IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 8, 2020, accepted September 17, 2020, date of publication September 21, 2020,
date of current version October 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3025559

An Effective Adjustment to the Integration of Optimal Computing Budget Allocation for Particle Swarm Optimization in Stochastic Environments

SEON HAN CHOI[1], (Member, IEEE), AND JANG WON BAE[2], (Member, IEEE)
[1]Department of IT Convergence and Application Engineering, Pukyong National University, Busan 48513, South Korea
[2]School of Industrial Management, Korea University of Technology and Education, Cheonan 31253, South Korea

Corresponding author: Jang Won Bae (jangwon_bae@koreatech.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Korea Government (Ministry of Science and ICT) under Grant 2019R1G1A1098951.

ABSTRACT Although particle swarm optimization (PSO) is a powerful evolutionary algorithm for solving nonlinear optimization problems in deterministic environments, many practical problems have some stochastic noise. The optimal computing budget allocation (OCBA) has been integrated into PSO in various ways to cope with this. The OCBA can mitigate the effect of noise on PSO by selecting the best solution under a limited evaluation budget. Recently, with the increasing complexity of PSO applications, the evaluation costs are also increasing rapidly, which has sparked the need for more efficient PSO in stochastic environments. This article proposes a simple yet effective adjustment to the integration of OCBA to further improve the efficiency of PSO. The proposed adjustment allows OCBA to expand its search space to find the global best position more correctly such that the entire swarm can move on a better direction under stochastic noise. The experimental results on various benchmarks demonstrate the improved performance of PSO by the proposed adjustment under a limited budget compared with the latest studies. In addition, the results regarding fighters' evasion flight optimization emphasize the practical need for the proposed adjustment.

INDEX TERMS Particle swarm optimization, optimal computing budget allocation, stochastic environment, computational efficiency.

I. INTRODUCTION

Inspired by the social behavior of swarm, particle swarm optimization (PSO) [1] is a population-based evolutionary algorithm widely used for solving complex optimization problems [2] or learning artificial neural networks [3]. PSO finds a good solution through a swarm of particles that iteratively and randomly explore the search-space at their velocities. The distinctive feature of PSO is that each particle memorizes its *personal best* (Pbest) solution found so far and shares the memory with its neighborhood to find the *global best* (Gbest) solution efficiently. However, because PSO was originally designed to work in deterministic environments, it can be inefficient for many real-life optimization problems that typically involve using stochastic factors to capture the uncertainty in the real-world [4]. Unlike in a deterministic environment, fitness evaluation for a solution in a stochastic environment yields a noisy value; thus, particles may fail to distinguish a better solution, which disorientates the particles' movement significantly. So far, several strategies for mitigating the effect of noise on PSO have been proposed, and these are mainly based on a single-evaluation approach or a resampling approach.

The single-evaluation approach evaluates the fitness of a solution only once but is aimed at reducing the effect of the evaluated noisy values on PSO through many iterations of PSO. Fernandez-Marquez and Arcos [5] proposed an evaporation factor to worsen the outdated Pbest solution as PSO iterates, thereby preventing particles from premature convergence due to noise. They further introduced a dynamic factor to improve the performance of the previous version [6]. Rada-Vilela *et al.* [7] reduced the noise effect by taking the average

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

for the noisy fitness values of neighboring particles, as the neighborhood's solutions become similar as PSO iterates.

On the other hand, the resampling approach takes multiple independent identically distributed samples to increase the accuracy of fitness evaluation for a solution [8]. The larger the number of samples collected is, the more likely one is to be obtain precise fitness values; however, this process can be very time consuming. In addition, it may lower the performance of PSO by reducing the number of iterations of PSO under a limited computing budget. Thus, the resampling approach combined ranking and selection (R&S) methods, such as the indifference-zone (IZ) [9], the optimal computing budget allocation (OCBA) [10], and the uncertainty evaluation (UE) [11], into PSO for efficient sample allocations. These R&S methods intelligently allocate a limited number of samples to correctly select the best solution from a finite set of alternatives.

Mohamed *et al.* [12] combined IZ to find the best solution from particles' current solutions, whereas Bartz-Beielstein *et al.* [13] and Choi [14] applied OCBA and UE, respectively, for the same purpose. However, to find the best solution more correctly, Choi considered the previous Gbest solution for allocation. Pan *et al.* [15] integrated the hypothesis test as well as OCBA into PSO such that the Pbest solutions could be changed with statistical significance. Horng *et al.* [16] applied PSO combined with OCBA to a two-stage algorithm to solve a wafer probe testing problem in semiconductor manufacturing. Rada-Vilela *et al.* [17] used OCBA to estimate particles' Pbest solutions more accurately, rather than using the current solutions. Although these works mainly focused on finding only the Gbest solution, Zhang *et al.* [18] proposed a new OCBA-based method to correctly select the Pbest solutions, as well as the Gbest solution, which showed superior performance compared with previous works. Meanwhile, Rada-Vilela *et al.* [19] developed a hybrid method involving both single-evaluation and resampling approaches, and Taghiyeh and Xu [20] suggested a new approach using a set of statistically equivalent Gbest solutions.

