IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 25, 2021, accepted July 30, 2021, date of publication August 2, 2021, date of current version August 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3101939

Optimal Stochastic Process Optimizer: A New Metaheuristic Algorithm With Adaptive Exploration-Exploitation Property

JIAHONG XU AND LIHONG XU, (Senior Member, IEEE)
Department of Electronics and Information Engineering, Tongji University, Shanghai 201804, China

Corresponding author: Lihong Xu (xulhk@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61973337.

ABSTRACT Metaheuristic algorithms are constructed to solve optimization problems, but they cannot solve all the problems with best solutions. This work proposes a novel self-adaptive metaheuristic optimization algorithm, named Optimal Stochastic Process Optimizer (OSPO), which can solve different kinds of optimization problems with promising performance. Specifically, OSPO regards the procedure of optimization as a realization of stochastic process, and with the help of Subjective Probability Distribution Function (SPDF) and Receding Sampling Strategy proposed in this paper, OSPO can control the exploration-exploitation property online by the adaptive modification of the parameters in SPDF. This adaptive exploration-exploitation property of OSPO contributes to dealing with different kinds of problems; thus, it makes OSPO have the potential to solve at least a vast majority of optimization problems. The proposed algorithm is first benchmarked on uni-modal, multi-modal and composite test functions both in low and high dimensions. The results are verified by comparative studies with seven well-performed metaheuristic algorithms. Then, 21 real-world optimization problems are used to further investigate the effectiveness of OSPO. The winners of CEC2020 Competition on Real-World Single Objective Constrained Optimization, SASS algorithm, sCMAgES algorithm, EnMODE algorithm and COLSHADE algorithm are used as four comparative algorithms in real-world optimization problems. The analysis of simulations demonstrates that OSPO is able to provide very competitive performance compared to the comparative metaheuristics both in benchmark functions and in real-world optimization problems; thus, the potential of OSPO to solve at least a vast majority of optimization problems is verified. A corresponding MATLAB APP demo is available on https://github.com/JiahongXu123/OSPO-algorithm.git.

INDEX TERMS Optimization, metaheuristic algorithm, benchmark, exploration, exploitation.

I. INTRODUCTION

Nature is the source of metaheuristic algorithms, and researchers tend to mimic different creatures or natural phenomena to get all kinds of metaheuristic algorithms to solve various optimization problems. However, according to ''No Free Lunch Theorem'' [1], every metaheuristic algorithm has its own advantages and limits in dealing with different kinds of problems. Specifically, one metaheuristic algorithm may perform well on some kinds of problems, but its performance may be degraded when solving other kinds of problems. Thus, for different optimization problems, different types of metaheuristics are needed to obtain best solutions, and this is one of the underlying motivations for researchers to create new metaheuristics.

Although researchers have put forward a number of novel metaheuristics, there exists a problem as stated in [2], ''A key question naturally arises: Which is the best one to use? Is there a universal tool that can be used to solve all or at least a vast majority of optimization problems? The simple truth is that there are no such algorithms.''

The main motivation of this contribution is to propose a self-adaptive metaheuristic algorithm which has the potential ''to solve at least a vast majority of optimization problems'' with promising performance.

There are two important characteristics in metaheuristics influencing the performance of the optimization: exploration and exploitation. And in this contribution,

The associate editor coordinating the review of this manuscript and approving it for publication was Bilal Alatas.

they are denoted as exploration-exploitation property for brevity.

For metaheuristics that are imitations of nature, the exploration-exploitation property can be regarded as a habit of a species - one metaheuristic algorithm is one species, and every species has its own habit. Accordingly, every metaheuristic algorithm has its own exploration-exploitation property which mimics a habit. In addition, each species has its own habitats, and these habitats are suitable for its living. Different kinds of problems play the role of different habitats, and the metaheuristic algorithm can perform well on specific problem if the habit and the habitat fits, like a duck to water. Furthermore, a species can change its habit in order to survive in a new habitat. If one metaheuristic algorithm can mimic this adaptation ability, this algorithm may have the ability to solve a vast majority of optimization problems with promising performance.

Today, many researchers focus on coming up with new metaheuristics which have new exploration-exploitation properties. In this way, people can choose an algorithm with suitable habit (the fittest exploration-exploitation property) to deal with a specific optimization problem (a specific habitat) in order to obtain best solutions.

Some recent proposed metaheuristic algorithms are: The Ant Lion Optimizer [3], Artificial Algae Algorithm [4], Moth-flame optimization algorithm [5], Yin-Yang-Pair Optimization [6], Salp Swarm Algorithm [7], PSSA Ant Lion Optimization [8], Shark Smell Optimization [9], Dynamic Butterfly Optimization [10], Lion Optimization Algorithm [11], Dragonfly Algorithm [12], Water Evaporation Algorithm [13], Competitive Optimization Algorithms [14], Galactic Swarm Optimization [15], Electromagnetic Field Optimization [16], Selfish Herds [17], Grasshopper Optimization algorithm [18], Thermal Exchange Optimization [19], Owl Search Algorithm [20], Tree Growth Algorithm [21], Squirrel Search Algorithm [22], Butterfly Optimization Algorithm [23], Henry Gas Solubility Optimization [24], Equilibrium Optimizer [25], Bald Eagle Search [26], Nuclear Reaction Optimization [27], just to name a few.

Unfortunately, choosing a suitable metaheuristic from algorithm library is not a simple task. On the other hand, it seems inefficient and even impossible to create specific algorithms for every kind of problem. Thus, the adaptive exploration-exploitation property which mimics the adaptation ability of intelligent species is needed to settle these two problems. To be specific, when intelligent species are put into a new environment, they can change their habits in order to survive. In this sense, the algorithm with adaptive exploration-exploitation property can be regarded as a nature-inspired metaheuristic algorithm, and it is denoted as adaptive metaheuristic algorithm hereafter.

An important issue of adaptive metaheuristic algorithms is the adaptive modification of exploration-exploitation property, and it can be divided into three sub-problems: 1) (What) the definitions of exploration and exploitation; 2) (When) the time to modify the exploration and exploitation; 3) (How) how to control exploration and exploitation.

Despite the hot research on metaheuristics, widely-accepted formal definitions of exploration and exploitation within the community are still absent [28]. Different researchers described these two characteristics in different ways. Such as in [28], Črepinšek defined exploration and exploitation as follows: ''Exploration is the process of visiting entirely new regions of a search space, whilst exploitation is the process of visiting those regions of a search space within the neighborhood of previously visited points.'' Under this definition, the search agents tend to search the new distant sub- search space in exploration phase. While in the exploitation phase, the search agents tend to search the existed nearby sub- search space. However, it is hard to recognize the border between distant exploration region and nearby exploitation region, which makes this definition somewhat ambiguous.

Similarly, Morales-Castañeda proposed definitions of exploration and exploitation in [29]: ''Exploration refers to the ability of a search algorithm to discover a diverse assortment of solutions, spread within different regions of the search space. On the other hand, exploitation emphasizes the idea of intensifying the search process over promising regions of the solution space, with the aim of finding better solutions or improving the existing ones.'' This definition relies on diversity. However, the relationship between diversity and exploration-exploitation property is unclear. The use of diversity makes this definition imprecise, and there is a need for clear definitions of exploration and exploitation.

In order to determine when to control exploration-exploitation property, a metric of exploration and exploitation is always needed. The simplest metric is iteration - when the optimization iteration is greater than a pre-defined threshold, the algorithm enters the phase of exploration, and all the iterations before are in the phase of exploitation. This method is also called deterministic scheme [30]. Another metric is the diversity of a population - when the performance of the best search agent does not improve for several iterations, increase the ability of exploration [31]; or when the value of diversity is lower than a pre-defined threshold, increase the ability of exploration [32]. This kind of method is called adaptive scheme. However, the deterministic scheme is too simple, and the adaptive scheme relies on a threshold which is hard to determine. Thus, there is a need for a more intelligent method to determine when to control exploration and exploitation.

As for controlling the exploration and exploitation, there exists three typical methods: diversity maintenance, diversity control and diversity learning [28]. However, all these methods are indirect control methods, and the diversity is not identical to the balance between exploration and exploitation. Thus, there is a need for a more direct method to control exploration and exploitation.

To summarize, there are three main difficulties in constructing adaptive metaheuristics: 1) the ambiguous definitions of exploration and exploitation; 2) it is hard to determine

when to control exploration and exploitation; 3) it is hard to control exploration and exploitation directly.

Some intelligent metaheuristics have been proposed recently with adaptive exploration-exploitation properties. In [33], authors introduced Spherical Search (SS) algorithm with self-adaptive control parameters c_i, *rank* and the size of population N. However, this algorithm: 1) takes the number of function evaluations as the criterion to distinguish exploration and exploitation, which is too simple; 2) uses search history to update new parameters in the next iteration, which may lead to local optimum. In [34], authors proposed self-adaptation versions of Grey Wolf Optimization (AMGWO) and Moth Flame Optimization (AMFO). The self-adaptive parameters of these two algorithms are controlled by the previous obtained solutions. However, these algorithms do not control the exploration-exploitation property directly; thus, the modification of parameters is hard to understand in the sense of exploration and exploitation. Other self-adaptive algorithms such as [35]–[40] have also been put forward in order to dynamically update the control parameters online. However, the aforementioned three problems (What-When-How) are not solved in these algorithms. In other words, they lack a direct control of exploration-exploitation property, and the adaptation mechanisms of the algorithms have weak relationships with exploration-exploitation property.

The adaptive metaheuristics reviewed above tend to judge the exploration-exploitation property in a reverse way - if an algorithm has obtained satisfactory solutions, then this algorithm is considered to have a good balance between exploration and exploitation. However, this reverse thinking does not take advantage of the exploration-exploitation property to improve the performance of the algorithm during the optimization. Thus, positive thinking is needed to utilize exploration-exploitation property online directly, and that's just what our adaptive metaheuristic algorithm OSPO tries to realize.

In order to have a better understanding of the term ''adaptive'' in adaptive metaheuristics, we discuss it briefly. This ''adaptive'' could mean two possibilities in the existing metaheuristics: one is the adaptive selection of the parameters and the other is the adaptive selection of the update operators. For the first possibility, such metaheuristics use only one update operator and rely on the control parameters to explore and exploit the search space. For example, PSO [53] uses three control parameters within one update operator, and these control parameters (w, c_1 and c_2) try to keep a balance between exploration and exploitation. For the second possibility, such metaheuristics have no particular way to control exploration and exploitation directly. More precisely, take TLBO [54] as an example, it does not have any control parameters, but it provides two update operators: teacher phase (for global exploration search) and student phase (for local exploitation phase). TLBO simply calls these two operators in a deterministic sequence (explore - exploit - explore - exploit ...) to perform the search process. However, these two possibilities have their own limitations: 1) for the adaptive selection of the control parameters, it is hard to determine the specific value of these parameters, and the relationship between these values and the exploration-exploitation property is unclear; 2) for the adaptive selection of the update operators, the exploration-exploitation property is clear since one operator can represent the exploration phase and the other operator can represent the exploitation phase. However, it is hard to intelligently determine when to change update operators, and a poor sequence of update operators may even degrade the performance of the algorithm.

Hyper-heuristics (HHs) comprise a set of approaches that are motivated (at least in part) by the goal of automating the design of heuristic methods to solve hard computational search problems [55], and they perform another kind of adaptive in a higher level. HHs have emerged as a way to raise the level of generality of search techniques, and the current state-of-the-art in HHs comprise a set of methods that broadly concerned with intelligently selecting or generating a suitable heuristic for a given situation [56]. A basic difference between metaheuristics and hyper-heuristics is that the former explore the solution space of a problem, whereas the latter focus on the solver space. So, a hyper-heuristic does not solve a problem directly. Instead, at each step of the solution process, it selects a heuristic for dealing with such a step [57]. As stated in [58], HHs can be regarded as a dual-stage methodology, the first stage is to compile a collection of Search Operators (SOs) from metaheuristics, and the second stage is to implement a Random Search approach to tune the hyper-heuristic. Once a metaheuristic is constructed by SOs, this metaheuristic is then used to solve the optimization problem in the solution space. The applications of HHs can be referred to [59]. Since HHs can design metaheuristics automatically, it avoids the problem of choosing suitable metaheuristic for the optimization problem. In other words, HHs may be the one direction to solve the motivation ''a self-adaptive metaheuristic algorithm which has the potential to solve at least a vast majority of optimization problems with promising performance''. However, 1) there is no theoretical promise that the HHs can outperform metaheuristics in all kinds of problems; 2) the online-construction of metaheuristic in HHs is another kind of choosing of metaheuristics; 3) in addition, the higher-level in HHs is something like a reset of the exploration-exploitation property in metaheuristics, and the algorithm proposed in this contribution (OSPO) also has this reset-ability ensured by the Receding Sampling Strategy. So, we do not focus on HHs in the current contribution, and our future research may focus on this direction.

In this paper, we propose a novel adaptive metaheuristic algorithm called Optimal Stochastic Process Optimizer (OSPO), which uses positive thinking to control exploration-exploitation property directly during the optimization. Specifically, OSPO regards the optimization procedure as a stochastic process, and the sample center and the sample extent at every optimization iteration determine the exploration-exploitation property directly. This proactive

control of exploration-exploitation property can improve both the global exploration ability and the local exploitation ability. In addition, the adaptive modification of parameters in Subjective Probability Distribution Function (SPDF) makes the sample path of the stochastic process become a searching trajectory, and then the Receding Sampling Strategy resets the exploration-exploitation property in order to find the global optimum of the problem. The simulation results of benchmark functions and real-world optimization problems demonstrate that, thanks to the adaptive modification of exploration-exploitation property, OSPO can solve different problems with competitive performance compared to other metaheuristic algorithms. Thus, OSPO is demonstrated to have the potential ''to solve at least a vast majority of optimization problems'' with promising performance.

The rest of the paper is organized as follows:

Section 2 presents the basic idea of Optimal Stochastic Process Optimizer. Then, the realization of OSPO is detailed in Section 3. The simulation results and discussion of benchmark functions as well as real-world optimization problems are presented in Sections 4. Finally, Section 5 concludes the contribution and suggests some directions for future studies.

II. BASIC IDEA OF OSPO

Most metaheuristic algorithms have the following properties: 1) they are nature-inspired; 2) they use some kind of random operators; 3) they do not calculate the gradient of hessian matrix of objective functions; 4) they have some parameters to be tuned.

On the one hand, these properties are the foundations of the superiorities of metaheuristics. On the other hand, these properties also limit the application of metaheuristic algorithms.

Nature inspired: most metaheuristic algorithms mimic some kind of natural phenomena or species; thus, they have some specific exploration-exploitation properties. However, if metaheuristics cannot change their exploration-exploitation properties online during the optimization procedure, they can only perform well on limited optimization problems.

Random operators: random operators let search agents have the abilities to explore the entire search space and to escape from the local optimal traps. However, these abilities are not reliable because random operators may even degrade the performance of the algorithm.

No gradient or hessian matrix computation: the avoidance of computation of gradient and hessian matrix can decrease the computation burden, but metaheuristics also lose a connection between adjacent iterations. As a result, the convergence speed of metaheuristics may become slow.

Parameter tuning: the performance of metaheuristics can be improved by parameter tuning. However, this parameter tuning is often off-line, and if poor parameters have been chosen, the performance of the algorithm cannot be promised.

The above four limitations can be explained in the sense of the exploration-exploitation property: 1) the exploration-exploitation property cannot be modified adaptively; 2) the exploration-exploitation property may not lead to the global optimum; 3) the exploration-exploitation property is not efficient enough for optimization; 4) the exploration-exploitation property cannot be controlled online.

The adaptive metaheuristic algorithm OSPO introduced in this contribution has the ability to solve these limitations by controlling the exploration-exploitation property online directly. The basic idea of OSPO is presented in this section.

A. STOCHASTIC PROCESS PERSPECTIVE ON OPTIMIZATION

In order to solve optimization problems, metaheuristic algorithms come up with different kinds of bio-inspired search agents $x \in \mathcal{R}^n$, such as artificial ants, to search for the global optimum in a search space $\Omega \subseteq \mathcal{R}^n$. Every single point x in the search space is related to a scalar value calculated by an objective function $f(x) \in \mathcal{R}$, and this function is what the algorithm tries to optimize (minimize in this contribution). The differences between different metaheuristics are mainly their distinctive definitions of the search agents as well as the relevant intelligent seeking strategies. These intelligent seeking strategies reflect the latent exploration-exploitation properties of the metaheuristics.

In probability theory, there exists a sample space $U \subseteq \mathcal{R}^n$ which is the set of all possible sample points $u \in \mathcal{R}^n$. A random variable $X(u)$ is a measurable function defined on a probability space that maps from a sample space U to a real number space $\mathcal{R}$.

Several significant similarities can be recognized between these two mechanisms: 1) the search space is similar to the sample space; 2) the search agents are the same as the sample points; 3) in addition, they both have a scalar mapping.

On the other hand, metaheuristics utilize an objective function as a ''feedback'' to guide the trajectory of search agents; thus, the search agents have the ability to eventually converge to the global optimum. Different feedback schemes result from different intelligent seeking strategies, and these strategies give algorithms different exploration-exploitation properties.

