS can provide a significant positive impact on the performance of the base algorithms by employing the advantages of each method in the right time. It can thus combine the local search capabilities of

HHO or WOA with the exploration strategies of DE in a systematic manner that is identified intelligently by reinforcement learning.

3.4. Generality of the framework

In this section, we show that the proposed MAB-OS can be used on different choices of base metaheuristic algorithms. To

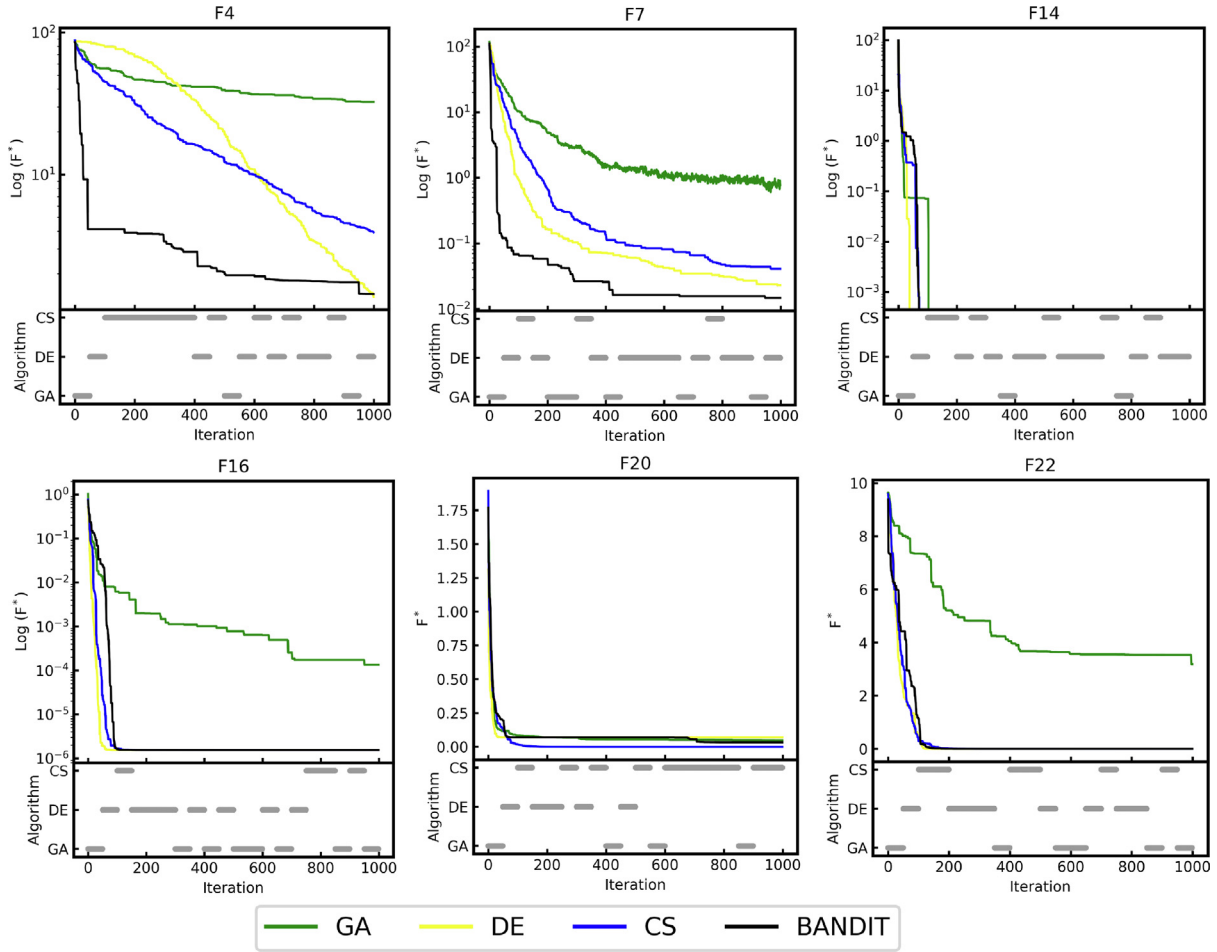


Fig. 8. The performance of multi-armed bandit (MAB) using Genetic Algorithm (GA), Differential Evolution (DE), and Cuckoo Search (CS) compared to the vanilla GA, DE, and CS on some benchmark problems.

show the generalizability of the framework, we conduct two more experiments on different groups of algorithms. In the first experiment, we select three algorithms of moth-flame optimizer (MFO) [44], multi-verse optimizer (MVO) [45], and sine-cosine algorithm (SCA) [47] and we denote this experiment as (MFO, MVO, SCA). In the second experiment, the base algorithms are Genetic Algorithm (GA) [1], Differential Evolution (DE) [2], and Cuckoo Search (CS) [50] and this experiment is denoted as (GA, DE, CS). An important remark in the first (MFO, MVO, SCA) experiment is that to let these three algorithms communicate with each other and optimize together, we need some extra book-keeping process that computes and keeps some parameters required by MFO in the other two algorithms. We observe that this extra computations are negligible in terms of computation time in this example, but it should be noted when selecting the base algorithms.