Practical optimization problems to which PSO is applied typically have a stochastic simulation model as the objective function. This is because the simulation can accurately analyze modern complex industrial systems that cannot be described as a closed-form analytic model, with just a few assumptions [21]. Recently, with the increasing complexity of the systems, the cost per simulation run (i.e., single evaluation of fitness) is also increasing rapidly, which sparks the need for more efficient PSO in stochastic environments. As shown in the literature, OCBA has been applied to PSO in various ways, including the dedicated variants for PSO, due to its superior efficiency. This article proposes a simple but effective adjustment to the integration of OCBA to further improve the efficiency of PSO under stochastic noise. The proposed adjustment allows OCBA to select the Gbest solution more correctly, thereby mitigating the effect of noise and increasing the performance of PSO under a limited evaluation budget. The experimental results for various benchmark problems demonstrate improved efficiency compared with the existing works, and a case study on a practical problem emphasizes the need for the proposed adjustment.

The remainder of this article is organized as follows: Section II briefly introduces PSO and OCBA. Section III proposes the effective adjustment, and Section IV exhibits the experimental results. Finally, a conclusion is given in Section V.

II. BACKGROUND

This section briefly introduces PSO in a stochastic environment. In addition, OCBA and its integration to PSO are described.

A. PARTICLE SWARM OPTIMIZATION

Due to its simple and superior performance, various variants of PSO have been developed for various requirements, which is summarized in [22] and [23]. However, in this article, a standard version of PSO [24] is employed to provide a baseline for clarifying the efficiency improvement of the proposed adjustment.

The basic concept of PSO is that a swarm of m particles with individual movements converges to the Gbest solution via sharing their information with, one another as bees gather honey. When it comes to describing the movement, each particle i in the D-dimensional search space has its position $\boldsymbol{X}_i^l = \left[x_{i1}^l, \ldots, x_{iD}^l\right]$ that encodes a solution to the problem, and velocity $\boldsymbol{V}_i^l = \left[v_{i1}^l, \ldots, v_{iD}^l\right]$, where the superscript l represents the iteration number of PSO. In addition, each particle has a memory to store its Pbest position $\boldsymbol{P}_i^l = \left[p_{i1}^l, \ldots, p_{iD}^l\right]$ found so far. In each iteration of PSO, particles define the Gbest position $\boldsymbol{G}^l = \left[g_1^l, \ldots, g_D^l\right]$ found so far in the entire swarm by sharing the memory. Then, the velocity and position of each particle are updated based on its $\boldsymbol{P}_i^l$ and $\boldsymbol{G}^l$ such that the entire swarm can converge to $\boldsymbol{G}^l$ while maintaining the explorations of each particle on the search space. For each dimension $d \in \{1, \ldots, D\}$, the standard PSO updates the velocity and position as follows:

$$v_{id}^{l+1} = \chi\left(v_{id}^l + c_1\varepsilon_1\left(p_{id}^l - x_{id}^l\right) + c_2\varepsilon_2\left(g_{id}^l - x_{id}^l\right)\right), \quad (1)$$

$$x_{id}^{l+1} = x_{id}^l + v_{id}^{l+1}. \quad (2)$$

In (1), c_1 and c_2 are constants called acceleration coefficients to control the balance of the attraction between $\boldsymbol{P}_i^l$ and $\boldsymbol{G}^l$. ε_1 and ε_2 are independent random numbers uniformly distributed in [0,1] to add some randomness to the explorations. χ is the constrictive factor used to achieve the convergence of swarm [25]. The entire procedure of standard PSO for deterministic environments is summarized in Algorithm 1.

Meanwhile, as mentioned previously, the exact value of fitness $f\left(\boldsymbol{X}_i^l\right)$ cannot be obtained in stochastic environments, and the j th fitness evaluation for $\boldsymbol{X}_i^l$ yields only a noisy sample $\hat{f}_j\left(\boldsymbol{X}_i^l\right)$. It is assumed that $\hat{f}_j\left(\boldsymbol{X}_i^l\right)$ follows a normal distribution with unknown mean $f\left(\boldsymbol{X}_i^l\right)$ and variance σ_i^2 (i.e., $\hat{f}_j\left(\boldsymbol{X}_i^l\right) = f\left(\boldsymbol{X}_i^l\right) + \mathcal{N}\left(0, \sigma_i^2\right)$). This normality assumption is reasonable in practical cases where $f(\cdot)$ is typically

defined as a simulation model. This is because the simulation output is obtained from an average value or batch means, so the central limit theorem holds [10]. For reducing stochastic noise in the evaluation, the resampling approach takes sample mean $\bar{f}\left(\boldsymbol{X}_i^l\right) = 1/n_i \cdot \sum_{j=1}^{n_i} \hat{f}_j\left(\boldsymbol{X}_i^l\right)$, where n_i is the number of evaluations for $\boldsymbol{X}_i^l$.