While in probability theory, such ''feedback-like ability'' is absent. Luckily, random variables indexed by time coordinates T (or other coordinates) form a stochastic process, and the realizations of such stochastic process are something like the ''searching trajectories'' in metaheuristics. However, these realizations of stochastic process are just ''records'' of the variation of random variables without any ''searching ability''. If the stochastic process can be equipped with some kind of searching ability, then the realization of stochastic process can be regarded as an intelligent seeking strategy, and the random property of the stochastic process becomes the adaptive exploration-exploitation property.

In order to build a bridge between optimization and stochastic process, a feedback mechanism should be introduced into stochastic process first. In this way, the realization of stochastic process is no longer a random record, but a searching trajectory related to optimization problems.

This optimization-oriented stochastic process will generate an optimization-oriented sample path, and the last sample point of the sample path is just the global optimal solution. In other words, the random variable approaches to the optimum ($X(u) \to f_{min}$) at the end of the sample path.

Definition 1: Let the search space of the optimization problem $\Omega \subseteq \mathcal{R}^n$ be the sample space, let the search agents $x \in \mathcal{R}^n$ be the sample points, and let the objective function $f(x)$ be the random variable $Y(x)$. Denote $\{Y(x,s), s \in \mathcal{S}\}$ as *the optimizing stochastic process* indexed by set $\mathcal{S}$, where $\mathcal{S} = \{1, 2, \ldots, l\}$ denotes the sampling iteration.

In Definition 1, the elements in metaheuristics are introduced into probability theory. Similar to time index which is mostly used in stochastic process, sampling iteration index $\mathcal{S}$ used here represents the iteration of optimization. Specifically, $\mathcal{S} = \{1, 2, \ldots, l\}$ where l is the length of the corresponding optimizing stochastic process.

With this optimizing stochastic process $\{Y(x,s), s \in \mathcal{S}\}$, the realization of such stochastic process (sample path) can now be regarded as a searching trajectory. At sampling iteration i, the current sample point is $x(s_i)$, and the value of random variable is $Y(x, s_i) = f(x(s_i))$ or just $Y(s_i)$ for brevity. Sample l times from sample space, then a sequence of random variables is obtained as follows

$$\{Y(x,s_1), Y(x,s_2), \ldots, Y(x,s_l)\}, \tag{1}$$

and this sequence is exactly one realization of the optimizing stochastic process $\{Y(x,s), s \in \mathcal{S}\}$.

There are countless realizations of such an optimizing stochastic process, and each specific sequence of random variables is called one sample path of the corresponding optimizing stochastic process.

Definition 2: One of the realizations of the optimizing stochastic process $\{Y^*(x,s), s \in \mathcal{S}^*\}$ is called *the optimal sample path*, and the corresponding optimizing stochastic process which can generate optimal sample path is denoted as *optimal stochastic process*. The optimal sample path must satisfy

$$\{Y^*(x,s), s \in \mathcal{S}^*\} = \left\{Y(x,s_1), Y(x,s_2), \ldots, Y(x,s_l); s_i \in \mathcal{S}^*\right\},$$

where

$$Y(x,s_l) \in \mathcal{B}_\delta^f(f^*).$$

In Definition 2, l is the length of sampling iteration set $\mathcal{S}$, f^* is the optimal value of the objective function, $\mathcal{B}_\delta^f(f^*)$ is a δ-neighborhood of f^* which satisfies

$$\mathcal{B}_\delta^f(f^*) = \left\{f \,|\, f^* - \delta \le f(x) \le f^* + \delta, \delta \ge 0\right\}. \tag{2}$$

The δ-neighborhood $\mathcal{B}_\delta^f(f^*)$ builds a bridge between the random variable $Y(x,s)$ and the algebraic variable $f(x)$. As for optimization problem, the algorithm wants to find the optimal solution x^{opt} which minimizes the objective function to the optimal value $f^* = f(x^{opt})$. However, the probability of a continuous random variable to any specific value is 0, in other words, $P\{Y(x) = f(x^{opt})\} = 0$. To overcome this, the optimal sample path tries to find a small enough region δ of the optimal solution instead of a specific point, then the probability of this region becomes

$$P\left\{f(x^{opt}) - \delta \le Y(x) \le f(x^{opt}) + \delta\right\} = \int_{\mathcal{B}_\delta^f(f(x^{opt}))} y(x)dx$$

which has a value bigger than 0.

The sample path is determined by the random variable at each sampling iteration, and each random variable has a corresponding probability distribution that represents the likelihood of the occurrence of any possible point. Thus, the problem of getting the optimal sample path becomes a problem of getting optimal probability distributions at each sample iteration. Once the probability of random variable $Y(x,s)$ that falls into the optimal region $\mathcal{B}_\delta^f(f(x^{opt}))$ increases as the sampling iteration s_i increases, then the realization of the optimizing stochastic process becomes the optimal sample path, and the optimization problem is solved.

Definition 3: Let x^{opt} be the optimal solution of the given optimization problem, and $f(x)$ be the corresponding objective function. Then $y^o(x)$ is said to be *the optimal probability distribution* if it satisfies

$$\int_D y^o(x)dx = 1, D = \mathcal{B}_\delta^f(f(x^{opt})). \tag{3}$$

Fig. 1 shows some possible optimal sample paths which all converge to the δ-neighborhood of the optimal solution. The dashed lines corresponding to s_i is the range of the random variable at s_ith sampling iteration. As shown in the figure, the range of random variables is decreasing over iterations, and finally this range decreases into $\mathcal{B}_\delta^f(f(x^{opt}))$. This decreasing tendency is not a necessary property in the definition of the optimal sample path, and it is added here for the purpose of alleviating the difficulty of obtaining the optimal probability distribution.

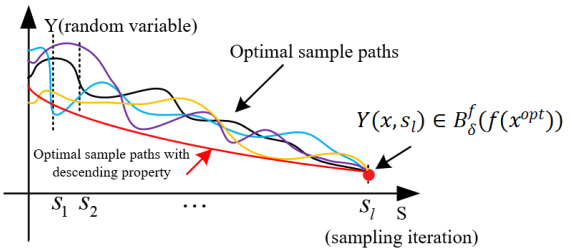

FIGURE 1. Optimal sample paths of stochastic process.

At every sampling iteration s_i, OSPO will obtain a specific random variable based on the probability distribution of the random-variable-generator (RVG). Since the length of the stochastic process is l, the RVG uses l sequential probability distributions to form a sample path with l random variables. In order to make this sample path become the optimal sample path which can solve the optimization problem, the joint distribution should satisfy some specific conditions.

Definition 4: Let $\{Y(x,s), s \in \mathcal{S}\}$ be an optimizing stochastic process, $f_i, i = 1, 2, \ldots, l$ be the reference values of the objective function. Then, for any $l \geq 1$ and $s_1, s_2, \ldots, s_l \in \mathcal{S}$, *the optimizing joint distribution function* of random variable vector $(Y(s_1), Y(s_2), \ldots, Y(s_l))$ is defined as

$$F(f_1, f_2, \ldots, f_l; s_1, s_2, \ldots, s_l) = P\{Y(s_1) \leq f_1, Y(s_2) \leq f_2, \ldots, Y(s_l) \leq f_l\},$$

where

$$f_1 \geq f_2 \geq \ldots \geq f_l.$$

Additionally, if the last reference value f_l satisfies

$$f_l \in \mathcal{B}_\delta^f\left(f(x^{opt})\right),$$

this optimizing joint distribution function is said to be *the optimal joint distribution function* and is denoted as F^o.

Proposition 1: Given a sampling iteration index $\mathcal{S}$ and an optimal joint distribution function F^o which both satisfy the consistency conditions proposed in Kolmogorov extension theorem, then there must exist a corresponding optimal stochastic process $\{Y^*(x,s), s \in \mathcal{S}^*\}$ whose joint distribution function is F^o according to Kolmogorov extension theorem.

In Definition 4, since $f_l \in \mathcal{B}_\delta^f(f(x^{opt}))$, $F^o(\cdot, f_l; \cdot, s_l)$ becomes the probability of generating the optimal sample path of an optimal stochastic process, and the existence of this optimal stochastic process is promised by Proposition 1. Thus, every optimization problem can be turned into the problem of finding an optimal sample path of the corresponding optimal stochastic process.

Despite the existence of the optimal stochastic process in the search space of the optimization problem, the probability distribution for generating the optimal sample path is still unknown. It is impossible to find this optimal sample path by totally random sampling, and there is a need to control the probability distribution intelligently. This intelligent control of the probability distribution is similar to the intelligent searching strategy of the search agents in metaheuristics.

Assumption 1: Assume the probability distribution among different sampling iterations are independent, then

$$\begin{aligned} &F(f_1, f_2, \ldots, f_l; s_1, s_2, \ldots, s_l) \\ &= P\{Y(s_1) \leq f_1, Y(s_2) \leq f_2, \ldots, Y(s_l) \leq f_l\} \\ &= \Pi_{i=1}^{l} P\{Y(s_i) \leq f_i\}. \end{aligned}$$

Definition 5: At sampling iteration s_i, regard the current probability distribution function (PDF) as the optimal probability distribution $y_i^{sp}(x)$, and then this type of PDF is denoted as *Subjective Probability Density Function* (SPDF).

The SPDF can be any kind of distribution, and in this contribution the *multivariate Gaussian distribution* is chosen for its simplicity. At every sampling iteration s_i, modify the parameters of SPDF adaptively and then use the updated SPDF to generate new sample points. But this modified SPDF is not the realistic distribution that will definitely form an optimal sample path; in other words, $Y(s_i) \leq f_i$ cannot be guaranteed. It is only the "subjective belief" of the algorithm that the optimal sample paths will occur by using SPDF.

With the help of SPDF, the optimization problem is simplified into the intelligent control of the parameters in the SPDF. These parameters are modified online based on the feedback information during the optimization procedure; thus, feedback mechanisms are introduced into OSPO. Based on this feedback mechanism, and inspired by the Model Predictive Control concept in control theory, the optimal sample path can be eventually obtained through a strategy called receding sampling.

Definition 6: At the outer sampling iteration $\mathcal{S}^{outer}$, set the initial label of the inner sampling iteration to s_1, and a series of SPDF is used to create a sample path. If the resulting sample path $\{Y^{outer}(x,s), s \in \mathcal{S}^{outer}\}$ is not one of the optimal sample paths, then take the k-th sample point $x(s_k)$ in the current sample path as the initial sample point $x(s_1)$ of another sample path at the next outer sampling iteration $\mathcal{S}^{outer+1}$. Repeat this operation until the current sample path becomes an optimal sample path $\{Y^*(x,s), s \in \mathcal{S}^*\}$, and this procedure is denoted as *Receding Sampling Strategy*. The outer sampling iteration is also denoted as receding sampling iteration.

The parameter k can be changed over iterations satisfying $1 \leq k \leq l$; or, $x(s_1)$ can be chosen randomly. The specific value of k depends on the performance of the current sample path. The simplest idea of choosing k is to choose the best sample point in every sample path, and this simplest receding sampling strategy is depicted in Fig. 2. The receding sampling idea can be understood as cascaded sample paths.

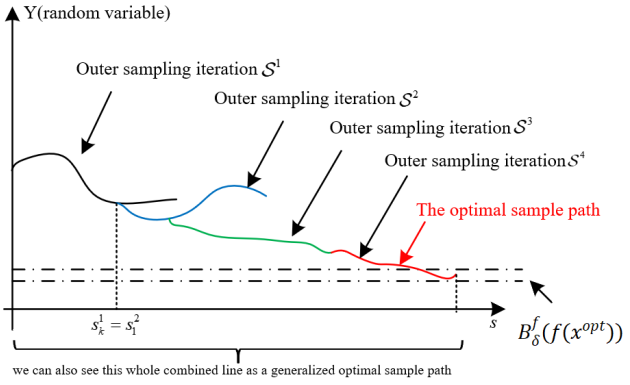

FIGURE 2. Illustration of receding sampling strategy.

Remark 1: In standard stochastic process, the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is fixed, while in our optimal stochastic process, the probability space is variable, which means $(\mathcal{F}, \mathbb{P})$ can be changed by SPDF. This changing probability space gives the optimal stochastic process the ability of optimization.

Remark 2: The essence of using receding sampling strategy is to reset the exploration-exploitation property of the algorithm in a higher level. Specifically, within single outer receding sampling iteration, the adaptive modification of SPDF has already changed the exploration-exploitation

property in order to form an optimal sample path. However, this adjustment is in a lower level and may lead to local optimum. If the resultant sample path is not the optimal sample path, this minor modification of exploration-exploitation property is not enough for optimization, and some aggressive modifications are needed to find the global optimum.

Based on the aforementioned discussions, the following proposition can be concluded about Optimal Stochastic Process Optimizer for solving optimization problems.

Proposition 2: For any optimization problem, given an initial point at the sample space, receding sampling strategy together with SPFD can be used to find an optimal sample path of a corresponding optimal stochastic process. The last random variable in this optimal sample path $Y^*(x^*, s_l)$ is the optimal value of the given optimization problem, and the corresponding sample point x^* is the optimal solution. This whole procedure for finding the optimal solution of the optimization problem is denoted as *Optimal Stochastic Process Optimizer* (OSPO).

III. THE REALIZATION OF OSPO

The brief flowchart of OSPO is illustrated in Fig. 3.

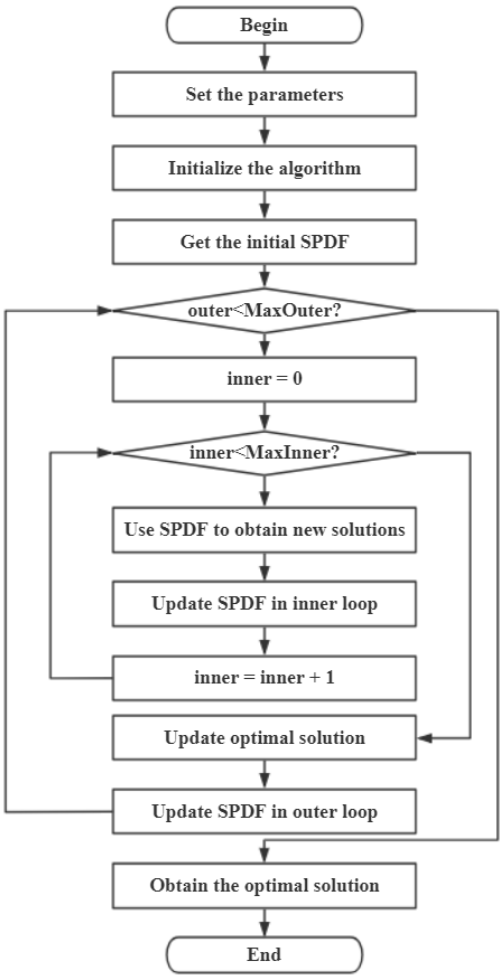

FIGURE 3. Brief flowchart of OSPO.

A. THE REALIZATION OF OSPO

1) MULTIVARIATE GAUSSIAN SAMPLING

Let the search space of the optimization problem be the sample space $\Omega \subseteq \mathcal{R}^n$, the N search agents be the sample points, and then the searching trajectory of the search agents in traditional metaheuristics is identical to the sample procedure at each sample iteration *iter* by using SPDF as the probability distribution. After obtaining N sample points at the current sample iteration, OSPO calculates the fitness value of each sample point by the objective function. The optimal sample point x_*^{iter} with the best objective function value is taken as the current random variable $Y(x, s_i) = f(x_*^{iter})$, and it is used to construct a sample path.

The most important issue in OSPO is the adaptive modification of the SPDF at each sampling iteration. In this contribution, OSPO uses Multivariate Gaussian Sampling (MGS) as its SPDF, and the parameters in MGS should be modified adaptively over iterations in order to construct an optimal sample path.

Specifically, for every inner sampling iteration s_i in the first outer receding sampling iteration, the SPDF has the following n- dimensional normal joint density function form

$$y_i^{sp}(x) = \frac{1}{\left(\sqrt{2\pi}\right)^n \sqrt{det(\Sigma^i)}} exp^{-\frac{(x-\mu^i)^T (\Sigma^i)^{-1} (x-\mu^i)}{2}}, \quad (4)$$

where mean matrix μ^i indicates the search center, covariance matrix Σ^i indicates the search extent, $x = [x_1, x_2, \ldots, x_n]$ indicates the possible sample points within the search extent, $y_i^{sp}(x)$ indicates the probability for every sample point to be sampled. The parameters (μ^i, Σ^i) at each sampling iteration is updated by the previous optimization information; thus, these parameters can be controlled online through feedback. The initial parameters (μ^0, Σ^0) are chosen randomly.

2) THE MEANING OF SAMPLE PROBABILITY IN OSPO

The underlying reasons for using SPDF to generate sample points as search agents at every optimization iteration are mainly twofold: a) the sample probability is assumed to be the optimality of each sampled point. In other words, the higher probability the sample point may be chosen, the higher optimality the sample point is in optimization problem; b) the probability and optimality are not truly identical, and it is only a subjective assumption. Thus, the probability needs to be corrected online in order to fit with the real optimality.