We apply MAB-OS on these two sets of three algorithms and on the 23 benchmark functions with unimodal and multimodal landscapes. In both experiments, we observe that the MAB-OS overall ranking is better than each of the base algorithms, meaning that if we have an unknown problem, MAB-OS is potentially the best choice. Fig. 7 shows the convergence curves for (MFO, MVO, SCA) on some of the benchmark problems and the overall statistical ranking (considering statistical ties) is: MAB-OS (1.087) < MFO (1.435) = MVO (1.435) < SCA (2.00). For the second experiment, Fig. 8 depicts the convergence curves on some of the benchmarks and the overall statistical ranking is: MAB-OS

(1.174) < DE (1.522) < CS (2.00) < GA (2.607). These two experiments show the generality of MAB-OS when applying to different choices of algorithms.

4. Conclusion

In this work, we proposed an online optimizer selection framework based on the multi-armed bandits problem as a classical reinforcement learning problem. In this framework, the behavior of the optimization algorithms during the optimization are evaluated and represented as estimated scores based on the rewards collected from their convergence curves. Then, the better algorithms have a higher chance of selection based on the soft sampling on the upper confidence bound of their scores. This technique results in adaptive operator selection without spending significant additional resources of time or fitness evaluation. The results of using this framework on the base algorithms containing HHO, DE, and WOA show that it can outperform them and achieve the best ranking on average which can be viewed as a new optimization algorithm with better performance without any manual changes on the update rules of base algorithms. While this work shows the effect of this framework, we should note that this framework has the potential to be used on different sets of base algorithms which can be an interesting direction for future works. Also, the Multi-Armed Bandit problem contains different parameters, reward definition, and implementation schemes which can be optimized for possible better results in the future works. Based