Algorithm 1 Particle Swarm Optimization for Deterministic Environments

generate m particles in the D-dimensional search space with an initial position $\boldsymbol{X}_i^1 = \left[x_{i1}^1, \ldots, x_{iD}^1\right]$ and velocity $\boldsymbol{V}_i^1 = \left[v_{i1}^1, \ldots, v_{iD}^1\right]$ randomly, where the particle index $i = 1, \ldots, m$.
set the personal best position $\boldsymbol{P}_i^0 \leftarrow \boldsymbol{X}_i^1$ **for each** particle i
for $l = 1$ to l_{max} **do**
  evaluate the fitness $f\left(\boldsymbol{X}_i^l\right)$ **for each** particle i
  update $\boldsymbol{P}_i^l$ **for each** particle i:
    if $f\left(\boldsymbol{X}_i^l\right) \leq f\left(\boldsymbol{P}_i^{l-1}\right)$; then **set** $\boldsymbol{P}_i^l \leftarrow \boldsymbol{X}_i^l$; else **set** $\boldsymbol{P}_i^l \leftarrow \boldsymbol{P}_i^{l-1}$
  update the global best position $\boldsymbol{G}^l \leftarrow \text{argmin}_{\boldsymbol{P}_i^l} f\left(\boldsymbol{P}_i^l\right)$
  update $\boldsymbol{V}_i^{l+1}$ and $\boldsymbol{X}_i^{l+1}$ **for each** particle i with (1) and (2)
end
return $\boldsymbol{G}$

B. OPTIMAL COMPUTING BUDGET ALLOCATION

Given k positions $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$, increasing n_i equally for all positions is inefficient. In particular, when $f(\cdot)$ is a practical simulation model, it can be very time consuming. OCBA is aimed at finding the truly best position for which fitness is a minimum among $f(\boldsymbol{X}_1), \ldots, f(\boldsymbol{X}_k)$ by allocating simulation budget T (i.e., a limited number of fitness evaluations) to $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$ efficiently. This objective can be defined as follows:

$$\begin{aligned} &\underset{n_1^*,\ldots,n_k^*}{\arg\max} \; P\left\{\bigcap_{i=1}^{k} f(\boldsymbol{X}_b) \leq f(\boldsymbol{X}_i)\right\} \\ &\text{s.t.} \sum_{i=1}^{k} n_i^* = T \quad \text{and } n_i^* \geq 0, \\ &\quad \text{where } b = \underset{i\in\{1,\ldots,k\}}{\arg\min} \bar{f}(\boldsymbol{X}_i). \end{aligned} \tag{3}$$

That is, OCBA determines n_i^*, the optimal allocations of T for each position to maximize the probability of the correct selection of $\boldsymbol{X}_b$, $P\{\text{CS}\}$. Here, $\boldsymbol{X}_b$ is the estimated best position based on the sample mean. n_i^* can be calculated as follows [10]:

$$\begin{aligned} \frac{n_i^*}{n_j^*} &= \left[\frac{\sigma_i / \left(\bar{f}(\boldsymbol{X}_b) - \bar{f}(\boldsymbol{X}_i)\right)}{\sigma_j / \left(\bar{f}(\boldsymbol{X}_b) - \bar{f}(\boldsymbol{X}_j)\right)}\right]^2, \\ &\quad i, j \in \{1, \ldots, k\}, \quad \text{and } i \neq j \neq b, \end{aligned} \tag{4}$$

$$n_b^* = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \left(\frac{n_i^*}{\sigma_i}\right)^2}. \tag{5}$$

Here, the unknown variance σ_i^2 can be approximated by the sample variance s_i^2 in practice. These optimal allocation rules of OCBA asymptotically maximize a lower bound of $P\{\text{CS}\}$ as $T \to \infty$ (cf. [10] for the detailed mathematical derivations).

Because n_i^* is calculated based on $\bar{f}(\boldsymbol{X}_i)$ and s_i^2, OCBA uses a heuristic sequential update procedure as shown in Algorithm 2. Before the iteration is started, n_0 samples are collected for every $\boldsymbol{X}_i$ to obtain the minimum data of $\bar{f}(\boldsymbol{X}_i)$ and s_i^2. Then, until T is exhausted, additional Δ samples are allocated in a sequential manner such that n_i^* can be calculated based on more accurate data.