As shown in Fig. 4, the mean matrix μ^i (denoted by point x_1) in SPDF represents the estimated optimal point, which means the highest probability implies the highest optimality. Unfortunately, this equivalence-relation is not the real fact, and the real highest optimality point is x_2 whose probability in SPDF is not the highest. The covariance matrix Σ^i represents the verification range: sample several points within the search extent, and then use these verified-optimality (the value of the objective function) to modify SPDF. A strategy called Preferential Sampling is introduced to update SPDF.

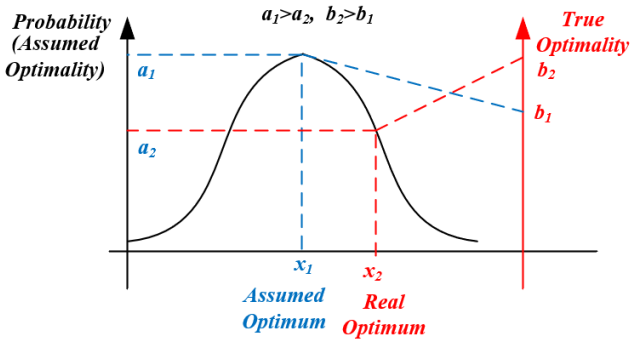

FIGURE 4. The relationship between probability and optimality.

3) PREFERENTIAL SAMPLING STRATEGY

In *iter*-th sampling iteration, sample m times based on SPDF, and OSPO obtains m sample points $x^{iter,j}, j = 1, 2, \ldots, m$. Sort these sample points according to their optimality (the objective function values $f(x^{iter,j})$), and then choose the best q sample points. These q ordered sample points form a matrix called "*Acceptable Sample Matrix*". This matrix is denoted as X_a hereafter, and it has the form as follows

$$X_a = \left[x^{r(1)}, x^{r(2)}, \ldots, x^{r(q)}\right]^T = [X_1, X_2, \ldots, X_n], \quad (5)$$

where the superscript $r(\cdot)$ represents the ordered label, and these labels satisfies

$$f\left(x^{r(1)}\right) \leq f\left(x^{r(2)}\right) \leq \ldots \leq f\left(x^{r(q)}\right) \leq f\left(x^{r(q+1)}\right) \leq \ldots \leq f(x^{r(m)}), \quad 1 < q < m.$$

The motivation of constructing Acceptable Sample Matrix X_a is as follows: the current SPDF is not accurate, and the resulting optimality of some sampled points may be poor, so the bad sampled points should be discarded. With the help of X_a, OSPO can utilize the better sampled points to update the SPDF. Thus, the consistency of the SPDF and the optimality can be further improved in the next sample iteration.

Here, the total m sample points are the sample, and this sample is obtained from the population based on the entire search space. However, what OSPO really needs is not the sample from the entire search space, but the sample from the promising sub-space which has the potential to get optimal solutions. In other words, OSPO wants to sample from the "optimizing subspace" which has the higher potential to generate optimal sample paths. The Acceptable Sample Matrix is a smaller sample whose population is no longer the initial space Ω, but some promising optimizing subspace Ω_0. The idea of using X_a to construct Ω_0 and then using this Ω_0 to update the SPDF in the next sample iteration is called *Preferential Sampling Strategy*.

The brief idea of preferential sampling strategy is illustrated in Fig. 5. As shown in the figure, the green points represent the Acceptable Sample Matrix X_a obtained by the current SPDF, the red points represent the real optimality of this X_a calculated by the objective function. Use this optimality information as feedback information to update the next SPDF, and OSPO obtains the green curve as a new SPDF to generate sample points in the next sample iteration.

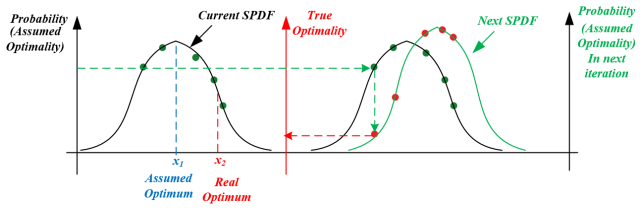

FIGURE 5. The brief idea of preferential sampling.

4) THE ADAPTIVE MODIFICATION OF μ, Σ

The adaptive modification of μ, Σ is achieved by using the feedback optimality information to update the probability distribution. Specifically, in the current sample iteration, OSPO utilizes X_a to update the mean vector μ^{i+1} and the covariance matrix Σ^{i+1} in the next sampling iteration. By implementing the preferential sampling strategy, the parameters of SPDF have been updated as follows

$$\mu^{i+1} = \frac{\sum_{j=1}^{q} \alpha_j x^{r(j)}}{\sum_{j=1}^{q} \alpha_j}, \quad (6)$$

$$\Sigma^{i+1} = cov\,(X_a) = cov\,([X_1, X_2, \ldots X_n]) = \begin{bmatrix} cov(X_1, X_1) & cov(X_2, X_1) & \cdots & cov(X_n, X_1) \\ cov(X_1, X_2) & cov(X_2, X_2) & \cdots & cov(X_n, X_2) \\ \vdots & \vdots & \ddots & \vdots \\ cov(X_1, X_n) & \cdots & \cdots & cov(X_n, X_n) \end{bmatrix}, \quad (7)$$

where $x^{r(j)}$ represents the j-th acceptable sample point, α_j represents the weight of the corresponding sample point, and this weight has the form $\alpha_j = 1/f(x^{r(j)})$ which is related to the objective function value; X_i represents the column vector of X_a. $cov\,(A, B)$ represents the computation of the covariance whose formula is

$$cov\,(A, B) = \frac{1}{N-1} \sum_{i=1}^{N} (A_i - \mu_A) * (B_i - \mu_B), \quad (8)$$

where μ_A, μ_B are the mean value of the vector A and B, A_i and B_i are the elements in the corresponding vector, and N represents the dimension of the vector.

5) THE COMPUTATION OF RANDOM VARIABLES IN SAMPLE PATH

In the current sampling iteration, OSPO gets m sample points, but as for the sample path, it only needs one value to be the current random variable. Remember, $x^{r(1)}$ denotes the best sample point in the current sample iteration, and OSPO takes this best sample point as the random variable of the current sample path. Thus, the random variable of the sample path in the i-th inner sample iteration is

$$Y^1\,(x, s_i) = f(x^{r(1)}\,(s_i)). \quad (9)$$

After l times sampling iterations, OSPO has implemented l times preferential sampling, and it obtains a sample path with the length of l denoted as $\{Y^1(x,s), s \in \mathcal{S}^1\}$. If the last random variable satisfies $Y^1(x,s_l) \in \mathcal{B}_\delta^f\left(f(x^{opt})\right)$, the current sample path becomes the optimal sample path and the optimization problem is solved; otherwise, OSPO needs to use receding sampling strategy to get new sample paths $\left\{Y^2(x,s), s \in \mathcal{S}^2\right\}, \left\{Y^3(x,s), s \in \mathcal{S}^3\right\}, \ldots$ until the optimal sample path is achieved or the terminal criterion is satisfied.

6) THE INTERACTIONS BETWEEN DIFFERENT SAMPLE PATHS

In the initial outer sample iteration $\mathcal{S}^1$, the initial SPDF can be chosen randomly as follows

$$\begin{aligned} \mu^0 &= lb + rand*(ub - lb), \\ \Sigma^0 &= diag\left(\left(\frac{ub - lb}{10}\right)^2\right), \end{aligned} \tag{10}$$

where *rand* is the random value in the range of $[0,1]^{Dim}$, *Dim* is the dimension of the problem; *ub* is the upper bound, *lb* is the lower bound, $diag(\cdot)$ is the diagonal matrix.

In the following outer sample iterations S^{outer}, $outer > 1$, the values of the initial SPDF for each new sample path cannot be chosen randomly, because random selection will discard the useful information of the previous sample paths. The utilization of the former information is the strength of the population-based strategy. In order to build connections between different sample paths and take advantage of the previous information, OSPO selects one sample point from the previous sample path as the initial point of the next new sample path. A simple and straightforward way is to choose the best sample point in the last sample path x_*^{outer} as the new initial point

$$\mu^0\left(\mathcal{S}^{outer+1}\right) = x_*^{outer}. \tag{11}$$

Here, $\mu^0\left(\mathcal{S}^{outer+1}\right)$ represents the initial search center of the new sample path, and the search extent is implied by the covariance matrix Σ^0 which has the following formula

$$\Sigma^0\left(\mathcal{S}^{outer+1}\right) = diag\left(\left(\frac{ub - lb}{10 + 10*outer}\right)^2\right), \tag{12}$$

where *outer* is the label of the current sample path. As the number of sample paths *outer* increases, the initial value of covariance matrix Σ^0 is decreased accordingly. As a result, OSPO will focus more on exploitation in the higher level as the optimization goes on.

When there are a number of local optima exist in the optimization problem, the sample path generated by SPDF may lead to local optimum. In order to search the entire space in these situations, the initial points of every sample path could be determined at the beginning of the optimization. In this way, a pre- sample path $\mathcal{S}^0$ is introduced to scatter the initial points of the sample paths $\mathcal{S}^{outer}$, $outer = 1, 2, \ldots$ in the whole search space uniformly.

7) THE STOP CRITERION

a: OUTER STOP CRITERION

Different metaheuristics have different populations and different searching strategies; thus, they have different computation burden for each iteration. Since the function evaluation may contribute the major computation burden in optimization, the maximum number of function evaluations *Max_FE* plays the role of stop criterion in this contribution to make fair comparisons between different metaheuristics.

Specifically, when the number of function evaluations exceeds *Max_FE*, the OSPO algorithm will be terminated, and the optimal solution obtained so far is regarded as the global optimal solution for the optimization problem.

b: INNER STOP CRITERION

For each sample path, the solutions may fall into local optimum. In these situations, OSPO should have the ability to terminate the current sample path early.

OSPO uses a criterion called three-repetition as the early termination condition in the inner loop. Specifically, for every sample path $\left\{Y^{outer}(x,s), s \in \mathcal{S}^{outer}\right\}$, if the variance of the best three optimal solutions $x_*^{bt1}, x_*^{bt2}, x_*^{bt3}$ obtained in the current sample path satisfy a predefined condition, then OSPO terminates the current sample path even the length is smaller than the l.

Here, the early termination condition is defined as follows

$$\frac{\left(Var\left(x_*^{bt1}, x_*^{bt2}\right) + Var\left(x_*^{bt2}, x_*^{bt3}\right) + Var\left(x_*^{bt3}, x_*^{bt1}\right)\right)}{3} < 1e^{-6}, \tag{13}$$

where $Var(a,b)$ is the variance of the vectors a and b.

The pseudo code of OSPO is shown in Fig. 6.

B. THE ADAPTIVE MODIFICATION OF EXPLORATION-EXPLOITATION PROPERTY

In the traditional metaheuristic algorithms, the exploration and exploitation need to be defined and measured in order to improve the performance of the algorithms. While in the OSPO, the exploration and exploitation are directly related to the sample center and sample extent in the SPDF. In other words, the parameters in the SPDF represent the exploration-exploitation property of the OSPO straightforwardly, and the dynamic control of these parameters is equivalent to the adaptive modification of the exploration-exploitation property.

OSPO uses multivariate Gaussian distribution as its SPDF in this contribution. The mean vector is the sample center representing the search center, and the covariance matrix is the sample extent representing the search extent. Within one single sample path, the search extent Σ^i is decreasing with the help of Acceptable Sample Matrix X_a, and the smaller covariance matrix implies the higher degree of exploitation. On the other hand, the variance of the mean vector μ^i implies the variance of the exploitation center, and a larger deviation of μ^i implies a higher degree of exploration.

Algorithm 1 Optimal Stochastic Process Algorithm

Define the parameters: population N, number of acceptable points q, length of sample path l, maximum number of receding sampling $MaxOuter$, maximum number of function evaluations Max_FE.
Initialize the population of sample points with the initial SPDF by Eq. (4) and Eq. (10)
Calculate the fitness of sample points and rank them
Select the acceptable sample points with highest rank and get the acceptable sample matrix X_a in Eq. (5)
Calculate μ^1 and Σ^1 as the parameters of SPDF for next sampling iteration using Eq. (6) and Eq. (7)
Activate the receding sampling and define $outer = 1$
while $outer < MaxOuter$
  for $i = 0: l-1$
    Use SPDF to generate new population of sample points
    Calculate the fitness of sample points and rank them
    Select the acceptable sample points with highest rank and get the acceptable sample matrix X_a in Eq. (5)
    Calculate μ^{i+1} and Σ^{i+1} as the parameters of SPDF for next sampling iteration using Eq. (6) and Eq. (7)
    Store the elite sample point as the random variable in the sample path in Eq. (9)
    $i = i + 1$
  end for
  if outer ==1
    The optimal value is $f^* = Y^{outer}(x, s_l)$, the optimal solution is $x^{opt} = x^{r(1)}(s_l)$
  else
   $f^* = \min(f^*, Y^{outer}(x, s_l))$,$x^{opt} = best(x^{opt}, x^{r(1)}(s_l))$
  end if
  if the number of function evaluations is bigger than Max_FE
    break **while**
  else
    Choose $x^{r(1)}(s_k)$ as the initial point of the next receding sampling iteration by Eq. (11)
    Calculate μ^0 and Σ^0 by Eq. (11) and Eq. (12)
   end if
   $outer = outer + 1$
end while
Return $\boldsymbol{x^{opt}}$ **and** $\boldsymbol{f^*}$

FIGURE 6. The pseudo code of OSPO.

TABLE 1. Uni-modal benchmark functions.

Function	Dim	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} (x_i - d_i)^2$	30/100	[−100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i - d_i \rvert + \prod_{i=1}^{n} \lvert x_i - d_i \rvert$	30/100	[−10,10]	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} (x_j - d_j) \right)^2$	30/100	[−100,100]	0
$F_4(x) = \max \{ \lvert x_i - d_i \rvert, 1 \le i \le n \}$	30/100	[−100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} [100((x_{i+1} - d_{i+1}) - (x_i - d_i)^2)^2 + ((x_{i-1} - d_{i-1}) - 1)^2]$	30/100	[−30,30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i - d_i + 0.5])^2$	30/100	[−100,100]	0
$F_7(x) = \sum_{i=1}^{n} i(x_i - d_i)^4$	30/100	[−1.28,1.28]	0

* d is a random vector satisfying $d = rand(1, dim) * (ub - lb) * 0.01$, and is kept the same for all algorithms

In the same sample path, the exploration-exploitation property is controlled by μ^i and Σ^i. When the current sample path leads to a local optimum, OSPO will reset the values of μ^i and Σ^i to jump out of this local optimal trap, and OSPO will generate another sample path using a new exploration-exploitation property. This reset mechanism is promised by the receding sampling strategy.

To summarize, there are mainly two types of active controls of exploration-exploitation property in OSPO:

1) In the sense of SPDF, the adaptive modification of the parameters mainly focuses on the exploitation property in order to find the local optimum;

2) In the sense of receding sampling strategy, the interactions between different sample paths mainly focus on the

TABLE 2. Multi-modal benchmark functions.

Function	Dim	Range	f_{min}
$F_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i-d_i\rvert})$	30/100	[−500,500]	−418.9828 × Dim
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10cos(2\pi x_i)+10]$	30/100	[−5.12,5.12]	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i)\right)+20+e$	30/100	[−32,32]	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30/100	[−600,600]	0
$F_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2+\sum_{i=1}^{n}u(x_i,10,100,4)\}$ where $y_i=1+\frac{x_i+1}{4}$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a<x_i<a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	30/100	[−50,50]	0
$F_{13}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\right\}+\sum_{i=1}^{n}u(x,5,100,4)$ where $u(x_i,a,k,m)$ is the same as F_{12}	30/100	[−50,50]	0

* d is a random vector satisfying $d=rand(1,dim)*(ub-lb)*0.01$, and is kept the same for all algorithms

TABLE 3. Composite benchmark functions.

Function	Dim	Range	f_{min}
$F_{14}(x)=\left(-\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$ where $a=[-32\ -16\ 0\ 16\ 32\ -32\ -16\ 0\ 16\ 32\ -32\ -16\ 0\ 16\ 32\ -32\ -16\ 0\ 16\ 32\ -32\ -16\ 0\ 16\ 32;$ $-32\ -32\ -32\ -32\ -32\ -16\ -16\ -16\ -16\ -16\ 0\ 0\ 0\ 0\ 0\ 16\ 16\ 16\ 16\ 16\ 32\ 32\ 32\ 32\ 32]$	2	[−65,65]	1
$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$ where $a=[0.1957\ 0.1947\ 0.1735\ 0.1600\ 0.0844\ 0.0627\ 0.0456\ 0.0342\ 0.0323\ 0.0235\ 0.0246]$ $b=[4\ 2\ 1\ 0.5\ 0.25\ 0.1667\ 0.125\ 0.1\ 0.0833\ 0.0714\ 0.0625]$	4	[−5,5]	0
$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[−5,5]	-1.0316
$F_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10$	2	[−5,5]	0.398
$F_{18}(x)=[1+(x_1+x_2+1)(19-14x_1+3x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2]\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)$	2	[−2,2]	3

exploration property in order to escape from the local optimal traps.