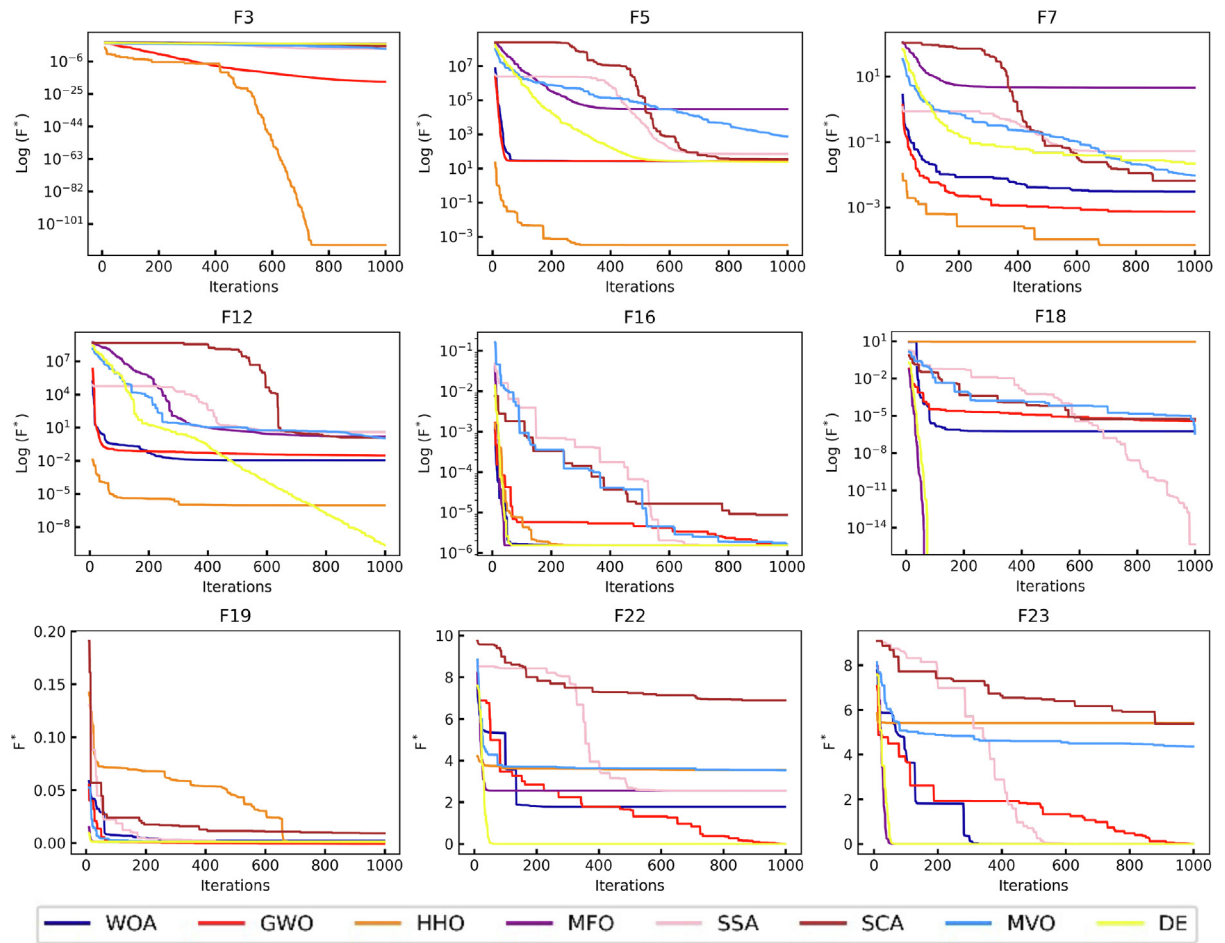


Fig. A.1. Comparison of the convergence curves of several metaheuristic optimization algorithms on some of the benchmark functions. We can observe that Harris Hawks Optimizer (HHO, orange curve), Differential Evolution (DE, yellow curve), and Whale Optimization Algorithm (WOA, Blue curve) have good performances among the evolutionary and swarm-based algorithms on these benchmark functions.

on the type of baseline algorithms and the problem environment, MAB-OS can be used in many different applications for efficient optimization as it attempts to combine time-efficiency and effective search for the best optimization algorithm. The data and codes used in this work are available at <https://github.com/BaratiLab/MAB-OS>.

CRedit authorship contribution statement

Kazem Meidani: Conceptualization, Methodology, Writing – original draft. **Seyedali Mirjalili:** Conceptualization, Writing – review & editing, Supervision. **Amir Barati Farimani:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Table B.1
Unimodal benchmark functions.

| Function | Range | Dim |
|--|-----------------|-----|
| $F_1(x) = \sum_{i=1}^N x_i^2$ | $[-100, 100]$ | 30 |
| $F_2(x) = \sum_{i=1}^N x_i + \prod_{i=1}^N x_i $ | $[-10, 10]$ | 30 |
| $F_3(x) = \sum_{i=1}^N (\sum_{j=1}^i x_j)^2$ | $[-100, 100]$ | 30 |
| $F_4(x) = \max_i \{ x_i , 1 \leq i \leq N\}$ | $[-100, 100]$ | 30 |
| $F_5(x) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | $[-30, 30]$ | 30 |
| $F_6(x) = \sum_{i=1}^N (x_i + 0.5)^2$ | $[-100, 100]$ | 30 |
| $F_7(x) = \sum_{i=1}^N ix_i^4 + \text{random}[0, 1)$ | $[-1.28, 1.28]$ | 30 |

Acknowledgments

This work is supported by the start-up fund provided by CMU Mechanical Engineering, United States and funding from National Science Foundation, USA (CBET-1953222), United States.

Appendix A. Performance of metaheuristic algorithms on some benchmark functions

See Fig. A.1.