Algorithm 2 Optimal Computing Budget Allocation Method

collect n_0 samples **for each** $\boldsymbol{X}_i$, $i \in \{1, \ldots, k\}$
set $t \leftarrow 0$ and $n_1 = \cdots = n_k \leftarrow n_0$
calculate $\bar{f}(\boldsymbol{X}_i)$ and s_i^2 **for each** i and **set** $b \leftarrow \underset{i}{\text{argmin}}\, \bar{f}(\boldsymbol{X}_i)$
while $\sum_{i=1}^{k} n_i < T$ **do**
  set $T^{t+1} \leftarrow \sum_{i=1}^{k} n_i + \min\left(\Delta, T - \sum_{i=1}^{k} n_i\right)$
  calculate n_i^* using (4) and (5) **for each** i, where $\sum_{i=1}^{k} n_i^* = T^{t+1}$
  collect $\max\left(n_i^* - n_i, 0\right)$ samples **for each** $\boldsymbol{X}_i$
  set $n_i \leftarrow n_i + \max\left(n_i^* - n_i, 0\right)$ **for each** i
  update $\bar{f}(\boldsymbol{X}_i)$ and s_i^2 **for each** i and **set** $b \leftarrow \underset{i}{\text{argmin}}\, \bar{f}(\boldsymbol{X}_i)$
  set $t \leftarrow t + 1$
end
return $\boldsymbol{X}_b$

As mentioned before, previous studies integrated OCBA into PSO to mitigate the effect of noise by finding the best one among the particles' current positions (i.e., [13], [15], and [16]) or Pbest positions (i.e., [17]). The next section proposes a simple yet effective adjustment to the integration of OCBA to further improve the efficiency of PSO in stochastic environments.

III. PROPOSED ADJUSTMENT

As shown in (1), the movement of particles depends on their own Pbest position and Gbest position; thus, if every $\boldsymbol{P}_i^l$ and $\boldsymbol{G}^l$ is selected correctly, the effect of noise can be eliminated (i.e., PSO in stochastic environments becomes the same as that in deterministic environments). For each particle, the correct selection of $\boldsymbol{P}_i^l$ is made when the inequality between $\bar{f}\left(\boldsymbol{X}_i^l\right)$ and $\bar{f}\left(\boldsymbol{P}_i^{l-1}\right)$ is equal to that between $f\left(\boldsymbol{X}_i^l\right)$ and $f\left(\boldsymbol{P}_i^{l-1}\right)$ (i.e., $\left(f\left(\boldsymbol{X}_i^l\right) - f\left(\boldsymbol{P}_i^{l-1}\right)\right)\left(\bar{f}\left(\boldsymbol{X}_i^l\right) - \bar{f}\left(\boldsymbol{P}_i^{l-1}\right)\right) \geq 0$). Given m particles, the number of correctly selected $\boldsymbol{P}_i^l$, $N\{\text{CSP}\}$ can be defined as follows:

$$N\{\text{CSP}\} = \sum_{i=1}^{m} I(i), \quad \text{where}$$

$$I(i) = \begin{cases} 1 & \text{if } \left(f\left(\boldsymbol{X}_i^l\right) - f\left(\boldsymbol{P}_i^{l-1}\right)\right) \\ & \left(\bar{f}\left(\boldsymbol{X}_i^l\right) - \bar{f}\left(\boldsymbol{P}_i^{l-1}\right)\right) \geq 0 \\ 0 & \text{else.} \end{cases} \tag{6}$$

That is, every $\boldsymbol{P}_i^l$ is selected correctly when $N\{\text{CSP}\} = m$. After $\boldsymbol{P}_i^l$ is selected, the Gbest position is defined as the best one from the selected $\boldsymbol{P}_i^l$ as shown in Algorithm 1. Because $f\left(\boldsymbol{P}_i^l\right)$ is less than or equal to $f\left(\boldsymbol{P}_i^{l-1}\right)$ and $f\left(\boldsymbol{X}_i^l\right)$ in deterministic environments, $\boldsymbol{G}^l$, which is the best among $\boldsymbol{P}_1^l, \ldots, \boldsymbol{P}_m^l$, is the same as the best among $\boldsymbol{X}_1^l, \ldots, \boldsymbol{X}_m^l$ and $\boldsymbol{P}_1^{l-1}, \ldots, \boldsymbol{P}_m^{l-1}$ as follows:

$$\boldsymbol{G}^l = \arg\min_{\boldsymbol{P}_{\forall i}^l} f\left(\boldsymbol{P}_i^l\right) = \arg\min_{\boldsymbol{X}\in\Theta} f\left(\boldsymbol{X}\right), \quad \text{where } \Theta = \left\{\boldsymbol{X}_1^l, \ldots, \boldsymbol{X}_m^l, \boldsymbol{P}_1^{l-1}, \ldots, \boldsymbol{P}_m^{l-1}\right\}. \quad (7)$$