C. POTENTIAL OF OSPO

OSPO can solve the three difficulties discussed in Introduction to some extent, and these difficulties are written again for ease of reading: 1) the ambiguous definitions of exploration and exploitation; 2) it is hard to determine when to control exploration and exploitation; 3) it is hard to directly control exploration and exploitation.

Firstly, OSPO uses SPDF to directly control the exploration-exploitation property: exploration means lager sample extent and exploitation means smaller sample extent in OSPO. Thus, exploration and exploitation are clear in OSPO formulation.

TABLE 4. The parameters for simulation.

parameter	OSPO
Population N	40
length of sample path l	30
Number of acceptable points q	10
Maximum number of receding sampling $MaxOuter$	5*Dim
Maximum number of function evaluations Max_FE	4e4

Secondly, the metric of exploration and exploitation is the difference between probability and optimality in preferential sampling. Thus, OSPO has the ability to determine when to control its exploration-exploitation property in a more intelligent way.

Thirdly, OSPO can not only control exploration and exploitation adaptively within one single sample path, but

TABLE 5. Results of uni-modal benchmark functions (30 dim).

Function		OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_1	Best	**5.2700E-11**	1.9200E-08	2.7100E-01	3.2944E-02	6.0700E-08	5.6400E-09	1.0389E+04	4.4294E+00
	Worst	2.3796E-04	3.1800E-08	3.4500E+00	1.8031E-01	2.1400E-07	**9.4700E-09**	1.4346E+04	7.3980E+00
	Ave	4.0819E-03	4.3900E-08	8.2100E+00	1.5332E+00	1.4000E-06	**1.2600E-08**	1.7019E+04	1.0566E+01
	Std	9.1552E-04	7.3100E-09	2.2900E+00	3.2296E-01	2.8700E-07	**2.0100E-09**	1.9611E+03	1.6720E+00
F_2	Best	2.8600E-05	**7.1400E-08**	1.5800E-01	1.6210E-01	2.2300E+00	2.5625E-04	4.4558E+01	1.0476E+00
	Worst	7.1636E-02	**1.0900E-07**	4.5400E-01	2.5922E-01	1.9798E+01	1.1010E+00	6.8194E+01	1.3291E+00
	Ave	3.3730E-01	**2.6500E-07**	8.0900E-01	5.2325E-01	1.2405E+02	6.4662E+00	8.6730E+01	1.4402E+00
	Std	7.3191E-02	**4.0800E-08**	1.7600E-01	8.6689E-02	3.0469E+01	1.5731E+00	9.6243E+00	9.3415E-02
F_3	Best	4.2200E-04	**1.0487E-04**	6.9100E+00	1.6501E+03	7.4400E+01	1.2565E+01	4.1998E+04	9.0375E+00
	Worst	7.7740E-02	**1.4417E-03**	1.6100E+01	1.6171E+04	3.2021E+02	8.5148E+01	5.6362E+04	1.5256E+01
	Ave	1.4100E+00	**8.0756E-03**	2.7900E+01	4.6287E+04	6.1067E+02	1.9816E+02	7.1149E+04	2.4227E+01
	Std	2.6354E-01	**2.1599E-03**	5.5500E+00	1.2466E+04	1.3705E+02	5.7268E+01	5.8333E+03	4.0674E+00
F_4	Best	1.3142E-01	**7.8100E-05**	4.2500E-01	1.6643E+00	9.2847E+00	1.6145E+00	7.2494E+01	7.4895E-01
	Worst	3.9074E-01	**1.6440E-03**	6.3200E-01	2.7245E+01	1.8868E+01	6.0716E+00	8.0513E+01	8.3428E-01
	Ave	9.2883E-01	**9.9892E-03**	8.6400E-01	8.6529E+01	3.0439E+01	1.2632E+01	8.8110E+01	8.6035E-01
	Std	2.0170E-01	**2.4540E-03**	1.4500E-01	3.1200E+01	5.7041E+00	3.2972E+00	4.3985E+00	3.0874E-02
F_5	Best	**2.4277E-02**	1.2008E+00	2.9242E+01	2.7345E+01	2.5123E+01	2.3036E+01	1.8639E+07	7.2700E+01
	Worst	1.7042E+01	**1.3670E+01**	5.2828E+01	3.0886E+01	2.1958E+02	1.2895E+02	3.0354E+07	8.9067E+01
	Ave	2.9004E+01	**2.1919E+01**	9.1839E+01	4.1303E+01	2.3012E+03	5.7816E+02	4.4831E+07	9.7213E+01
	Std	1.2168E+01	4.9092E+00	1.6968E+01	**3.6795E+00**	5.2027E+02	1.5518E+02	6.6332E+06	5.7625E+00
F_6	Best	**1.5200E-10**	1.9700E-08	3.8600E-01	4.9127E-02	5.7700E-08	7.3600E-09	1.0602E+04	3.7914E+00
	Worst	8.6000E-04	3.4600E-08	1.2142E+00	3.4491E-01	1.7500E-07	**1.0900E-08**	1.3471E+04	5.6172E+00
	Ave	1.5221E-02	4.9300E-08	4.6899E+00	7.2015E-01	4.1000E-07	**1.5300E-08**	1.6521E+04	7.1790E+00
	Std	2.8685E-03	7.6300E-09	1.0068E+00	2.0396E-01	1.0700E-07	**2.3700E-09**	1.6812E+03	1.0237E+00
F_7	Best	2.2200E-04	3.5161E-03	3.5000E-04	4.4200E-05	7.1500E-02	2.8500E-02	7.7200E+00	**2.2100E-05**
	Worst	5.3900E-03	6.8808E-03	8.6500E-04	1.1385E-03	1.8102E-01	6.7645E-02	1.2182E+01	**2.3063E-04**
	Ave	2.6800E-02	1.4050E-02	1.3000E-03	6.0768E-03	3.1555E-01	1.1671E-01	1.9617E+01	**7.0964E-04**
	Std	7.0900E-03	2.9798E-03	2.9200E-04	1.5253E-03	7.2897E-02	2.4912E-02	3.4860E+00	**1.6369E-04**

TABLE 6. Friedman ranks for the uni-modal test functions (30 dim).

Algorithms	F1	F2	F3	F4	F5	F6	F7	Average Rank	Rank
OSPO	1	2	2	2	1	1	3	**1.71**	**1**
SASS	3	1	1	1	2	3	5	2.29	2
GWO	6	4	3	3	6	6	4	4.57	4
WOA	5	5	7	6	5	5	2	5.00	5
ALO	4	7	6	7	4	4	7	5.57	7
SSA	2	3	5	5	3	2	6	3.71	3
MFO	8	8	8	8	8	8	8	8.00	8
HHO	7	6	4	4	7	7	1	5.14	6

TABLE 7. p-Values of the Wilcoxon ranksum test over uni-modal benchmark functions (30 dim).

Function	OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_1	N/A	1.60E-05	6.80E-08	6.80E-08	1.60E-05	1.60E-05	6.80E-08	6.80E-08
F_2	N/A	6.78E-08	1.66E-07	7.95E-07	6.80E-08	1.04E-04	6.80E-08	6.80E-08
F_3	N/A	6.22E-04	6.80E-08	6.80E-08	6.80E-08	6.80E-08	6.80E-08	6.80E-08
F_4	N/A	6.80E-08	7.70E-05	6.80E-08	6.80E-08	6.80E-08	6.80E-08	1.20E-06
F_5	N/A	5.08E-01	6.80E-08	1.43E-07	1.60E-05	5.17E-06	6.80E-08	6.80E-08
F_6	N/A	1.60E-05	6.79E-08	6.79E-08	1.60E-05	1.60E-05	6.79E-08	6.79E-08
F_7	N/A	1.60E-05	6.80E-08	6.80E-08	1.60E-05	1.60E-05	6.80E-08	6.80E-08

also reset exploration and exploitation in different sample paths in a higher level. Before local optimum has been found, OSPO improves its exploitation ability by preferential sampling; and when the algorithm falls into local optimum, OSPO improves its exploration ability by receding sampling. Thus, OSPO can control exploration and exploitation in an intelligent way.

Since the meanings of exploration and exploitation in OSPO are clear and the exploration-exploitation property can be controlled directly either, OSPO has the potential to change its exploration-exploitation property adaptively online to fit with different optimization problems. In this sense, OSPO has the ability to solve ''at least a vast majority of optimization problems'' proposed in [2].

TABLE 8. Results of multi-modal benchmark functions (30 dim).

Function		OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_8	Best	**-1.2569E+04**	-1.2569E+04	-8.0030E+03	-1.2560E+04	-7.1587E+03	-9.3205E+03	-7.7262E+03	-1.2548E+04
	Worst	**-1.2569E+04**	-1.2541E+04	-6.3845E+03	-1.1972E+04	-7.1531E+03	-7.9400E+03	-7.0430E+03	-1.2542E+04
	Ave	**-1.2569E+04**	-1.2449E+04	-3.7198E+03	-8.8119E+03	-7.1528E+03	-6.0285E+03	-6.3089E+03	-1.2536E+04
	Std	**3.0200E-10**	4.5492E+01	9.7847E+02	9.3398E+02	1.3157E+00	7.7499E+02	4.4288E+02	4.0652E+00
F_9	Best	**1.1100E-10**	8.5022E-02	1.4305E+00	1.7828E-02	6.0692E+01	2.0894E+01	2.6619E+02	1.3143E+00
	Worst	**2.8096E-03**	1.1157E+00	1.0169E+01	6.3971E-01	1.1676E+02	5.6364E+01	3.0054E+02	2.1773E+00
	Ave	**1.7809E-02**	5.5563E+00	2.8297E+01	4.4807E+00	2.2884E+02	9.1536E+01	3.3339E+02	2.9790E+00
	Std	**3.7473E-03**	1.5662E+00	6.1734E+00	1.4303E+00	4.4928E+01	1.7312E+01	1.9410E+01	5.1096E-01
F_{10}	Best	2.8300E-06	**5.7000E-08**	8.9700E-02	6.4846E-02	8.4000E-05	2.1100E-05	1.6543E+01	1.3459E+00
	Worst	1.8032E-01	**3.0300E-06**	5.7100E-01	1.8146E-01	4.5466E+00	1.9277E+00	1.7436E+01	1.5630E+00
	Ave	1.3404E+00	**5.6200E-05**	1.0800E+00	8.3419E-01	1.7181E+01	3.1589E+00	1.8718E+01	1.6715E+00
	Std	4.4548E-01	**1.2500E-05**	2.6100E-01	1.9437E-01	6.0651E+00	8.7030E-01	6.3958E-01	9.1809E-02
F_{11}	Best	**4.2600E-09**	2.5300E-08	9.3204E-02	7.2674E-02	4.2400E-05	2.7400E-08	1.0130E+02	4.7829E-01
	Worst	1.1232E-02	**2.0951E-03**	7.4875E-01	2.1344E-01	9.1894E-03	1.6099E-02	1.2696E+02	6.4827E-01
	Ave	4.6556E-02	**1.2316E-02**	1.0344E+00	5.3115E-01	2.4965E-02	5.1640E-02	1.5925E+02	8.3325E-01
	Std	1.3068E-02	**3.8574E-03**	2.6431E-01	1.2455E-01	8.7058E-03	1.7744E-02	1.7070E+01	1.0432E-01
F_{12}	Best	**6.4700E-12**	1.8900E-08	2.4600E-03	1.1500E-04	5.1479E+00	1.1339E+00	1.2562E+07	1.6385E-02
	Worst	**1.7900E-05**	5.1835E-03	5.8995E-03	1.8298E-03	9.6267E+00	5.3495E+00	5.9976E+07	2.5109E-02
	Ave	**1.2600E-04**	1.0367E-01	1.4760E-02	1.0464E-02	1.5144E+01	9.7898E+00	1.1012E+08	3.0550E-02
	Std	**3.3200E-05**	2.3181E-02	3.0564E-03	2.7864E-03	2.8822E+00	2.6009E+00	2.4460E+07	3.8307E-03
F_{13}	Best	**9.2100E-11**	1.7800E-08	3.0514E-01	3.0783E-02	9.8500E-08	4.2600E-10	5.1645E+07	2.9728E-01
	Worst	1.0290E-02	**3.3700E-08**	8.9347E-01	4.2846E-01	3.3988E+00	1.0884E-02	1.4098E+08	3.5202E-01
	Ave	9.7371E-02	**6.1100E-08**	1.6076E+00	1.0103E+00	4.2011E+01	5.4779E-02	2.0881E+08	3.6931E-01
	Std	1.8544E-02	**1.0800E-08**	4.3813E-01	3.1736E-01	1.0644E+01	1.5030E-02	3.8564E+07	2.1972E-02

TABLE 9. Friedman ranks for the multi-modal test functions (30 dim).

Algorithms	F8	F9	F10	F11	F12	F13	Average Rank	Rank
OSPO	1	1	2	1	1	1	**1.17**	**1**
SASS	2	3	1	2	2	3	2.17	2
GWO	6	5	6	6	4	7	5.67	6
WOA	3	2	5	5	3	5	3.83	3
ALO	8	7	4	4	7	4	5.67	7
SSA	5	6	3	3	6	2	4.17	4
MFO	7	8	8	8	8	8	7.83	8
HHO	4	4	7	7	5	6	5.50	5

TABLE 10. p-Values of the Wilcoxon ranksum test over multi-modal benchmark functions (30 dim).

Function	OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_8	N/A	1.44E-02	1.64E-01	7.71E-03	6.80E-08	6.80E-08	6.80E-08	2.36E-06
F_9	N/A	8.01E-09	8.01E-09	8.01E-09	7.43E-10	8.01E-09	8.01E-09	8.01E-09
F_{10}	N/A	6.80E-08	6.80E-08	6.80E-08	6.80E-08	6.80E-08	6.80E-08	6.80E-08
F_{11}	N/A	1.92E-07	1.61E-04	1.61E-04	1.80E-06	2.67E-06	6.79E-08	6.79E-08
F_{12}	N/A	7.56E-01	6.80E-08	6.80E-08	2.98E-01	1.20E-01	6.80E-08	6.80E-08
F_{13}	N/A	4.70E-03	6.80E-08	6.80E-08	6.80E-08	6.80E-08	6.80E-08	6.80E-08

The ability of OSPO is then verified by benchmark functions as well as real-world optimization problems in the next section.

IV. RESULTS

A. BENCHMARK FUNCTIONS

In this section, 19 test functions are selected to demonstrate the effectiveness of the proposed OSPO algorithm. The test functions are classified into three main groups: uni-modal (Table 1), multi-modal (Table 2), and composite functions (Table 3).

As these names suggest, uni-modal test functions have only one single optimum, and it can measure the abilities of the exploitation and the convergence of algorithms. While in multi-model benchmark functions, they have several optima which make them more complicated than uni-modal functions. One of these optima is denoted as global optimum and the others are called local optima. Algorithms should avoid falling into local optima and converge to the global optimum. Therefore, the superiorities of exploration and local optima avoidance of algorithms are measured by multi-modal test functions. As for composite functions, they are mainly the combined, rotated, shifted, and biased versions of other uni-modal and multi-modal test functions. This kind of test functions simulate the difficulties of real search spaces by providing a large number of local optima for different regions

TABLE 11. Results of uni-modal benchmark functions (100 dim).