Appendix B. Benchmark functions

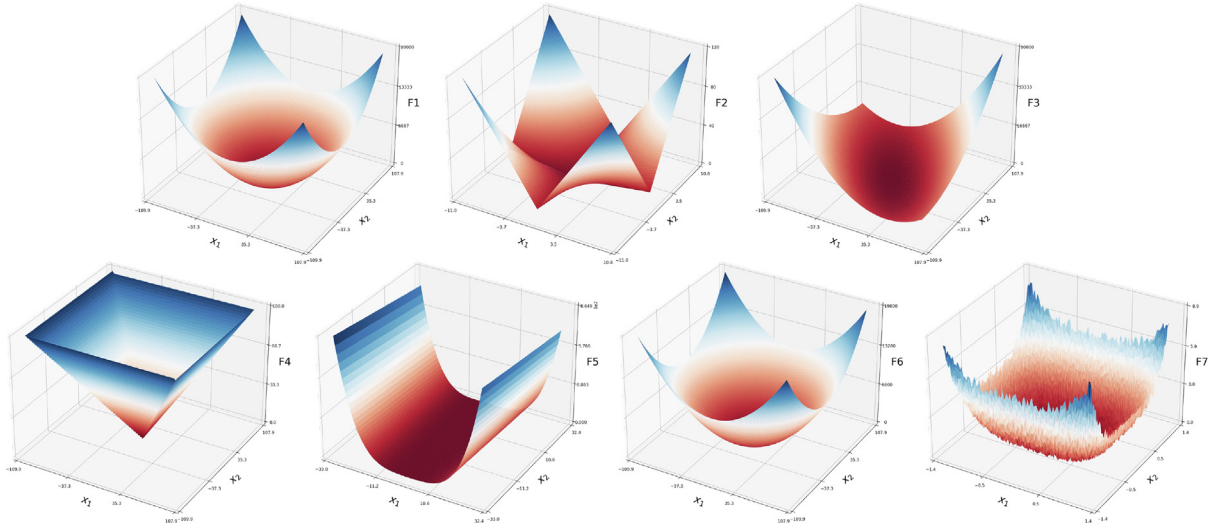


Fig. B.1. 2D version of unimodal test functions (30 dimensional versions are used for the experiments).

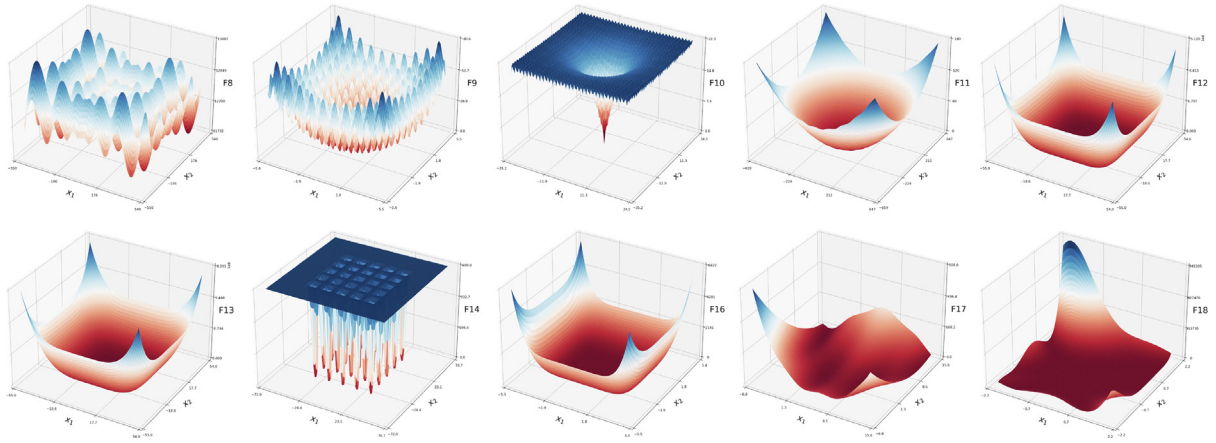


Fig. B.2. 2D version of some multimodal test functions (F_8 – F_{13}) and fixed dimension multimodal functions (F_{14} – F_{18}).

Table B.1 shows the functional form of the unimodal functions, and the multimodal functions are listed in Table B.2. For the details of composition functions, please read the full details on CEC-2017 benchmark [41] (see Figs. B.1 and B.2).

Appendix C. MAB-OS performance on higher dimensions

The convergence curves are shown for some of the unimodal and multimodal benchmark functions defined in a 100 dimensional search space. The results in the main manuscript were for 30 dimensions. This change applies only on the high dimensional benchmark functions. The other fixed dimension functions remain the same. The results qualitatively shown that the MAB-OS can be effective when applied to high dimensional landscapes (see Fig. C.1).

Table B.2
Multimodal benchmark functions.

| Function | Range | Dim |
|---|------------------|-----|
| $F_8(x) = \sum_{i=1}^N -x_i \sin \sqrt{ x_i } + 12\,569.487$ | $[-500, 500]$ | 30 |
| $F_9(x) = \sum_{i=1}^N [x_i^2 - 10 \cos 2\pi x_i + 10]$ | $[-5, 12.5, 12]$ | 30 |
| $F_{10}(x) = -20 \exp - 0.2 \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2} - \exp(\frac{1}{N} \sum_{i=1}^N \cos(2\pi x_i)) + 20 + e$ | $[-32, 32]$ | 30 |
| $F_{11}(x) = \frac{1}{4000} \sum_{i=1}^N x_i^2 - \prod_{i=1}^N \cos(\frac{x_i}{\sqrt{i}}) + 1$ | $[-600, 600]$ | 30 |
| $F_{12}(x) = \frac{\pi}{N} \{10 \sin(\pi y_1) + \sum_{i=1}^{N-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_N - 1)^2\} + \sum_{i=1}^N u(x_i, 10, 100, 4), \quad y_i = 1 + \frac{x_i + 1}{4}$ | $[-50, 50]$ | 30 |
| $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ | | |

(continued on next page)

Table B.2 (continued).

| Function | Range | Dim |
|--|-------------|-----|
| $F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^N (x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_N - 1)^2[1 + \sin^2(2\pi x_N)]\} + \sum_{i=1}^N u(x_i, 5, 100, 4)$ | $[-50, 50]$ | 30 |
| $F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6})^{-1} - 1$ | $[-65, 65]$ | 2 |
| $F_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2 + 0.0027$ | $[-5, 5]$ | 4 |
| $F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{5}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 + 1.03163$ | $[-5, 5]$ | 2 |
| $F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10 - 0.397$ | $[-5, 5]$ | 2 |
| $F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] - 3$ | $[-2, 2]$ | 2 |
| $F_{19}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^3 a_{ij}(x_j - p_{-ij})^2) + 3.864$ | $[1, 3]$ | 3 |
| $F_{20}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2) + 3.32$ | $[0, 1]$ | 6 |
| $F_{21}(x) = -\sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i]^{-1} + 10.1532$ | $[0, 10]$ | 4 |
| $F_{22}(x) = -\sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i]^{-1} + 10.4028$ | $[0, 10]$ | 4 |
| $F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1} + 10.5363$ | $[0, 10]$ | 4 |

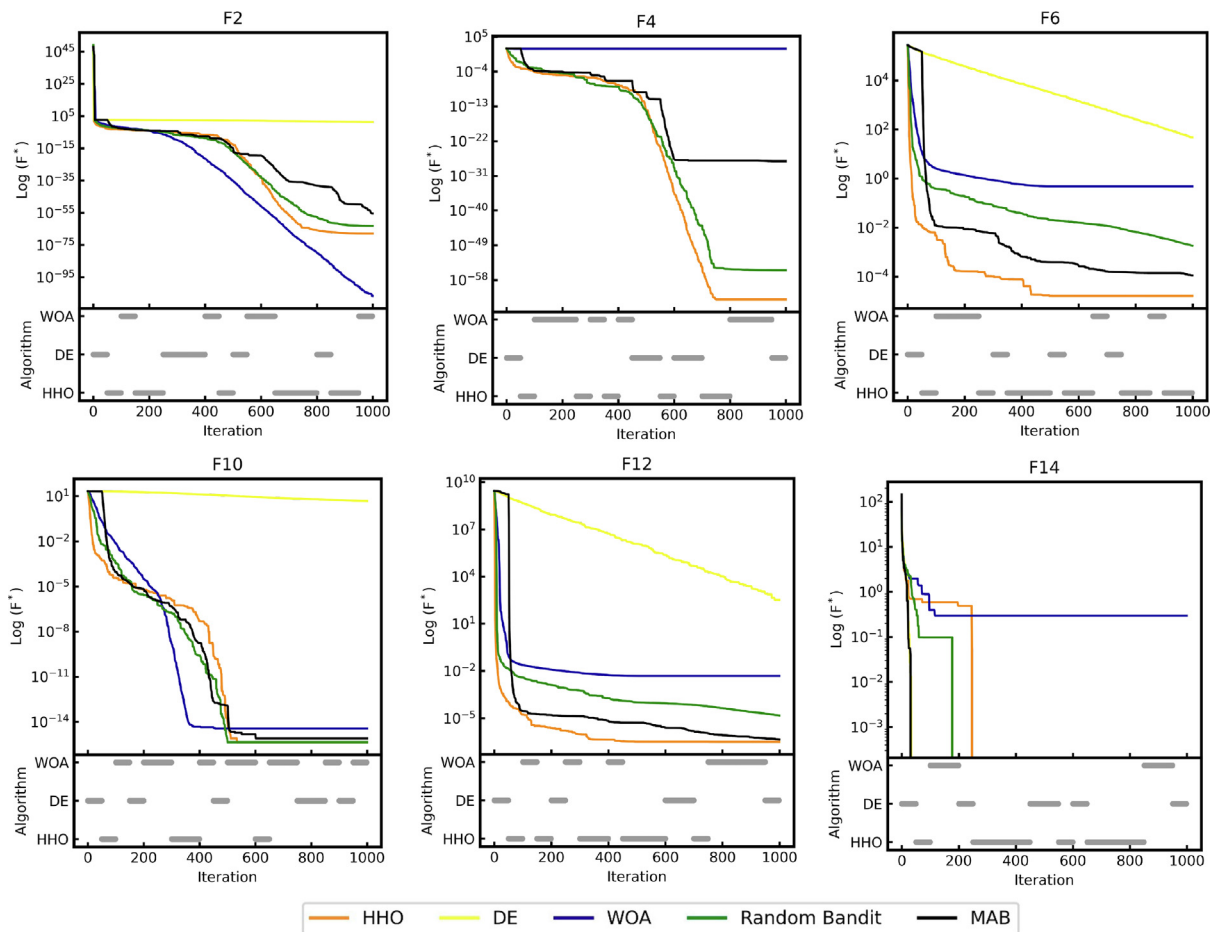


Fig. C.1. Comparison of the performance of MAB-OS with base algorithms of HHO, DE, and WOA, as well as random bandit on some 100 dimensional benchmark functions.

References

- [1] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, 1992.
- [2] R. Storn, K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.* 11 (4) (1997) 341–359, <http://dx.doi.org/10.1023/A:1008202821328>.
- [3] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, *Inform. Sci.* 179 (13) (2009) 2232–2248.
- [4] J. Kennedy, R. Eberhart, Particle swarm optimization, in: *Proceedings of ICNN'95-International Conference on Neural Networks*, Vol. 4, IEEE, 1995, pp. 1942–1948.
- [5] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Global Optim.* 39 (3) (2007) 459–471.
- [6] M. Abdel-Basset, L. Abdel-Fatah, A.K. Sangaiah, Chapter 10 - metaheuristic algorithms: A comprehensive review, in: A.K. Sangaiah, M. Sheng, Z. Zhang (Eds.), *Computational Intelligence for Multimedia Big Data on the*