Thus, in stochastic environments, the correct selection of $\boldsymbol{G}^l$ is made when the selected $\boldsymbol{G}^l$ based on $\bar{f}\left(\boldsymbol{X}_i^l\right)$ and $\bar{f}\left(\boldsymbol{P}_i^{l-1}\right)$ is truly the best. The probability of the correct selection of $\boldsymbol{G}^l$, $P\{\text{CSG}\}$ can be defined as follows:

$$P\{\text{CSG}\} = P\left\{\left[\bigcap_{i=1}^{m} f\left(\boldsymbol{G}^l\right) \le f\left(\boldsymbol{P}_i^{l-1}\right)\right] \cap \left[\bigcap_{i=1}^{m} f\left(\boldsymbol{G}^l\right) \le f\left(\boldsymbol{X}_i^l\right)\right]\right\}, \text{ where}$$
$$\boldsymbol{G}^l = \arg\min_{\boldsymbol{X}\in\Theta} \bar{f}\left(\boldsymbol{X}\right) \text{ and}$$
$$\Theta = \left\{\boldsymbol{X}_1^l, \ldots, \boldsymbol{X}_m^l, \boldsymbol{P}_1^{l-1}, \ldots, \boldsymbol{P}_m^{l-1}\right\}. \quad (8)$$

When both $N\{\text{CSP}\}$ and $P\{\text{CSG}\}$ are maximized to m and 1, respectively, the effect of noise in PSO can be eliminated. However, a huge number of fitness evaluations are required to achieve this, which leads to the inefficiency of PSO in stochastic environments. If increasing $N\{\text{CSP}\}$ or $P\{\text{CSG}\}$, respectively, has a different influence on the performance of PSO, it may be more efficient to concentrate a limited evaluation budget on maximizing the dominant one of the two. To verify this strategy, numerical experiments for the nine benchmark problems have been conducted using the most basic resampling approach, PSO_ER, which equally distributes the number of fitness evaluations to $\boldsymbol{X}_1^l, \ldots, \boldsymbol{X}_m^l$. A summary of the nine problems can be found in Table 2. Here, the experimental environments were designed based on [18] because it showed the best performance among the previous studies integrated OCBA into PSO. Gaussian noise with the variance of 25 was added to each objective function to create a stochastic environment [18]. For identifying the influence of $N\{\text{CSP}\}$ and $P\{\text{CSG}\}$, several variants that select $\boldsymbol{P}_i^l$ and $\boldsymbol{G}^l$ in different ways have been suggested based on PSO_ER, which are summarized in Table 1.

As shown in Table 1, PSO_ER basically selects $\boldsymbol{P}_i^l$ and $\boldsymbol{G}^l$ based on sample mean $\bar{f}\left(\boldsymbol{X}_i^l\right)$ obtained via iterative fitness evaluations. However, the first variant selects both $\boldsymbol{P}_i^l$ and $\boldsymbol{G}^l$ correctly based on $f\left(\boldsymbol{X}_i^l\right)$ regardless of $\bar{f}\left(\boldsymbol{X}_i^l\right)$ (i.e., it is the same as the PSO in deterministic environments). For checking the influence of $N\{\text{CSP}\}$, the second variant selects only $\boldsymbol{P}_i^l$ correctly based on $f\left(\boldsymbol{X}_i^l\right)$, and $\boldsymbol{G}^l$ is selected based on $\bar{f}\left(\boldsymbol{P}_i^l\right)$ of the correctly selected $\boldsymbol{P}_i^l$. On the other hand, the third variant selects $\boldsymbol{P}_i^l$ based on $\bar{f}\left(\boldsymbol{X}_i^l\right)$ but selects $\boldsymbol{G}^l$ correctly depending on (7), regardless of the selected $\boldsymbol{P}_i^l$, to identify the influence of $P\{\text{CSG}\}$. Meanwhile, like the third variant, the fourth one also selects $\boldsymbol{P}_i^l$ based on $\bar{f}\left(\boldsymbol{X}_i^l\right)$. However, $\boldsymbol{G}^l$ is defined as the best one of the selected $\boldsymbol{P}_i^l$ (i.e., the probability of the correct selection under the selected $\boldsymbol{P}_i^l$, $P\{\text{CS}/\boldsymbol{P}_{\forall i}^l\}$ is 1); thus, the selected $\boldsymbol{G}^l$ may be incorrect depending on the selected $\boldsymbol{P}_i^l$ (i.e., $P\{\text{CSG}\} \le 1$). All of the variants, along with the original version, have the same typical settings: $m = 50$, $c_1 = c_2 = 2.05$, and $\chi = 0.72984$. The maximum number of iterations of PSO is set to 100, and the number of fitness evaluations in the lth iteration, T^l is defined as $5,000 + 100\,(l-1)$, where $l = 1, \ldots, 100$. That is, each particle in the lth iteration will be given $100+2\,(l-1)$ fitness evaluations.