Function		OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_1	Best	5.49738E-04	2.21000E-07	3.92680E+01	4.58521E-01	2.44574E+04	**1.34000E-07**	2.05418E+05	2.79300E+01
	Worst	5.84011E-03	8.62000E-07	5.10350E+01	7.65373E-01	4.43228E+04	**1.85000E-07**	2.33023E+05	3.02398E+01
	Ave	3.56982E-02	6.51000E-06	6.30755E+01	1.47517E+00	7.97420E+04	**2.34000E-07**	2.48629E+05	3.15022E+01
	Std	7.70438E-03	1.40000E-06	5.41850E+00	2.57762E-01	1.65135E+04	**2.64000E-08**	1.11602E+04	9.78332E-01
F_2	Best	1.79136E-01	**1.02000E-03**	4.14136E+00	7.75968E-01	1.61418E+02	4.72344E+00	6.88000E+25	4.37907E+00
	Worst	4.16132E-01	**1.05000E-02**	5.07026E+00	1.05376E+00	8.97451E+03	1.06058E+01	6.37000E+33	4.57234E+00
	Ave	6.63376E-01	**5.47000E-02**	5.74704E+00	1.51741E+00	1.06205E+05	1.60257E+01	1.14000E+35	4.80521E+00
	Std	1.34300E-01	**1.34000E-02**	4.38822E-01	1.64205E-01	2.75749E+04	3.39314E+00	2.53000E+34	1.25703E-01
F_3	Best	**3.73743E+00**	2.58114E+02	1.60653E+02	1.45367E+05	8.36962E+04	7.87707E+03	4.39058E+05	6.03223E+01
	Worst	**3.24431E+01**	4.94109E+02	4.05186E+02	4.91670E+05	1.62201E+05	1.70474E+04	5.47604E+05	1.04289E+02
	Ave	**1.04619E+02**	9.41656E+02	5.95712E+02	6.20459E+05	2.94981E+05	2.93125E+04	6.79729E+05	1.33545E+02
	Std	2.90934E+01	1.76349E+02	1.06140E+02	1.29091E+05	5.43832E+04	6.15650E+03	7.05341E+04	**2.32900E+01**
F_4	Best	**5.57483E-01**	1.73000E+01	1.26124E+00	5.08716E+01	3.60190E+01	1.75812E+01	9.39239E+01	9.67780E-01
	Worst	**7.51251E-01**	2.14600E+01	1.26124E+00	8.12648E+01	4.87528E+01	2.47762E+01	9.57141E+01	9.83397E-01
	Ave	**9.14465E-01**	2.59491E+01	1.26125E+00	9.58538E+01	6.31756E+01	3.22131E+01	9.72483E+01	9.91949E-01
	Std	8.20106E-02	2.53799E+00	**6.85000E-07**	1.33948E+01	8.28172E+00	3.66000E+00	8.74565E-01	8.40937E-03
F_5	Best	**2.33921E-01**	9.51513E+01	7.46942E+02	1.01699E+02	1.06037E+07	9.40090E+01	8.87256E+08	3.63455E+02
	Worst	**9.09487E+01**	2.07044E+02	8.43429E+02	1.06334E+02	2.57479E+07	3.26959E+02	1.08422E+09	3.83754E+02
	Ave	**9.83751E+01**	2.92766E+02	9.48223E+02	1.16673E+02	6.28171E+07	1.88954E+03	1.23646E+09	3.93844E+02
	Std	2.46638E+01	4.71222E+01	5.54519E+01	**3.86297E+00**	1.34847E+07	4.12717E+02	9.21953E+07	9.41308E+00
F_6	Best	8.67091E-04	2.00000E-07	1.50327E+01	1.92142E-01	2.28284E+04	**1.50000E-07**	2.05147E+05	2.85788E+01
	Worst	2.35410E-02	3.81000E-07	1.87235E+01	4.30364E-01	3.83993E+04	**1.87000E-07**	2.24374E+05	3.02851E+01
	Ave	1.04943E-01	1.33000E-06	2.37664E+01	8.34116E-01	6.94251E+04	**2.18000E-07**	2.59778E+05	3.16945E+01
	Std	2.08433E-02	2.40000E-07	2.43944E+00	1.71678E-01	1.15011E+04	**2.28000E-08**	1.26300E+04	1.03870E+00
F_7	Best	5.66484E-04	8.51356E-02	1.07000E-03	1.99252E-04	1.43193E+01	3.90000E-01	1.27000E+03	**2.70000E-05**
	Worst	1.21840E-03	1.64475E-01	1.93605E-03	9.38327E-04	4.24476E+01	5.57876E-01	1.54663E+03	**1.84297E-04**
	Ave	3.57032E-03	3.15206E-01	4.62366E-03	3.33285E-03	8.10512E+01	6.85899E-01	1.89070E+03	**5.69429E-04**
	Std	5.74762E-04	6.24445E-02	8.49892E-04	8.74090E-04	1.87580E+01	1.00737E-01	1.65290E+02	**1.44000E-04**

TABLE 12. Friedman ranks for the uni-modal test functions (100 dim).

Algorithms	F1	F2	F3	F4	F5	F6	F7	Average Rank	Rank
OSPO	3	2	1	1	1	3	3	**2.00**	**1**
SASS	2	1	4	4	3	2	5	3.00	2
GWO	6	4	3	3	6	5	4	4.43	5
WOA	4	3	7	7	4	4	2	4.43	6
ALO	7	7	6	6	7	7	7	6.71	7
SSA	1	6	5	5	2	1	6	3.71	3
MFO	8	8	8	8	8	8	8	8.00	8
HHO	5	5	2	2	5	6	1	3.71	4

TABLE 13. p-Values of the Wilcoxon ranksum test over uni-modal benchmark functions (100 dim).

Function	OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_1	N/A	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09
F_2	N/A	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09
F_3	N/A	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	5.45E-08
F_4	N/A	3.01E-09	2.34E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09
F_5	N/A	7.59E-08	3.01E-09	3.01E-09	3.01E-09	8.46E-05	3.01E-09	3.01E-09
F_6	N/A	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09
F_7	N/A	3.01E-09	7.79E-05	9.21E-03	3.01E-09	3.01E-09	3.01E-09	3.40E-09

of the search space [3]. Algorithms should keep a satisfying balance between exploration and exploitation to find the global optimum of this kind of test functions. Hence, the effectiveness of algorithms to solve real-world problems is tested by composite functions.

Since OSPO has the ability to modify its exploration-exploitation property adaptively, it is expected to have a dynamic balance between exploration and exploitation. In addition, OSPO is believed to solve different kinds of problems with satisfactory performance. In order to verify the performance of OSPO, seven metaheuristics are used to compare with OSPO, and these comparative algorithms are: The Ant Lion Optimizer (ALO) [3], Moth-flame optimization algorithm (MFO) [5], Harris hawks optimization (HHO) [41], Salp Swarm Algorithm (SSA) [7], Grey Wolf Optimizer (GWO) [36], Whale Optimization Algorithm (WOA) [37]

TABLE 14. Results of multi-modal benchmark functions (100 dim).

Function		OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_8	Best	**-4.1898E+04**	-4.0838E+04	-2.0216E+04	-4.1898E+04	-2.3353E+04	-2.9484E+04	-1.9518E+04	-4.1810E+04
	Worst	**-4.1898E+04**	-4.0237E+04	-1.6088E+04	-4.0997E+04	-2.3252E+04	-2.7020E+04	-1.7550E+04	-4.1801E+04
	Ave	**-4.1898E+04**	-3.9738E+04	-6.6179E+03	-3.5589E+04	-2.3247E+04	-2.3875E+04	-1.4267E+04	-4.1798E+04
	Std	**5.20024E-04**	2.60197E+02	3.88655E+03	1.71785E+03	2.36819E+01	1.33274E+03	1.44099E+03	3.21304E+00
F_9	Best	**5.97249E-04**	2.77430E+01	3.47922E+01	2.75997E-01	4.89862E+02	1.09445E+02	1.37578E+03	9.67021E+00
	Worst	**8.24953E-03**	3.48189E+01	4.34934E+01	4.64601E-01	6.35447E+02	1.82575E+02	1.48610E+03	1.13172E+01
	Ave	**5.98053E-02**	4.61990E+01	6.09785E+01	9.48083E-01	8.10049E+02	2.54709E+02	1.54617E+03	1.32514E+01
	Std	**1.20784E-02**	3.81674E+00	6.57209E+00	1.80799E-01	8.36788E+01	3.98041E+01	4.48779E+01	8.49536E-01
F_{10}	Best	**3.46035E-03**	3.31000E+00	1.79565E+00	1.23341E-01	1.52137E+01	3.31000E+00	2.07335E+01	1.63343E+00
	Worst	**7.37152E-02**	4.05000E+00	2.05233E+00	1.73727E-01	1.70807E+01	4.79038E+00	2.08734E+01	1.71727E+00
	Ave	1.32322E+00	5.18000E+00	2.21424E+00	**2.86645E-01**	1.82122E+01	6.76292E+00	2.09894E+01	1.78451E+00
	Std	2.46526E-01	5.22000E-01	1.10210E-01	4.76615E-02	6.84010E-01	9.19612E-01	7.50374E-02	**4.31567E-02**
F_{11}	Best	1.99686E-03	**2.58000E-07**	1.31177E+00	3.38241E-01	1.92603E+02	1.50000E-05	1.72851E+03	1.08268E+00
	Worst	1.73510E-02	1.96256E-02	1.47607E+00	4.74390E-01	3.48614E+02	**1.46422E-03**	2.03387E+03	1.10092E+00
	Ave	6.36985E-02	6.88466E-02	1.56970E+00	6.80839E-01	6.58278E+02	**1.03041E-02**	2.25647E+03	1.13165E+00
	Std	1.72296E-02	2.41393E-02	6.14470E-02	1.02836E-01	1.10861E+02	**3.06112E-03**	1.68362E+02	1.39817E-02
F_{12}	Best	**6.26377E-07**	1.67000E+00	9.40417E-03	2.42866E-04	6.95047E+04	8.15414E+00	2.36004E+09	2.13997E-02
	Worst	**1.64545E-05**	5.31352E+00	1.44342E-02	4.91729E-04	3.79541E+06	1.15351E+01	2.78481E+09	2.72247E-02
	Ave	**1.05166E-04**	1.19648E+01	3.90573E-02	1.13653E-03	1.81859E+07	1.73273E+01	3.11490E+09	3.27543E-02
	Std	**2.54984E-05**	2.82963E+00	6.01592E-03	2.27728E-04	4.42975E+06	2.56918E+00	2.10567E+08	4.06540E-03
F_{13}	Best	**3.23332E-05**	4.94000E-03	1.42734E+01	7.06764E-01	1.02783E+07	1.23000E+02	3.88092E+09	1.09838E+00
	Worst	**2.39951E-02**	1.03000E+01	1.61326E+01	2.53139E+00	4.53883E+07	1.59004E+02	5.02878E+09	1.15048E+00
	Ave	**8.46645E-02**	1.20000E+02	1.90036E+01	5.95241E+00	1.26083E+08	1.97181E+02	5.80622E+09	1.17367E+00
	Std	**2.34452E-02**	2.63000E+01	1.05191E+00	1.44952E+00	2.82737E+07	1.99807E+01	4.12069E+08	2.50650E-02

TABLE 15. Friedman ranks for the multi-modal test functions (100 dim).

Algorithms	F8	F9	F10	F11	F12	F13	Average Rank	Rank
OSPO	1	1	1	3	1	1	**1.33**	**1**
SASS	4	4	5	1	5	2	3.50	3
GWO	7	5	4	6	3	5	5.00	5
WOA	2	2	2	4	2	3	2.50	2
ALO	6	7	7	7	7	7	6.83	7
SSA	5	6	6	2	6	6	5.17	6
MFO	8	8	8	8	8	8	8.00	8
HHO	3	3	3	5	4	4	3.67	4

TABLE 16. p-Values of the Wilcoxon ranksum test over multi-modal benchmark functions (100 dim).

Function	OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_8	N/A	3.01E-09	3.01E-09	3.01E-09	1.06E-09	3.01E-09	3.01E-09	3.01E-09
F_9	N/A	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09
F_{10}	N/A	3.01E-09	3.01E-09	2.11E-06	3.01E-09	3.01E-09	3.01E-09	3.01E-09
F_{11}	N/A	3.78E-01	3.01E-09	3.01E-09	3.01E-09	2.01E-07	3.01E-09	3.01E-09
F_{12}	N/A	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09
F_{13}	N/A	9.60E-06	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09	3.01E-09

and Self-Adaptive Spherical Search (SASS) [42]. The last comparative algorithm SASS is an adaptive metaheuristic algorithm which is the winner of the ''CEC-2020 Competitions on Real-World Single Objective Constrained Optimization''.

To make the quantitative results more convincing, we run each algorithm 30 times on every test function, and the best approximated solutions as well as the worst, average and standard deviation of these 30 solutions are reported. These four values can verify which algorithm has the superiority when solving the test functions. In addition, Friedman ranks based on best solutions are also used to compare the performance of the algorithms, which can verify the optimality of the algorithms. Finally, Wilcoxon's rank-sum test at the 0.05 significance level is employed to compare the performance of algorithms. The Wilcoxon's rank-sum test is a non-parametric test in statics that can be used to determine whether two sets of solutions are different statistically significant. It also tests the null hypothesis as to whether both populations are of the same distribution. This statistical test returns a parameter called p-vale. A p-value determines the significance level of two algorithms. In this contribution an algorithm is statistically significant if and only if it results in a p-value less than 0.05.

TABLE 17. Results of composite benchmark functions.

Function		OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_{14}	Best	**9.9800E-01**	9.9800E-01	9.9800E-01	9.9800E-01	9.9800E-01	9.9800E-01	9.9800E-01	9.9800E-01
	Worst	**9.9800E-01**	9.9800E-01	4.3321E+00	2.1312E+00	2.0384E+00	9.9800E-01	1.0990E+00	1.0477E+00
	Ave	**9.9800E-01**	9.9800E-01	1.0763E+01	1.0763E+01	5.9288E+00	9.9800E-01	2.0244E+00	1.9920E+00
	Std	2.5100E-16	2.6700E-12	3.9061E+00	2.2276E+00	1.3013E+00	**1.1400E-16**	3.1098E-01	2.2227E-01
F_{15}	Best	**3.0749E-04**	3.0749E-04	3.0749E-04	3.0795E-04	4.2430E-04	6.5162E-04	7.5616E-04	3.0985E-04
	Worst	**3.0749E-04**	3.0749E-04	4.5018E-03	6.6699E-04	3.7269E-03	8.5716E-04	2.7570E-03	3.3236E-04
	Ave	**3.0749E-04**	3.0749E-04	2.0363E-02	1.5089E-03	2.0363E-02	1.2339E-03	2.0363E-02	3.7638E-04
	Std	**1.8300E-17**	3.4500E-10	8.1449E-03	4.1356E-04	7.1727E-03	2.2141E-04	6.0221E-03	1.9700E-05
F_{16}	Best	**-1.0316E+00**	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00
	Worst	**-1.0316E+00**	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00
	Ave	**-1.0316E+00**	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00
	Std	5.3800E-16	2.8400E-11	2.6600E-09	1.1300E-11	4.1200E-14	7.4800E-15	**2.2800E-16**	1.3600E-09
F_{17}	Best	**3.9789E-01**	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01
	Worst	**3.9789E-01**	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01
	Ave	**3.9789E-01**	3.9789E-01	3.9800E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9789E-01	3.9791E-01
	Std	**0.0000E+00**	2.5500E-11	2.5400E-05	1.3900E-07	1.9900E-14	2.0000E-14	0.0000E+00	5.7100E-06
F_{18}	Best	**3.0000E+00**	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00
	Worst	2.9100E+01	**3.0000E+00**	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00
	Ave	3.0000E+01	**3.0000E+00**	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00
	Std	4.9295E+00	2.0900E-11	4.7600E-06	1.1100E-05	1.8100E-13	3.7100E-14	**1.1200E-15**	5.0200E-08
F_{19}	Best	**-3.8628E+00**	-3.8628E+00	-3.8628E+00	-3.8628E+00	-3.8628E+00	-3.8628E+00	-3.8628E+00	-3.8628E+00
	Worst	**-3.8628E+00**	-3.8628E+00	-3.8612E+00	-3.8609E+00	-3.8628E+00	-3.8628E+00	-3.8628E+00	-3.8611E+00
	Ave	**-3.8628E+00**	-3.8628E+00	-3.8549E+00	-3.8548E+00	-3.8628E+00	-3.8628E+00	-3.8628E+00	-3.8535E+00
	Std	**2.2500E-15**	8.3400E-11	3.1448E-03	2.6697E-03	9.6100E-15	2.5300E-14	2.2800E-15	2.2611E-03

TABLE 18. Friedman ranks for the composite test functions.

Algorithms	F14	F15	F16	F17	F18	F19	Average Rank	Rank
OSPO	1	1	1	1	1	1	1.00	**1**
SASS	2	2	2	2	2	2	2.00	2
GWO	3	3	3	3	3	3	3.00	3
WOA	4	4	4	4	4	4	4.00	4
ALO	5	6	5	5	5	5	5.17	5
SSA	6	7	6	6	6	6	6.17	6
MFO	7	8	7	7	7	7	7.17	7
HHO	8	5	8	8	8	8	7.50	8

TABLE 19. p-Values of the Wilcoxon ranksum test over composite benchmark functions.

Function	OSPO	SASS	GWO	WOA	ALO	SSA	MFO	HHO
F_{14}	N/A	3.14E-02	6.61E-08	7.68E-08	2.38E-02	5.69E-01	6.61E-08	6.61E-08
F_{15}	N/A	N/A	7.80E-06	2.01E-03	1.61E-04	N/A	1.63E-01	2.07E-03
F_{16}	N/A	1.29E-04	7.88E-09	8.01E-09	7.99E-09	7.95E-09	7.99E-09	8.01E-09
F_{17}	N/A	N/A	6.57E-05	N/A	N/A	N/A	N/A	3.42E-01
F_{18}	N/A	N/A	8.01E-09	6.68E-05	N/A	N/A	N/A	2.55E-05
F_{19}	N/A	3.13E-09	2.78E-07	2.78E-07	3.13E-09	3.13E-09	3.13E-09	1.06E-07

The parameters of OSPO for simulation is defined in Table 4, and the parameters of the comparative algorithms are kept the same as their initial values in literatures. Since the maximum number of the function evaluations of all the algorithms are equal, the comparisons are quite fair.

The corresponding MATLAB APP demo is illustrated in Fig. 7, and is available online:

https://github.com/JiahongXu123/OSPO-algorithm.git.