- Cloud with Engineering Applications, in: *Intelligent Data-Centric Systems*, Academic Press, 2018, pp. 185–231, <http://dx.doi.org/10.1016/B978-0-12-813314-9.00010-4>, URL <https://www.sciencedirect.com/science/article/pii/B9780128133149000104>.
- [7] J.S. Soerensen, L. Johannesen, U. Grove, K. Lundhus, J.-P. Couderc, C. Graff, A comparison of IIR and wavelet filtering for noise reduction of the ECG, in: *2010 Computing in Cardiology*, IEEE, 2010, pp. 489–492.
 - [8] A. Hemmasian, K. Meidani, S. Mirjalili, A. Barati Farimani, Vecmetapy: A vectorized framework for metaheuristic optimization in python, *Adv. Eng. Softw.* 166 (2022) 103092, <http://dx.doi.org/10.1016/j.advengsoft.2022.103092>, URL <https://www.sciencedirect.com/science/article/pii/S0965997822000059>.
 - [9] P. Kerschke, H.H. Hoos, F. Neumann, H. Trautmann, Automated algorithm selection: Survey and perspectives, *Evol. Comput.* 27 (1) (2019) 3–45, http://dx.doi.org/10.1162/evco_a_00242.
 - [10] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.* 1 (1) (1997) 67–82.
 - [11] D. Yang, Z. Liu, J. Zhou, Chaos optimization algorithms based on chaotic maps with different probability distribution and search speed for global optimization, *Commun. Nonlinear Sci. Numer. Simul.* 19 (4) (2014) 1229–1246, <http://dx.doi.org/10.1016/j.cnsns.2013.08.017>, URL <https://www.sciencedirect.com/science/article/pii/S1007570413003675>.
 - [12] M. Jamil, H.-J. Zepernick, 3 - Lévy flights and global optimization, in: X.-S. Yang, Z. Cui, R. Xiao, A.H. Gandomi, M. Karamanoglu (Eds.), *Swarm Intelligence and Bio-Inspired Computation*, Elsevier, Oxford, 2013, pp. 49–72, <http://dx.doi.org/10.1016/B978-0-12-405163-8.00003-X>, URL <https://www.sciencedirect.com/science/article/pii/B978012405163800003X>.
 - [13] K. Meidani, A. Hemmasian, S. Mirjalili, A. Barati Farimani, Adaptive grey wolf optimizer, *Neural Comput. Appl.* (2022) <http://dx.doi.org/10.1007/s00521-021-06885-9>.
 - [14] Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization, *IEEE Trans. Syst. Man Cybern. B* 39 (6) (2009) 1362–1381, <http://dx.doi.org/10.1109/TSMCB.2009.2015956>.
 - [15] J.H. Drake, A. Kheiri, E. Özcan, E.K. Burke, Recent advances in selection hyper-heuristics, *European J. Oper. Res.* 285 (2) (2020) 405–428, <http://dx.doi.org/10.1016/j.ejor.2019.07.073>, URL <https://www.sciencedirect.com/science/article/pii/S0377221719306526>.
 - [16] J.M. Cruz-Duarte, I. Amaya, J.C. Ortiz-Bayliss, S.E. Conant-Pablos, H. Terashima-Marín, Y. Shi, Hyper-heuristics to customise metaheuristics for continuous optimisation, *Swarm Evol. Comput.* 66 (2021) 100935, <http://dx.doi.org/10.1016/j.swevo.2021.100935>, URL <https://www.sciencedirect.com/science/article/pii/S2210650221000961>.
 - [17] M. Scoczynski, D. Oliva, E. Rodríguez-Esparza, M. Delgado, R. Lüders, M.E. Yafarani, L. Ledo, M.A. Elaziz, M. Pérez-Cisneros, A selection hyperheuristic guided by Thompson sampling for numerical optimization, in: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Association for Computing Machinery, New York, NY, USA, 2021, pp. 1394–1402, URL <https://doi.org/10.1145/3449726.3463140>.
 - [18] M.R. Nakkala, A. Singh, A. Rossi, Multi-start iterated local search, exact and matheuristic approaches for minimum capacitated dominating set problem, *Appl. Soft Comput.* 108 (2021) 107437, <http://dx.doi.org/10.1016/j.asoc.2021.107437>, URL <https://www.sciencedirect.com/science/article/pii/S1568494621003604>.