TABLE 1. Variants of PSO_ER to identify the influence of N{CSP} and P{CSG}.

name	P_i^l and G^l selection ways	
PSO_ER	P_i^l: if $\bar{f}(X_i^l) \le \bar{f}(P_i^{l-1})$; $P_i^l \leftarrow X_i^l$; else $P_i^l \leftarrow P_i^{l-1}$	$N\{\text{CSP}\} \le m$
	G^l: $G^l = \text{argmin}_{P_i^l} \bar{f}(P_i^l)$	$P\{\text{CSG}\} \le 1$
Var. 1 (without noise)	P_i^l[a]: if $\boldsymbol{f}(X_i^l) \le \boldsymbol{f}(P_i^{l-1})$; $P_i^l \leftarrow X_i^l$; else $P_i^l \leftarrow P_i^{l-1}$	$\boldsymbol{N\{\text{CSP}\} = m}$
	G^l: $G^l = \text{argmin}_{P_i^l} \boldsymbol{f}(P_i^l)$	$\boldsymbol{P\{\text{CSG}\} = 1}$
Var. 2 ($N\{\text{CSP}\} = m$)	P_i^l: if $\boldsymbol{f}(X_i^l) \le \boldsymbol{f}(P_i^{l-1})$; $P_i^l \leftarrow X_i^l$; else $P_i^l \leftarrow P_i^{l-1}$	$\boldsymbol{N\{\text{CSP}\} = m}$
	G^l: $G^l = \text{argmin}_{P_i^l} \bar{f}(P_i^l)$	$P\{\text{CSG}\} \le 1$
Var. 3 ($P\{\text{CSG}\} = 1$)	P_i^l: if $\bar{f}(X_i^l) \le \bar{f}(P_i^{l-1})$; $P_i^l \leftarrow X_i^l$; else $P_i^l \leftarrow P_i^{l-1}$	$N\{\text{CSP}\} \le m$
	G^l: $G^l = \text{argmin}_{\boldsymbol{X}\in\boldsymbol{\Theta}} \boldsymbol{f}(X)$, where $\boldsymbol{\Theta} = \{\boldsymbol{X}_{\forall i}^l, \boldsymbol{P}_{\forall i}^{l-1}\}$	$\boldsymbol{P\{\text{CSG}\} = 1}$
Var. 4 ($P\{\text{CS}/P_i^l\} = 1$)	P_i^l: if $\bar{f}(X_i^l) \le \bar{f}(P_i^{l-1})$; $P_i^l \leftarrow X_i^l$; else $P_i^l \leftarrow P_i^{l-1}$	$N\{\text{CSP}\} \le m$
	G^l: $G^l = \text{argmin}_{P_i^l} \boldsymbol{f}(P_i^l)$	$\boldsymbol{P\{\text{CS}/\boldsymbol{P}_i^l\} = 1}$[b] ($P\{\text{CSG}\} \le 1$)

[a] The shaded area indicates the modified part from PSO_ER, and the bold text emphasizes the differences to the original version.

[b] Since the selected $\boldsymbol{P}_1^l, \ldots, \boldsymbol{P}_m^l$ based on $\bar{f}\left(\boldsymbol{X}_i^l\right)$ may not include the Gbest position, $P\{\text{CSG}\}$ is less than or equal to 1.

For the nine benchmark problems, the average value of $f\left(\boldsymbol{G}^l\right)$, the average value of $N\{\text{CSP}\}$, and $P\{\text{CSG}\}$ are estimated via 1,000 independently repeated experiments for each variant. The results for Levy problem are shown in Fig. 1. In addition, Fig. 2 represents that the average value of $f\left(\boldsymbol{G}^{100}\right)$ after 100 iterations of PSO is normalized between 0 and 1 based on the results of the first variant (i.e., without noise) and the original PSO_ER. As shown in Figs. 1 and 2, maximizing $N\{\text{CSP}\}$ or $P\{\text{CSG}\}$ can improve the performance of PSO in stochastic environments. However, maximizing $P\{\text{CSG}\}$ has more influence than maximizing $N\{\text{CSP}\}$ does. As shown in Figs. 1(b) and (c), although the third variant maximizing $P\{\text{CSG}\}$ had the lowest average value of $N\{\text{CSP}\}$, it yielded a good solution closest to the solution of the first variant. On the other hand, the second variant maximizing $N\{\text{CSP}\}$ had the lowest value of $P\{\text{CSG}\}$ and yielded an inferior solution compared with the third one. In addition, even though the fourth variant maximizing