1) UNI-MODAL TEST FUNCTIONS

The simulation results of OSPO and 7 comparative algorithms on the uni-modal test functions with low dimensions (30 dimensions) are shown in Table 5. Table 6 illustrates the Friedman ranks based on the best solutions of the 8 algorithms, and Table 7 reports the p-values of the Wilcoxon's rank-sum test.

According to the results of the algorithms on uni-modal test functions shown in Table 5 and Table 6, it is evident that OSPO algorithm can perform well on all these test functions. OSPO can solve these problems with best or at least promising solutions. The Friedman ranks results reported in Table 6 illustrate that OSPO algorithm has the best performance when dealing with different uni-modal test functions, and the average rank value is 1.71. The p-values reported

TABLE 20. 21 real-world constrained optimization problems.

Problem	Name	D	g	h	Max_FEs
RW1	Optimal Operation of Alkylation Unit	7	14	0	1e5
RW2	Reactor Network Design (RND)	6	1	4	1e5
RW3	Process synthesis problem 1	2	2	0	1e5
RW4	Process synthesis and design problem	3	1	1	1e5
RW5	Process flow sheeting problem	3	3	0	1e5
RW6	Process synthesis problem 2	7	9	0	1e5
RW7	Process design Problem	5	3	0	1e5
RW8	Multi-product batch plant	10	10	0	1e5
RW9	Weight Minimization of a Speed Reducer	7	11	0	1e5
RW10	Tension/compression spring design (case 1)	3	3	0	1e5
RW11	Welded beam design	4	5	0	1e5
RW12	Three-bar truss design problem	2	3	0	1e5
RW13	Planetary gear train design optimization problem	9	10	1	1e5
RW14	Step-cone pulley problem	5	8	3	1e5
RW15	Robot gripper problem	7	7	0	1e5
RW16	Hydro-static thrust bearing design problem	4	7	0	1e5
RW17	10-bar truss design	10	3	0	1e5
RW18	Gas Transmission Compressor Design (GTCD)	4	1	0	1e5
RW19	Tension/compression spring design (case 2)	3	8	0	1e5
RW20	Himmelblau's Function	5	6	0	1e5
RW21	Topology Optimization	30	30	0	2e5

* D is the total number of decision variables of the problem, g is the number of inequality constraints and *h* is the number of equality constraints. Max_FEs is the maximum number of function evaluations for optimization.

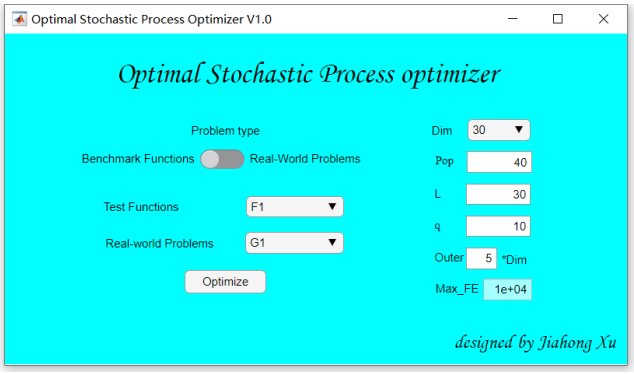

FIGURE 7. The MATLAB APP demo for OSPO.

in Table 7 also show that this superiority is statistically significant since the p-values are much less than 0.05. The SASS algorithm takes the second place with the average rank value 2.29.

Considering the characteristics of uni-modal test functions, it can be stated that OSPO has satisfactory exploitation ability. This high exploitation assists the OSPO algorithm to successfully converge towards the optimal solution and exploit it accurately.

2) MULTI-MODAL TEST FUNCTIONS

The simulation results of OSPO and 7 comparative algorithms on the multi-modal test functions with low dimensions (30 dimensions) are shown in Table 8. Table 9 illustrates the Friedman ranks based on the best solutions of the 8 algorithms, and Table 10 illustrates the p-values of the Wilcoxon's rank-sum test.

According to the results of the algorithms on multi-modal test functions in Table 8 and Table 9, it is evident that OSPO algorithm can perform well on all these test functions. OSPO can solve these problems with best or at least promising solutions. The Friedman ranks results reported in Table 9 illustrate that OSPO algorithm has the best performance when dealing with different multi-modal test functions, and the average rank value is 1.17. The p-values reported in Table 10 also show that this superiority is statistically significant since the p-values are much less than 0.05 for most of the cases. The SASS algorithm takes the second place with the average rank value 2.27.

Considering the characteristics of multi-modal test functions, it can be stated that OSPO has satisfactory exploration ability. This high exploration assists the OSPO algorithm to finally converge towards the global optimum.

3) HIGHER DIMENSION OF TEST FUNCTIONS

In order to verify the effectiveness of the OSPO algorithm when the dimensions are large, the dimensions of uni-modal and multi-modal test functions are increased to 100. The simulation results are reported in Table 11 to Table 16.

According to the results of the algorithms on higher-dimension test functions in Table 11 and Table 14, it is evident that OSPO algorithm can still perform well on both uni-modal and multi-modal test functions even when their dimensions have been increased to 100.

The Friedman ranks results reported in Table 12 and Table 15 illustrate that OSPO algorithm has the best performance when dealing different test functions with 100 dimensions, and the average rank values are 2.00 and 1.33 respectively. The superiority of OSPO is more evident on multi-modal test functions. SASS takes the second place over unimodal test functions and the third place over multi-modal test functions.

TABLE 21. Results of 21 real-world constrained optimization problems.

Problem		OSPO	SASS	COLSHADE	EnMODE	sCMAgES
RW_1	Best	**-4.53000E+03**	-1.42719E+02	-4.52912E+03	-4.52912E+03	-4.52668E+03
	Worst	-4.31000E+03	-1.42719E+02	-4.36668E+03	**-4.52912E+03**	-4.15560E+03
	Ave	-3.71000E+03	-1.42719E+02	-3.71691E+03	**-4.52912E+03**	-3.32360E+03
	Std	2.64000E+02	1.07000E-05	3.33325E+02	**1.62000E-12**	3.82430E+02
RW_2	Best	-3.83335E-01	**-3.88260E-01**	-3.88260E-01	-3.88260E-01	-3.88239E-01
	Worst	-3.77800E-01	**-3.88260E-01**	-3.80364E-01	-3.74543E-01	-3.87560E-01
	Ave	-3.67946E-01	**-3.88255E-01**	-3.58218E-01	-3.69744E-01	-3.74859E-01
	Std	3.46231E-03	**1.27000E-06**	1.15739E-02	3.45799E-03	2.98951E-03
RW_3	Best	**2.00000E+00**	2.00000E+00	2.00000E+00	2.00000E+00	2.00000E+00
	Worst	**2.00000E+00**	2.00000E+00	2.00000E+00	2.00000E+00	2.00000E+00
	Ave	**2.00000E+00**	2.00000E+00	2.00000E+00	2.00000E+00	2.00000E+00
	Std	**6.41000E-17**	6.12000E-15	2.60000E-16	2.28000E-16	1.84000E-16
RW_4	Best	**2.54173E+00**	2.55765E+00	2.55765E+00	2.55765E+00	2.55765E+00
	Worst	**2.54277E+00**	2.55765E+00	2.55765E+00	2.55765E+00	2.55765E+00
	Ave	**2.54370E+00**	2.55765E+00	2.55765E+00	2.55765E+00	2.55765E+00
	Std	5.47172E-04	2.74000E-10	9.11000E-16	**9.11000E-16**	8.01000E-09
RW_5	Best	**1.07654E+00**	1.07654E+00	1.07654E+00	1.07654E+00	1.07654E+00
	Worst	**1.07654E+00**	1.07654E+00	1.08522E+00	1.17194E+00	1.07654E+00
	Ave	**1.07654E+00**	1.07654E+00	1.25000E+00	1.25000E+00	1.07654E+00
	Std	**4.53000E-16**	1.18000E-13	3.87861E-02	8.85355E-02	2.40000E-14
RW_6	Best	**2.92483E+00**	2.92483E+00	2.92483E+00	2.92483E+00	2.92483E+00
	Worst	3.07461E+00	**2.92483E+00**	2.93600E+00	2.92483E+00	2.93411E+00
	Ave	3.92483E+00	**2.92483E+00**	3.08173E+00	2.92483E+00	2.95336E+00
	Std	3.19522E-01	8.63000E-10	3.52393E-02	**9.11000E-16**	1.15242E-02
RW_7	Best	**2.63963E+04**	2.68874E+04	2.68874E+04	2.68874E+04	2.68874E+04
	Worst	2.71240E+04	**2.68874E+04**	2.68874E+04	2.68874E+04	2.68874E+04
	Ave	2.80292E+04	**2.68874E+04**	2.68874E+04	2.68874E+04	2.68874E+04
	Std	4.17040E+02	**4.07000E-10**	7.46000E-12	1.12000E-11	1.12000E-11
RW_8	Best	**5.36401E+04**	5.85055E+04	5.85054E+04	5.85054E+04	5.36994E+04
	Worst	6.01446E+04	5.87752E+04	5.85054E+04	5.85054E+04	**5.49516E+04**
	Ave	7.36803E+04	6.18773E+04	**5.85054E+04**	5.85054E+04	5.92723E+04
	Std	6.28830E+03	9.33629E+02	**2.83000E-11**	7.83000E-09	1.67633E+03
RW_9	Best	**2.93640E+03**	2.99442E+03	2.99442E+03	2.99442E+03	2.99442E+03
	Worst	**2.98860E+03**	2.99442E+03	2.99442E+03	2.99442E+03	2.99442E+03
	Ave	3.00371E+03	**2.99442E+03**	2.99442E+03	2.99442E+03	2.99442E+03
	Std	1.56765E+01	5.91000E-09	**9.33000E-13**	1.40000E-12	7.66000E-12
RW_{10}	Best	1.26710E-02	**1.26652E-02**	1.26652E-02	1.26652E-02	1.26653E-02
	Worst	1.28078E-02	**1.26652E-02**	1.26652E-02	1.27110E-02	1.26671E-02
	Ave	1.31336E-02	**1.26652E-02**	1.26653E-02	1.27191E-02	1.26745E-02
	Std	1.11414E-04	**8.13000E-10**	5.11000E-09	1.97000E-05	2.53000E-06
RW_{11}	Best	1.67022E+00	**1.67022E+00**	1.67022E+00	1.67022E+00	1.67022E+00
	Worst	1.69146E+00	**1.67022E+00**	1.67022E+00	1.67022E+00	1.67022E+00
	Ave	1.82640E+00	**1.67022E+00**	1.67022E+00	1.67022E+00	1.67022E+00
	Std	4.72533E-02	7.72000E-13	6.83000E-16	**6.97000E-16**	1.57000E-13
RW_{12}	Best	**2.63896E+02**	2.63896E+02	2.63896E+02	2.63896E+02	2.63896E+02
	Worst	**2.63896E+02**	2.63896E+02	2.63896E+02	2.63896E+02	2.63896E+02
	Ave	**2.63896E+02**	2.63896E+02	2.63896E+02	2.63896E+02	2.63896E+02
	Std	1.44000E-13	1.55000E-12	**0.00000E+00**	0.00000E+00	6.66000E-13
RW_{13}	Best	5.37059E-01	**5.25589E-01**	5.25589E-01	5.25769E-01	5.25967E-01
	Worst	5.66935E-01	7.50359E-01	5.34538E-01	**5.26943E-01**	5.31063E-01
	Ave	7.49534E-01	1.54190E+00	5.67003E-01	5.30000E-01	5.56667E-01
	Std	5.51106E-02	3.36885E-01	9.67291E-03	**1.66298E-03**	6.63094E-03
RW_{14}	Best	1.64398E+01	**1.60699E+01**	1.60699E+01	1.60699E+01	1.60700E+01
	Worst	1.78634E+01	**1.60699E+01**	1.60699E+01	1.60699E+01	1.62332E+01
	Ave	1.86590E+01	**1.60699E+01**	1.60699E+01	1.60699E+01	1.67314E+01
	Std	6.15872E-01	3.47000E-09	**4.89000E-15**	3.04000E-14	1.84998E-01
RW_{15}	Best	2.70831E+00	**2.54379E+00**	2.54379E+00	2.54379E+00	2.61969E+00
	Worst	3.50774E+00	**2.54379E+00**	2.54379E+00	2.54379E+00	2.88049E+00
	Ave	4.44867E+00	**2.54379E+00**	2.54379E+00	2.54379E+00	3.34953E+00
	Std	3.99562E-01	5.27000E-09	**5.11000E-14**	4.75000E-12	1.90884E-01
RW_{16}	Best	1.76014E+03	**1.61612E+03**	1.61612E+03	1.61612E+03	2.13571E+03
	Worst	2.14051E+03	1.61696E+03	1.62173E+03	**1.61612E+03**	3.01664E+03
	Ave	2.54814E+03	1.63688E+03	1.66895E+03	**1.61612E+03**	3.62473E+03

TABLE 21. *(Continued.)* **Results of 21 real-world constrained optimization problems.**

RW_{17}	Std	1.93788E+02	4.15098E+00	1.47019E+01	**1.00000E-10**	3.83711E+02
	Best	5.24519E+02	5.24458E+02	**5.24451E+02**	5.24451E+02	5.24504E+02
	Worst	5.25742E+02	5.24472E+02	**5.24451E+02**	5.24451E+02	5.24667E+02
	Ave	5.29139E+02	5.24509E+02	**5.24451E+02**	5.24451E+02	5.24859E+02
RW_{18}	Std	1.03362E+00	9.84351E-03	**1.05000E-09**	4.32000E-09	1.13582E-01
	Best	**2.96490E+06**	2.96490E+06	2.96490E+06	2.96490E+06	2.96490E+06
	Worst	2.97295E+06	**2.96490E+06**	2.96490E+06	2.96490E+06	2.96490E+06
	Ave	2.99850E+06	**2.96490E+06**	2.96490E+06	2.96490E+06	2.96494E+06
RW_{19}	Std	9.80210E+03	1.74000E-09	**9.56000E-10**	9.56000E-10	1.08536E+01
	Best	**2.65856E+00**	2.65856E+00	2.65856E+00	2.65856E+00	2.97530E+00
	Worst	2.87157E+00	**2.65856E+00**	2.65856E+00	2.70743E+00	4.28976E+00
	Ave	3.62625E+00	**2.65856E+00**	2.65856E+00	3.63593E+00	6.35519E+00
RW_{20}	Std	1.91827E-01	2.28000E-11	**4.56000E-16**	2.18548E-01	9.08792E-01
	Best	**-3.06655E+04**	-3.06655E+04	-3.06655E+04	-3.06655E+04	-3.06655E+04
	Worst	-3.05908E+04	**-3.06655E+04**	-3.06655E+04	-3.06655E+04	-3.06655E+04
	Ave	-3.02212E+04	**-3.06655E+04**	-3.06655E+04	-3.06655E+04	-3.06655E+04
RW_{21}	Std	1.08757E+02	2.17000E-09	**8.35000E-13**	3.73000E-12	2.01000E-10
	Best	**2.63935E+00**	2.63935E+00	2.63935E+00	2.63935E+00	2.63935E+00
	Worst	2.63936E+00	**2.63935E+00**	2.63935E+00	2.63935E+00	2.63935E+00
	Ave	2.63948E+00	**2.63935E+00**	2.63935E+00	2.63935E+00	2.63935E+00
	Std	3.18000E-05	1.87000E-09	1.28000E-15	**7.28000E-16**	1.14000E-15

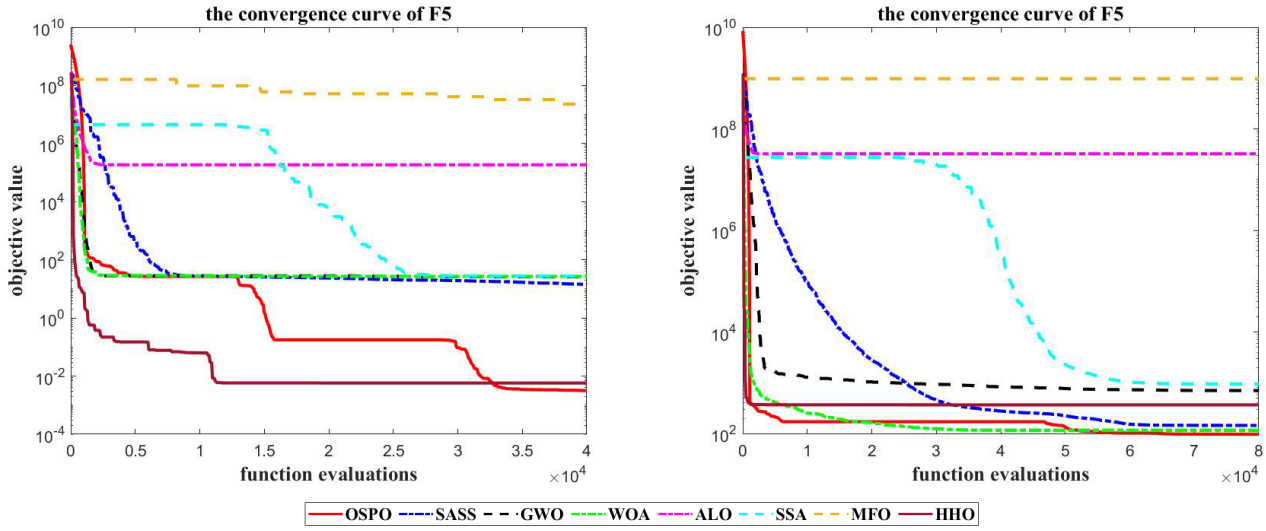

FIGURE 8. The convergence curves of F5 in 30 dim (left) and 100 dim (right).