 - [19] R.-E. Precup, R.-C. David, R.-C. Roman, A.-I. Szedlak-Stinean, E.M. Petriu, Optimal tuning of interval type-2 fuzzy controllers for nonlinear servo systems using slime mould algorithm, *Internat. J. Systems Sci.* (2021) 1–16, <http://dx.doi.org/10.1080/00207721.2021.1927236>, arXiv:https://doi.org/10.1080/00207721.2021.1927236.
 - [20] P. Kerschke, H. Trautmann, Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco, in: N. Bauer, K. Ickstadt, K. Lübke, G. Szepannek, H. Trautmann, M. Vichi (Eds.), *Applications in Statistical Computing – from Music Data Analysis to Industrial Quality Improvement*, in: *Studies in Classification, Data Analysis, and Knowledge Organization*, Springer, 2019, pp. 93–123, http://dx.doi.org/10.1007/978-3-030-25147-5_7.
 - [21] K. Meidani, S. Mirjalili, A. Barati Farimani, Online metaheuristic algorithm selection, *Expert Syst. Appl.* 201 (2022) 117058, <http://dx.doi.org/10.1016/j.eswa.2022.117058>, URL <https://www.sciencedirect.com/science/article/pii/S0957417422004729>.
 - [22] M.S. Gibbs, H.R. Maier, G.C. Dandy, Using characteristics of the optimisation problem to determine the genetic algorithm population size when the number of evaluations is limited, *Environ. Model. Softw.* 69 (C) (2015) 226–239, <http://dx.doi.org/10.1016/j.envsoft.2014.08.023>.
 - [23] S. Picek, D. Jakobovic, From fitness landscape to crossover operator choice, in: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, in: *GECCO '14*, Association for Computing Machinery, New York, NY, USA, 2014, pp. 815–822, <http://dx.doi.org/10.1145/2576768.2598320>.
 - [24] K.M. Sallam, S.M. Elsayed, R.A. Sarker, D.L. Essam, Landscape-assisted multi-operator differential evolution for solving constrained optimization problems, *Expert Syst. Appl.* 162 (2020) 113033, <http://dx.doi.org/10.1016/j.eswa.2019.113033>, URL <https://www.sciencedirect.com/science/article/pii/S095741741930750X>.
 - [25] T. Takahama, S. Sakai, Differential evolution with dynamic strategy and parameter selection by detecting landscape modality, in: *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8, <http://dx.doi.org/10.1109/CEC.2012.6256613>.
 - [26] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, *SCIENCE* 220 (4598) (1983) 671–680.
 - [27] T. Wauters, K. Verbeeck, P. De Causmaecker, G. Vanden Berghe, Boosting metaheuristic search using reinforcement learning, in: *Hybrid Metaheuristics*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 433–452, http://dx.doi.org/10.1007/978-3-642-30671-6_17.
 - [28] B.K.M. Powell, D. Machalek, T. Quah, Real-time optimization using reinforcement learning, *Comput. Chem. Eng.* 143 (2020) 107077, <http://dx.doi.org/10.1016/j.compchemeng.2020.107077>.
 - [29] A. Seyyedabbasi, R. Aliyev, F. Kiani, M.U. Gulle, H. Basyildiz, M.A. Shah, Hybrid algorithms based on combining reinforcement learning and metaheuristic methods to solve global optimization problems, *Knowl.-Based Syst.* 223 (2021) 107044, <http://dx.doi.org/10.1016/j.knsys.2021.107044>, URL <https://www.sciencedirect.com/science/article/pii/S0950705121003075>.
 - [30] D. Bouneffouf, I. Rish, A survey on practical applications of multi-armed and contextual bandits, 2019, arXiv:1904.10040.
 - [31] M. Gagliolo, J. Schmidhuber, Algorithm selection as a bandit problem with unbounded losses, in: C. Blum, R. Battiti (Eds.), *Learning and Intelligent Optimization*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 82–96.