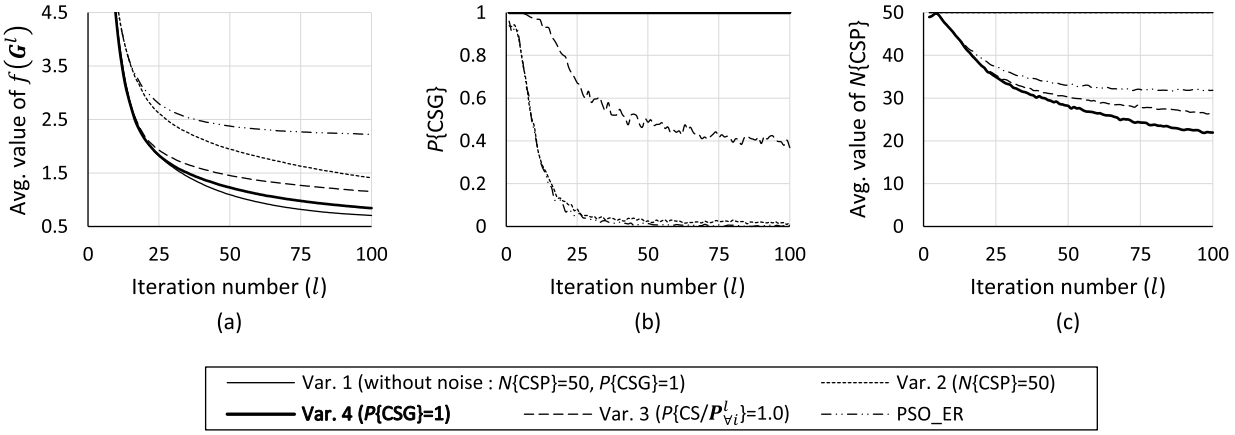

FIGURE 1. Graphs (a)-(c) represent the experimental results of the variants of PSO_ER in Table 1 for Levy problem: (a) the average value of $f\left(G^l\right)$, (b) P{CSG}, and (c) the average value of N{CSP}.

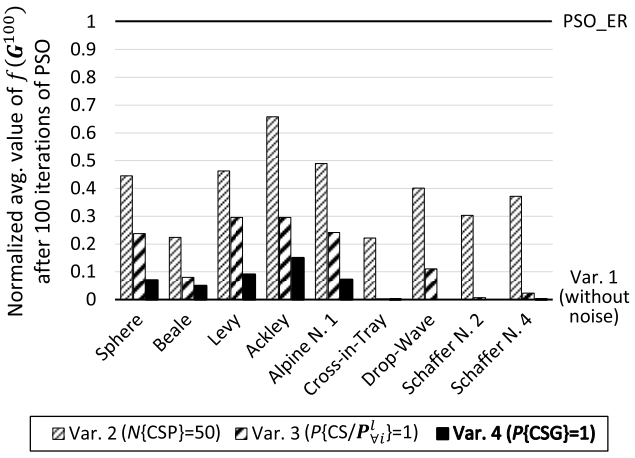

FIGURE 2. Average values of $f\left(G^{100}\right)$ of each variant of PSO_ER in Table 1 for the nine problems normalized to between 0 and 1 based on the results of the first variant (without noise) and PSO_ER. Here, the error bars were not depicted due to their tiny values after repeating 1,000 times.

$P\{\text{CS}/\boldsymbol{P}^l_{\forall i}\}$ could not achieve 1 of $P\{\text{CSG}\}$, it converged to a better solution than that of the second one. This tendency was similarly observed in the experimental results of the remaining eight benchmark problems, as shown in Fig. 2.

The reason for the greater influence of maximizing $P\{\text{CSG}\}$ can be found in fact that unlike $\boldsymbol{P}^l_i$, $\boldsymbol{G}^l$ attracts all particles. That is, the correctly selected $\boldsymbol{G}^l$ can lead the entire swarm in a better direction for convergence. In addition, as shown in Fig. 1(c), $N\{\text{CSP}\}$ had a certain value based on the most basic equal resampling method, whereas $P\{\text{CSG}\}$ was close to zero based on the second variant maximizing $N\{\text{CSP}\}$. In other words, it is expected that maximizing $P\{\text{CSG}\}$ secures a certain value of $N\{\text{CSP}\}$, which helps to further reduce the effect of noise. To sum up, the performance of PSO in stochastic environments can be improved efficiently by concentrating a limited evaluation budget on finding the Gbest position correctly.

Algorithm 3 $(l, \boldsymbol{X}_{\forall i}, \boldsymbol{P}_{\forall i}, \boldsymbol{\mathcal{N}}_{\forall i}, n_0, \Delta, T)$ Adjusted Optimal Computing Budget Allocation Method for PSO