The p-values reported in Table 13 and Table 16 also show that the superiorities of OSPO are statistically significant since the p-values are much less than 0.05.

4) COMPOSITE TEST FUNCTIONS

The simulation results of OSPO and 7 comparative algorithms on the composite test functions are shown in Table 17. Table 18 illustrates the Friedman ranks based on the best solutions of the 8 algorithms, and Table 19 illustrates the p-values of the Wilcoxon's rank-sum test.

According to the results of the algorithms on composite modal test functions in Table 17 and Table 18, it is evident that OSPO algorithm can perform well on all these test functions. OSPO can solve these problems with best or at least promising solutions. The Friedman ranks results reported in Table 18 illustrate that OSPO algorithm has the best performance when dealing with different composite test functions, and the average rank value is 1.00. The p-values reported in Table 19 also show that this superiority is statistically significant since the p-values are much less than 0.05 for most cases. The SASS algorithm takes the second place with the average rank value 2.00.

Composite benchmark functions can measure both the exploration and exploitation of the algorithms. Therefore, the results imply that OSPO can lead to a fascinating balance between exploration and exploitation when facing complicated search space. Since the composite search spaces can properly mimic the real search spaces, these results also

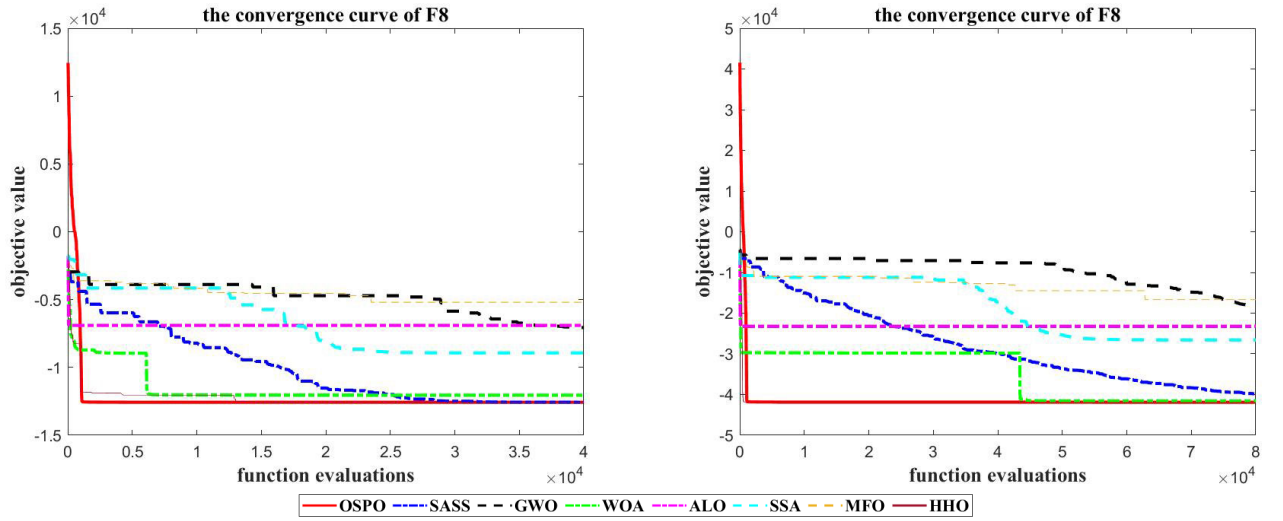

FIGURE 9. The convergence curves of F8 in 30 dim (left) and 100 dim (right).

imply OSPO has the ability to solve real-world challenging optimization problems.

5) CONVERGENCE BEHAVIOR ANALYSIS

The convergence curves of F5 (uni-modal test function example), F8 (multi-modal test function example) and F14 (composite test function example) are shown in Fig. 8, Fig. 9 and Fig. 10 respectively.

According Fig. 8, OSPO algorithm converges to the solutions better than other algorithms both in 30 (left sub-figure) and 100 (right sub-figure) dimensions. These results demonstrate that OSPO can converge to optimum in uni-modal test functions, and its exploitation ability is promising. However, as also shown in Fig. 8, the convergence speed of OSPO is not the fastest at the beginning, and this is due to the usage of receding sampling strategy in OSPO. Specifically, OSPO tends to reset the exploitation to find other attractive regions, which slows down its convergence speed when dealing with uni-modal test functions.

According to Fig. 9, OSPO algorithm converges to the global optimum successfully both in 30 (left sub-figure) and 100 (right sub-figure) dimensions. In addition, the convergence speed of OSPO is also faster than other algorithms. These results demonstrate the superiority of OSPO in solving multi-modal test functions, and its exploration ability is promising. The receding sampling strategy helps OSPO to avoid falling into local optima.

According to Fig. 10, OSPO algorithm converges to the optimal solution quickly, which demonstrate the effectiveness of OSPO in solving composite test functions. It can be stated that OSPO can balance exploration and exploitation to find best solutions in complicated problems.

The convergence curves of other test functions are reported in Fig. 11, Fig. 12 and Fig. 13 in the Appendix. These results demonstrate the promising performance of OSPO in dealing with uni-modal, multi-modal and composite test functions.

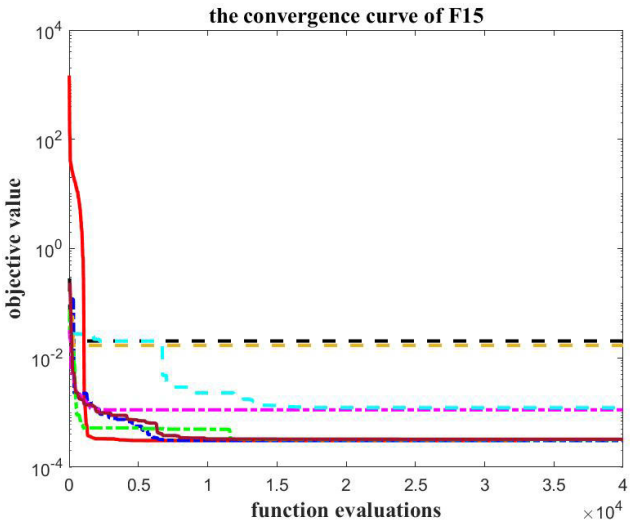

FIGURE 10. The convergence curve of F15.

B. REAL-WORLD OPTIMIZATION PROBLEMS

To further verify the effectiveness of OSPO algorithm, 21 real-world optimization problems are tested in this section. The best four algorithms of "CEC2020 Competition on Real-World Single Objective Constrained Optimization": SASS algorithm [42], sCMAgES [43], EnMODE [44] and COLSHADE [45] are used as comparative algorithms, and the SASS algorithm is the winner of this competition.

The brief properties of problems as well as the maximum number of function evaluations of each problem are reported in Table 20. For details of these problems, one can refer to [46].

Real-world problems are constrained by inequality and equality constraints, and the constraint handling technique is as important as the optimization algorithm itself. There are a number of constraint handling techniques available, such as Deb's rule [47], stochastic ranking [48], global

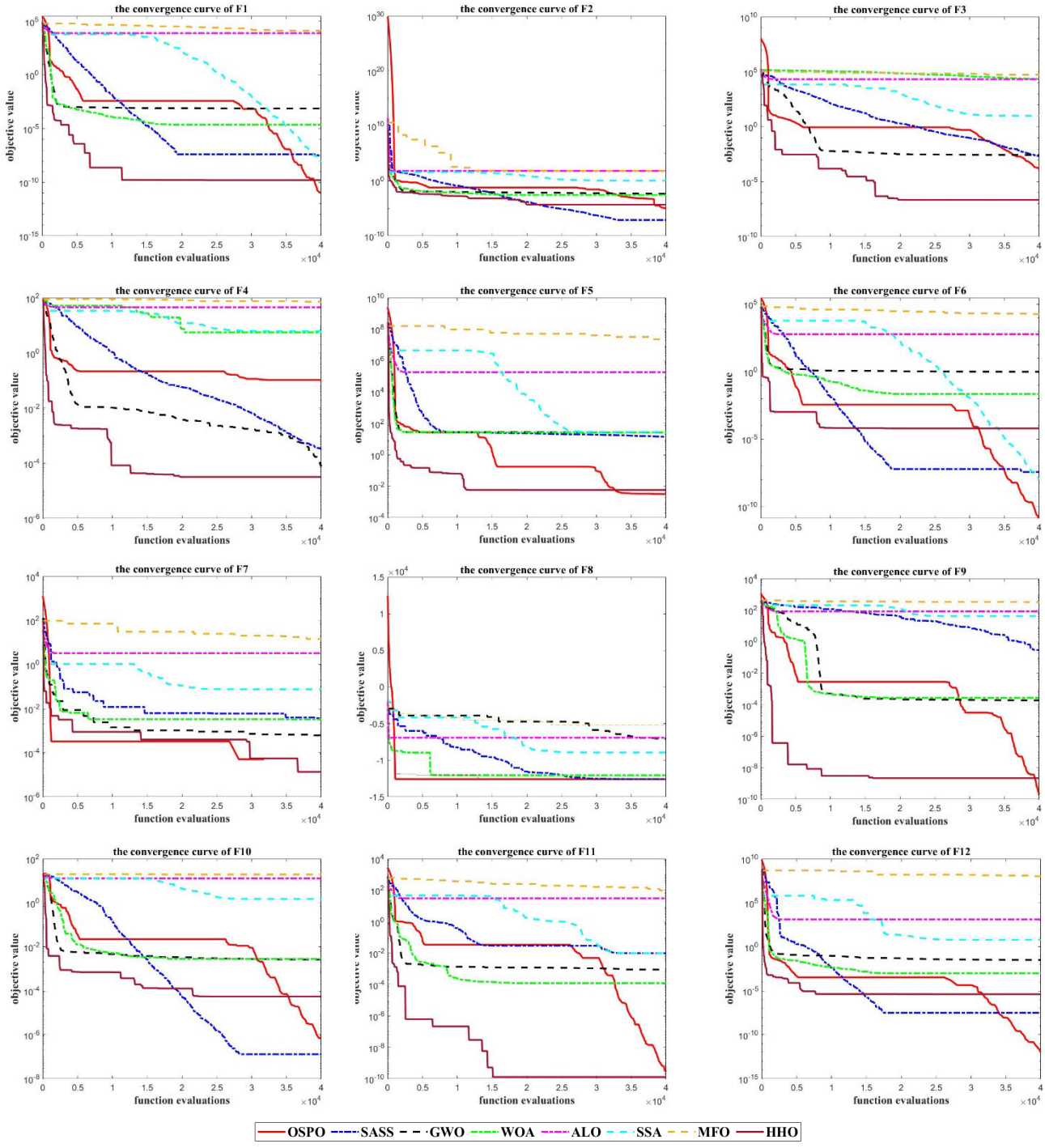

FIGURE 11. The convergence of F1-F12 (30 dimensions).

competitive ranking [49], adaptive penalty method [50], multiple ranking [51] and self-adaptive penalty strategy [52], just to name a few. The choice of constraint handling technique is out of the scope of this contribution, and here OSPO only uses self-adaptive penalty strategy as its constraint handling technique.

The parameters of OSPO is the same as that defined in Table 4, and the parameters of the four comparative algorithms are kept the same as their initial settings in literatures. Since the function evaluations of each problem is kept the same for each algorithm, these comparisons are quite fair.

The simulation results over these 21 real-world problems are reported in Table 21 and Table 22. As shown in Table 21, OSPO can solve most of these problems with promising performance, and some of solutions are even better than comparative algorithms. According to Table 22, the Friedman ranks

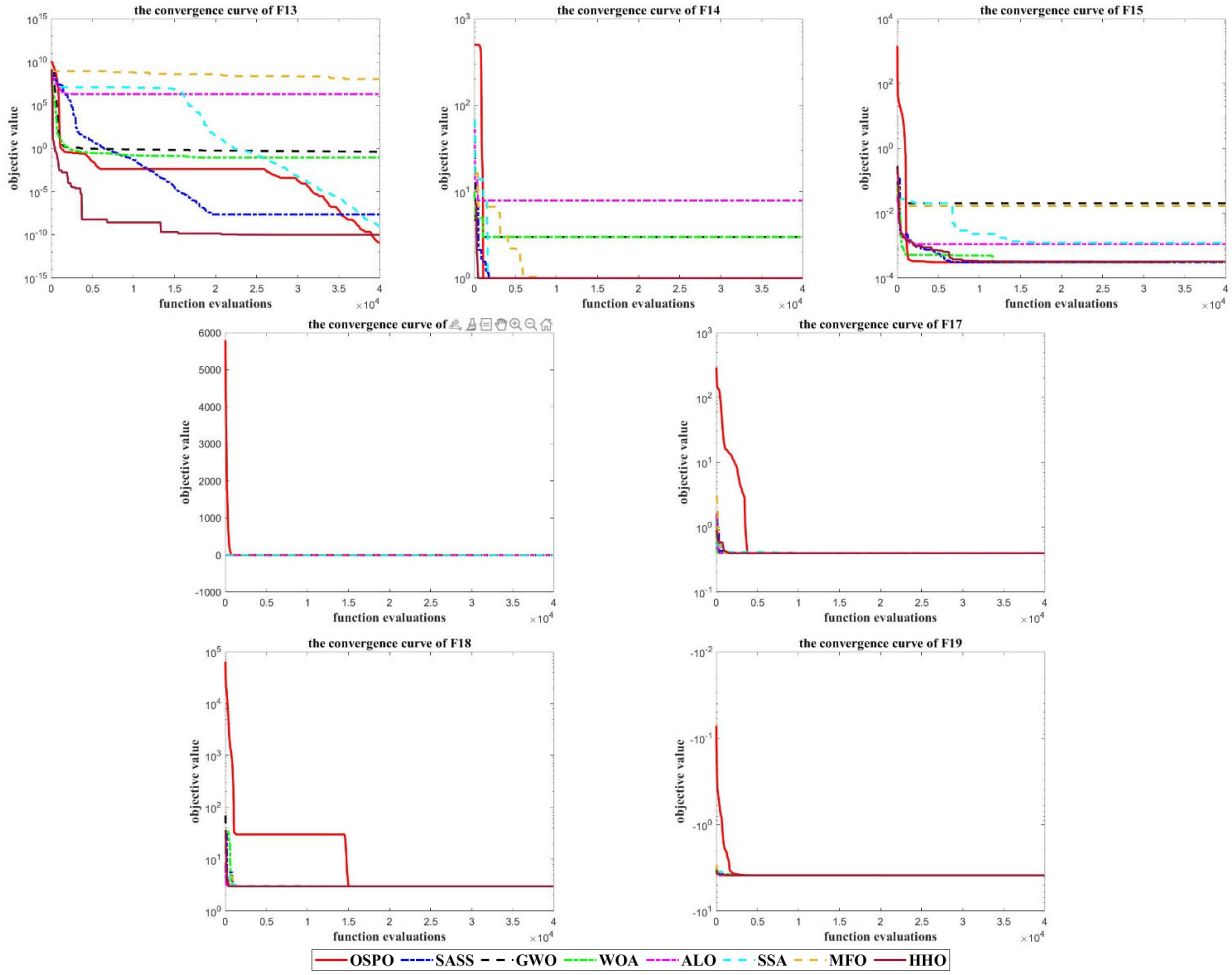

FIGURE 12. The convergence of F13-F19 (30 dimensions).

TABLE 22. Friedman ranks for the 21 real-world constrained optimization problem.