 - [32] M. Schmidt, J. Gasteringer, S. Nicolas, A. Schülke, HAMLET – A learning curve-enabled multi-armed bandit for algorithm selection, 2020, arXiv:2001.11261.
 - [33] A. Fialho, L. Da Costa, M. Schoenauer, M. Sebag, Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms, in: T. Stützle (Ed.), *Learning and Intelligent Optimization*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 176–190.
 - [34] K. Li, A. Fialho, S. Kwong, Q. Zhang, Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition, *IEEE Trans. Evol. Comput.* 18 (1) (2014) 114–130, <http://dx.doi.org/10.1109/TEVC.2013.2239648>.
 - [35] A. Fialho, L. Da Costa, M. Schoenauer, M. Sebag, Analyzing bandit-based adaptive operator selection mechanisms, *Ann. Math. Artif. Intell.* 60 (1) (2010) 25–64, <http://dx.doi.org/10.1007/s10472-010-9213-y>.
 - [36] A. Carpentier, A. Lazaric, M. Ghavamzadeh, R. Munos, P. Auer, Upper-confidence-bound algorithms for active learning in multi-armed bandits, in: J. Kivinen, C. Szepesvári, E. Ukkonen, T. Zeugmann (Eds.), *Algorithmic Learning Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 189–203.
 - [37] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.* 97 (2019) 849–872, <http://dx.doi.org/10.1016/j.future.2019.02.028>, URL <https://www.sciencedirect.com/science/article/pii/S0167739X18313530>.
 - [38] S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Softw.* 95 (2016) 51–67.
 - [39] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, *Int. J. Comput. Math.* 77 (4) (2001) 481–506, <http://dx.doi.org/10.1080/00207160108805080>, arXiv:https://doi.org/10.1080/00207160108805080.
 - [40] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.* 69 (2014) 46–61.
 - [41] N.H. Awad, M.Z. Ali, P.N. Suganthan, J.J. Liang, B.Y. Qu, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization, 2016.
 - [42] R.A. Khurma, I. Aljarah, A. Sharihe, S. Mirjalili, Evolopy-FS: An open-source nature-inspired optimization framework in python for feature selection, in: S. Mirjalili, H. Faris, I. Aljarah (Eds.), *Evolutionary Machine Learning Techniques: Algorithms and Applications*, Springer Singapore, Singapore, 2020, pp. 131–173, http://dx.doi.org/10.1007/978-981-32-9990-0_8.
 - [43] R. Qaddoura, H. Faris, I. Aljarah, P. Castillo, Evoclust: An open-source nature-inspired optimization clustering framework in python, 2020, pp. 20–36, http://dx.doi.org/10.1007/978-3-030-43722-0_2.
 - [44] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowl.-Based Syst.* 89 (2015) 228–249.
 - [45] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, *Neural Comput. Appl.* 27 (2) (2016) 495–513.

- [46] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, *Adv. Eng. Softw.* 114 (2017) 163–191, <http://dx.doi.org/10.1016/j.advengsoft.2017.07.002>, URL <https://www.sciencedirect.com/science/article/pii/S0965997816307736>.
- [47] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, *Knowl.-Based Syst.* 96 (2016) 120–133.
- [48] Y. Zhang, X. Zhou, P.-C. Shih, Modified Harris Hawks optimization algorithm for global optimization problems, *Arab. J. Sci. Eng.* 45 (12) (2020) 10949–10974, <http://dx.doi.org/10.1007/s13369-020-04896-7>.
- [49] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, *IEEE Trans. Evol. Comput.* 12 (1) (2008) 107–125, <http://dx.doi.org/10.1109/TEVC.2007.895272>.
- [50] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: *Nature & Biologically Inspired Computing*, 2009. NaBIC 2009. World Congress on, IEEE, 2009, pp. 210–214.