Problem	OSPO	SASS	COLSHADE	EnMODE	sCMAgES
RW_1	**1**	5	2	3	4
RW_2	5	3	1	2	4
RW_3	**1**	2	3	4	5
RW_4	**1**	2	3	4	5
RW_5	**1**	2	3	4	5
RW_6	4	1	2	3	5
RW_7	**1**	2	3	4	5
RW_8	**1**	5	3	4	2
RW_9	**1**	2	3	4	5
RW_{10}	5	1	2	3	4
RW_{11}	5	1	2	3	4
RW_{12}	**1**	2	3	4	5
RW_{13}	5	1	2	3	4
RW_{14}	5	1	2	3	4
RW_{15}	5	1	2	3	4
RW_{16}	4	1	2	3	5
RW_{17}	5	3	1	2	4
RW_{18}	5	1	2	3	4
RW_{19}	**1**	2	3	4	5
RW_{20}	**1**	2	3	4	5
RW_{21}	**1**	5	2	3	4
Average rank	2.81	**2.14**	2.33	3.33	4.38
Rank	3	1	2	4	5

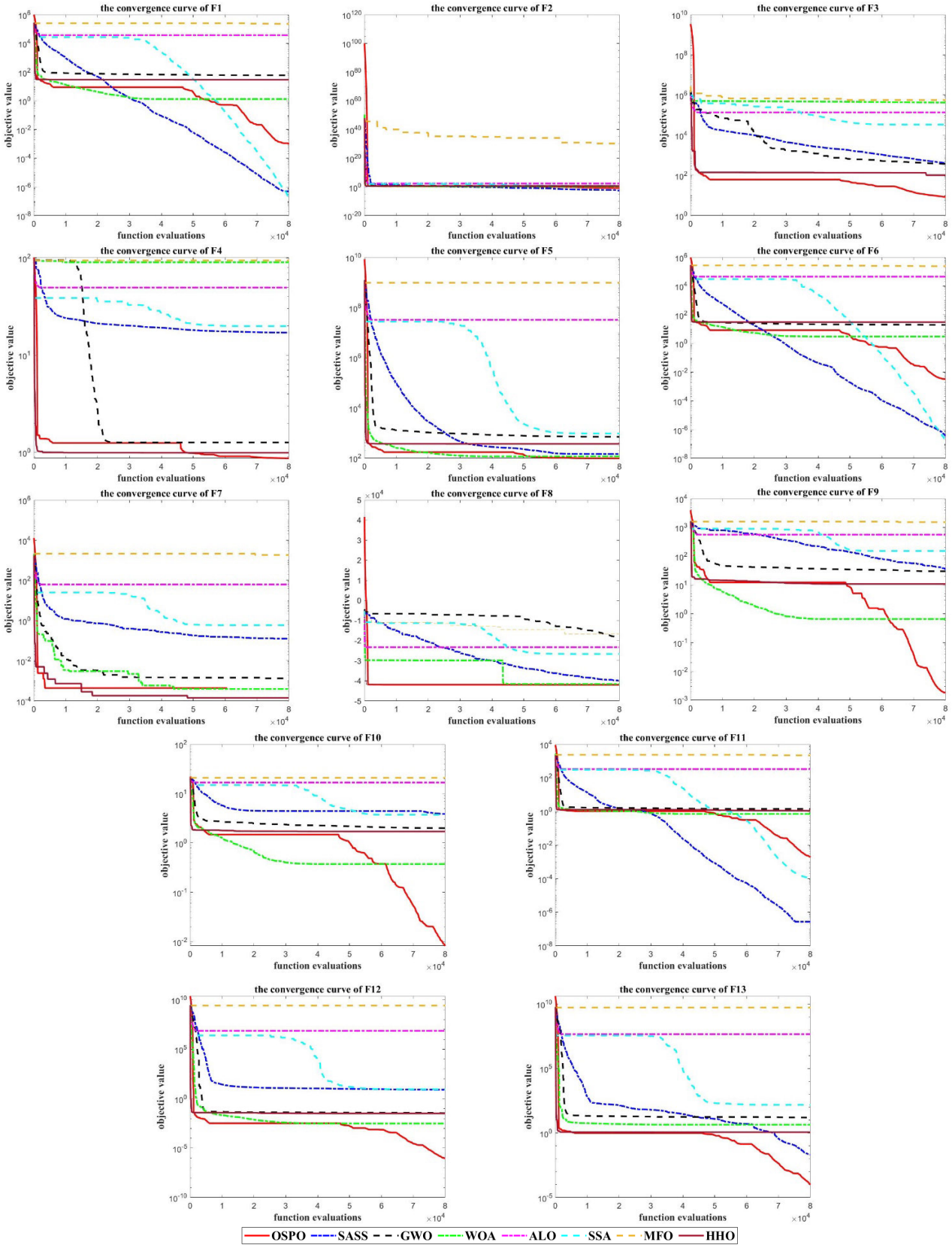

FIGURE 13. The convergence of F1-F13 (100 dimensions).

of OSPO is 2.81, and the average ranks of SASS and COLSHADE are 2.14 and 2.33 respectively. OSPO outperforms EnMODE and sCMAgES, and has competitive performance compared to SASS and COLSHADE.

In conclude, OSPO can solve different kinds of real-world problems successfully with satisfactory performances, and its overall performance is quite competitive compared to other algorithms. There are some improvements left for OSPO to solve real-world constrained optimization problems, because the constraint handling technique has not been investigated to fit with the characteristics of OSPO.

V. CONCLUSION

A novel metaheuristic algorithm with adaptive exploration-exploitation property named OSPO is proposed in this study. The OSPO algorithm regards the optimization procedure as the realization of optimizing stochastic process, and then OSPO constructs optimal sample paths to seek the global optimum of the optimization problem. SPDF and Receding Sampling Strategy are introduced accordingly to help constructing the optimal sample path, and the adaptive modification of the parameters in SPDF is a key to directly control the exploration-exploitation property online. This direct control of exploration-exploitation property gives the algorithm the ability to deal with different kinds of problems.

Different kinds of test functions with different dimensions are employed in order to benchmark the performance of the proposed algorithm. The simulation results show that OSPO can solve these problems with competitive performance compared to other metaheuristics, and the corresponding analysis demonstrates that OSPO can solve at least a vast majority of optimization problems with promising performance.

In addition, 21 real-world optimization problems are tested with four well-performed comparative algorithms, and the results illustrate that OSPO can also solve different kinds of real-world optimization problems with satisfactory performance. Besides, the constraint handling technique can be modified to further improve the performance of OSPO.

Nonlinear Model Predictive Control (NMPC) is an advanced control strategy which is popular in industry, and the receding horizon strategy used in NMPC formulation makes the optimization problem at each sample instant different from each other. Since OSPO has the potential to solve different optimization problems with promising performance, OSPO may be a suitable online solver for the NMPC formulation, and this will be investigated in our future researches.

This contribution is the initial idea of OSPO. For future studies, a dedicated constraint handling technique for OSPO can be constructed. Besides, a parallel population version of OSPO can be extended. Moreover, different types of SPDF can be developed to adaptively control the exploration-exploitation property.

APPENDIX

See Figs 11 to 13 here.

REFERENCES

[1] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997, doi: 10.1109/4235.585893.

[2] X.-S. Yang, "Nature-inspired optimization algorithms: Challenges and open problems," *J. Comput. Sci.*, vol. 46, Oct. 2020, Art. no. 101104.

[3] S. Mirjalili, "The ant lion optimizer," *Adv. Eng. Softw.*, vol. 83, pp. 80–98, May 2015.

[4] S. A. Uymaz, G. Tezel, and E. Yel, "Artificial algae algorithm (AAA) for nonlinear global optimization," *Appl. Soft Comput.*, vol. 31, pp. 153–171, Jun. 2015.

[5] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[6] V. Punnathanam and P. Kotecha, "Yin-Yang-pair optimization: A novel lightweight optimization algorithm," *Eng. Appl. Artif. Intel.*, vol. 54, pp. 62–79, Sep. 2016.

[7] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[8] J. S. Liu, Y. Huo, and Y. Li, "Preferred strategy based self-adaptive ant lion optimization algorithm," *Pattern Recognit. Artif. Intell.*, vol. 33, no. 2, pp. 121–132, Feb. 2020.

[9] O. Abedinia, N. Amjady, and A. Ghasemi, "A new metaheuristic algorithm based on shark smell optimization," *Complexity*, vol. 21, no. 5, pp. 97–116, Dec. 2016.

[10] M. Tubishat, M. Alswaitti, S. Mirjalili, M. A. Al-Garadi, M. T. Alrashdan, and T. A. Rana, "Dynamic butterfly optimization algorithm for feature selection," *IEEE Access*, vol. 8, pp. 194303–194314, 2020, doi: 10.1109/ACCESS.2020.3033757.

[11] M. Yazdani and F. Jolai, "Lion optimization algorithm (LOA): A nature-inspired metaheuristic algorithm," *J. Comput. Des. Eng.*, vol. 3, no. 1, pp. 24–36, Jan. 2016.

[12] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[13] A. Kaveh and T. Bakhshpoori, "Water evaporation optimization: A novel physically inspired optimization algorithm," *Comput. Struct.*, vol. 167, pp. 69–85, Apr. 2016.

[14] Y. Sharafi, M. A. Khanesar, and M. Teshnehlab, "COOA: Competitive optimization algorithm," *Swarm Evolut. Comput.*, vol. 30, pp. 39–63, Oct. 2016.

[15] V. Muthiah-Nakarajan and M. M. Noel, "Galactic swarm optimization: A new global optimization metaheuristic inspired by galactic motion," *Appl. Soft. Comput.*, vol. 38, pp. 771–787, Jan. 2016.

[16] H. Abedinpourshotorban, S. M. Shamsuddin, Z. Beheshti, and D. N. A. Jawawi, "Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm," *Swarm Evol. Comput.*, vol. 26, pp. 8–22, Feb. 2016.

[17] F. Fausto, E. Cuevas, A. Valdivia, and A. González, "A global optimization algorithm inspired in the behavior of selfish herds," *Biosystems*, vol. 160, pp. 39–55, Oct. 2017.

[18] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.

[19] A. Kaveh and A. Dadras, "A novel meta-heuristic optimization algorithm: Thermal exchange optimization," *Adv. Eng. Softw.*, vol. 110, pp. 69–84, Aug. 2017.

[20] M. Jain, S. Maurya, A. Rani, and V. Singh, "Owl search algorithm: A novel nature-inspired heuristic paradigm for global optimization," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1573–1582, Mar. 2018.

[21] A. Cheraghalipour, M. Hajiaghaei-Keshteli, and M. M. Paydar, "Tree growth algorithm (TGA): A novel approach for solving optimization problems," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 393–414, Jun. 2018.

[22] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," *Swarm Evol. Comput.*, vol. 44, pp. 148–175, Feb. 2019.

[23] S. Arora and S. Singh, "Butterfly optimization algorithm: A novel approach for global optimization," *Soft Comput.*, vol. 23, no. 3, pp. 715–734, 2019.

[24] F. A. Hashim, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany, and S. Mirjalili, "Henry gas solubility optimization: A novel physics-based algorithm," *Future Gener. Comput. Syst.*, vol. 101, pp. 646–667, Dec. 2019.

[25] A. Faramarzi, M. Heidarinejad, B. Stephens, and S. Mirjalili, "Equilibrium optimizer: A novel optimization algorithm," *Knowl.-Based Syst.*, vol. 191, Mar. 2020, Art. no. 105190.
[26] H. A. Alsattar, A. A. Zaidan, and B. B. Zaidan, "Novel meta-heuristic bald eagle search optimisation algorithm," *Artif. Intell. Rev.*, vol. 53, no. 3, pp. 2237–2264, Mar. 2020.
[27] Z. Wei, C. Huang, X. Wang, T. Han, and Y. Li, "Nuclear reaction optimization: A novel and powerful physics-based algorithm for global optimization," *IEEE Access*, vol. 7, pp. 66084–66109, 2019, doi: 10.1109/ACCESS.2019.2918406.
[28] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–33, Jun. 2013.
[29] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, "A better balance in metaheuristic algorithms: Does it exist?" *Swarm Evol. Comput.*, vol. 54, May 2020, Art. no. 100671.
[30] J. Hesser and R. Manner, "Towards an optimal mutation probability in genetic algorithms," in *Proc. 1st Parallel Problem Solving From Nature*. Berlin, Germany: Springer, 1991, pp. 115–124.
[31] F. G. Lobo and C. F. Lima, "Revisiting evolutionary algorithms with on-the-fly population size adjustment," in *Proc. 8th Annu. Conf. Genet. Evol. Comput. (GECCO)*, 2006, pp. 41–50.
[32] R. K. Ursem, "Diversity-guided evolutionary algorithms," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1633–1640.
[33] A. Kumar, R. K. Misra, D. Singh, S. Mishra, and S. Das, "The spherical search algorithm for bound-constrained global optimization problems," *Appl. Soft Comput.*, vol. 85, Dec. 2019, Art. no. 105734.
[34] E. De Mendonça Mesquita, R. C. Sampaio, H. V. H. Ayala, and C. H. Llanos, "Recent meta-heuristics improved by self-adaptation applied to nonlinear model-based predictive control," *IEEE Access*, vol. 8, pp. 118841–118852, 2020, doi: 10.1109/ACCESS.2020.3005318.
[35] S. Gao, Y. Gao, Y. Zhang, and L. Xu, "Multi-strategy adaptive cuckoo search algorithm," *IEEE Access*, vol. 7, pp. 137642–137655, 2019, doi: 10.1109/ACCESS.2019.2916568.
[36] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009, doi: 10.1109/TEVC.2009.2014613.
[37] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665, doi: 10.1109/CEC.2014.6900380.
[38] A. Gálvez and A. Iglesias, "New memetic self-adaptive firefly algorithm for continuous optimisation," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 5, pp. 300–317, Jan. 2016.
[39] H. Wang, X. Zhou, H. Sun, X. Yu, J. Zhao, H. Zhang, and L. Cui, "Firefly algorithm with adaptive control parameters," *Soft Comput.*, vol. 21, no. 17, pp. 5091–5102, Sep. 2017.
[40] J. Wu, R. Nan, and L. Chen, "Improved salp swarm algorithm based on weight factor and adaptive mutation," *J. Experim. Theor. Artif. Intell.*, vol. 31, no. 3, pp. 493–515, Feb. 2019.
[41] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[42] A. Kumar, S. Das, and I. Zelinka, "A self-adaptive spherical search algorithm for real-world constrained optimization problems," in *Proc. Genet. Evol. Comput. Conf. Companion (GECCO)*, Jul. 2020, pp. 13–14.
[43] A. Kumar, S. Das, and I. Zelinka, "A modified covariance matrix adaptation evolution strategy for real-world constrained optimization problems," in *Proc. Genet. Evol. Comput. Conf. Companion*, Jul. 2020, pp. 11–12.
[44] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Multi-operator differential evolution algorithm for solving real-world constrained optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185722.
[45] J. Gurrola-Ramos, A. Hernandez-Aguirre, and O. Dalmau-Cedeno, "COLSHADE for real-world single-objective constrained optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.
[46] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm Evol. Comput.*, vol. 56, Aug. 2020, Art. no. 100693.
[47] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods Appl. Mech. Eng.*, vol. 186, nos. 2–4, pp. 311–338, Jun. 2000.
[48] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 284–294, Sep. 2000, doi: 10.1109/4235.873238.
[49] R. Sarker, M. Mohammadian, X. Yao, T. Runarsson, and X. Yao, "Constrained evolutionary optimization: The penalty function approach," in *Evolutionary Optimization*. Norwell, MA, USA: Kluwer, pp. 87–113, 2002.
[50] H. J. Barbosa and A. C. Lemonge, "An adaptive penalty scheme in genetic algorithms for constrained optimization problems," *Int. J. Numer. Methods Eng.*, vol. 59, no. 5, pp. 703–736, 2004.
[51] P. Y. Ho and K. Shimizu, "Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme," *Inf. Sci.*, vol. 177, no. 14, pp. 2985–3004, Jul. 2007.
[52] B. Tessema and G. G. Yen, "A self adaptive penalty function based algorithm for constrained optimization," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 246–253, doi: 10.1109/CEC.2006.1688315.
[53] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
[54] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303–315, Mar. 2011.
[55] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, 2013.
[56] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," *Eur. J. Oper. Res.*, vol. 285, no. 2, pp. 405–428, Sep. 2020.
[57] M. Sánchez, J. M. Cruz-Duarte, J. C. Ortíz-Bayliss, H. Ceballos, H. Terashima-Marin, and I. Amaya, "A systematic review of hyper-heuristics on combinatorial optimization problems," *IEEE Access*, vol. 8, pp. 128068–128095, 2020, doi: 10.1109/ACCESS.2020.3009318.
[58] J. M. Cruz-Duarte, I. Amaya, J. C. Ortiz-Bayliss, S. E. Conant-Pablos, and H. Terashima-Marín, "A primary study on hyper-heuristics to customise metaheuristics for continuous optimisation," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185591.
[59] N. Pillay and R. Qu, *Hyper-Heuristics: Theory and Applications*. Cham, Switzerland: Springer, 2018.

JIAHONG XU received the B.S. degree from the School of Control Engineering, Northeastern University at Qinhuangdao, China, in 2016. He is currently pursuing the Ph.D. degree in control science and engineering with Tongji University, Shanghai, China. His current research interests include optimization, optimal control, and model predictive control.

LIHONG XU (Senior Member, IEEE) received the Ph.D. degree in control theory and control engineering from Southeast University, Nanjing, China, in 1991. He is currently a Full Professor with the Department of Electronics and Information Engineering, Tongji University, Shanghai, China. He is also doing a joint research work as a Visiting Professor and an Advisor of Greenhouse Research Team, BEACON, USA. His research interests include control theory, computational intelligence, and optimization theory. He was awarded the First Prize of Science and Technology Advancement Award of Ministry of Education of China, in 2005, and the Second Prize of National Science and Technology Advancement Award of China, in 2007. He is the President of IEEE CIS's Shanghai Chapter.