collect n_0 samples **for each** $\boldsymbol{X}_i$, $i \in \{1, \ldots, m\}$
set $n_1 = \cdots = n_m \leftarrow n_0$ and **calculate** $\bar{f}(\boldsymbol{X}_i)$ and s_i^2 **for each** i
if $l > 1$; then **load** $\boldsymbol{\mathcal{N}}_i$ **for each** $\boldsymbol{P}_i$ to $\boldsymbol{X}_{m+i}$:
  set $\boldsymbol{X}_{m+i} \leftarrow \boldsymbol{P}_i$, $\bar{f}(\boldsymbol{X}_{m+i}) \leftarrow \boldsymbol{\mathcal{N}}_i.\bar{f}(\boldsymbol{P}_i)$, $s^2_{m+i} \leftarrow \boldsymbol{\mathcal{N}}_i.s_i^2$, and
    $n_{m+i} \leftarrow \boldsymbol{\mathcal{N}}_i.n_i$ **for each** $i \in \{1, \ldots, m\}$
  set $k \leftarrow 2m$
else; then **set** $k \leftarrow m$
set $b \leftarrow \bar{f}(\boldsymbol{X}_i)$ and $t \leftarrow 0$
while $\sum_{i=1}^k n_i < T$ **do**
  set $T^{t+1} \leftarrow \sum_{i=1}^k n_i + \min\left(\Delta, T - \sum_{i=1}^k n_i\right)$
  calculate n_i^* using (4) and (5) **for each** $i \in \{1, \ldots, k\}$, where $\sum_{i=1}^k n_i^* = T^{t+1}$
  collect $\max\left(n_i^* - n_i, 0\right)$ samples **for each** $\boldsymbol{X}_i$
  set $n_i \leftarrow n_i + \max\left(n_i^* - n_i, 0\right)$ **for each** i
  update $\bar{f}(\boldsymbol{X}_i)$ and s_i^2 **for each** i and **set** $b \leftarrow \underset{i \in \{1,\ldots,k\}}{\text{argmin}}\, \bar{f}(\boldsymbol{X}_i)$
  set $t \leftarrow t + 1$
end
save $\bar{f}(\boldsymbol{X}_i)$, s_i^2, and n_i to $\boldsymbol{\mathcal{M}}_i$ **for each** $i \in \{1, \ldots, m\}$
if $l > 1$; then **save** $\bar{f}(\boldsymbol{X}_i)$, s_i^2, and n_i to $\boldsymbol{\mathcal{N}}_i$ **for each** $i \in \{m+1, \ldots, k\}$; **else** $\mathcal{N}_{\forall i} \leftarrow \boldsymbol{\mathcal{M}}_{\forall i}$
return $[\boldsymbol{\mathcal{M}}_{\forall i}, \boldsymbol{\mathcal{N}}_{\forall i}]$

For one to find the Gbest position correctly, it is necessary to consider not only $\boldsymbol{X}^l_1, \ldots, \boldsymbol{X}^l_m$, but also $\boldsymbol{P}^{l-1}_1, \ldots, \boldsymbol{P}^{l-1}_m$ as shown in (8). However, previous studies integrated OCBA to select the best position from only $\boldsymbol{X}^l_1, \ldots, \boldsymbol{X}^l_m$ (i.e., [13], [15], and [16]) or $\boldsymbol{P}^{l-1}_1, \ldots, \boldsymbol{P}^{l-1}_m$ (i.e., [17]). The selected best position may help some with finding the correct Gbest position, but it has limitations when it comes to maximizing $P\{\text{CSG}\}$ as shown in the results of the fourth variant. Thus, this article proposes a simple adjustment that allows OCBA integrated into PSO to select the Gbest position correctly

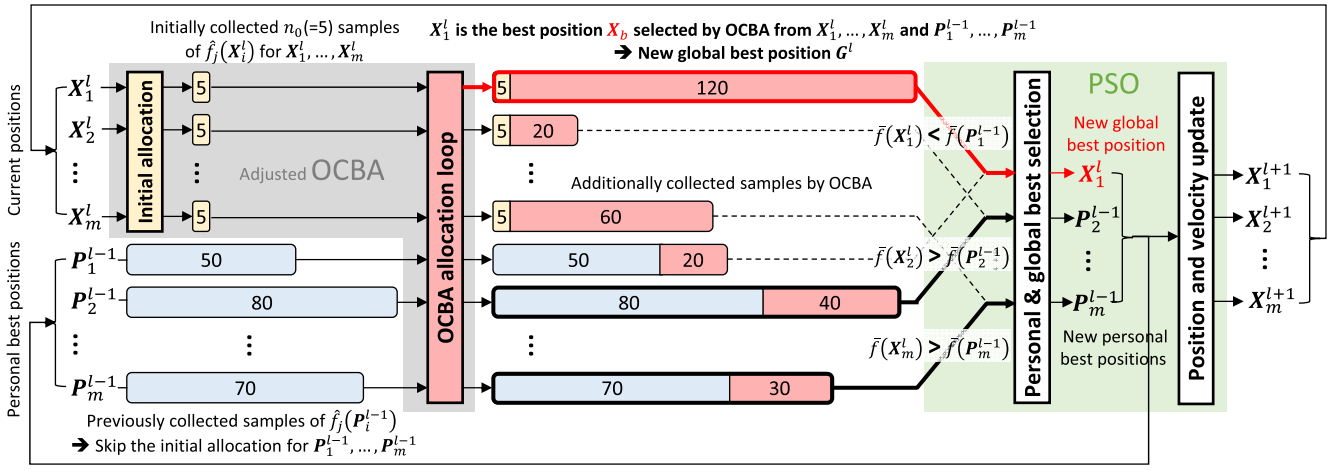

FIGURE 3. An allocation example of the OCBA method with the proposed adjustment to improve the efficiency of PSO in stochastic environments: the adjusted OCBA aims to select G^l correctly by considering both $X_1^l, \ldots, X_m^l$ and $P_1^{l-1}, \ldots, P_m^{l-1}$, where the previous evaluation data of P_i^{l-1} is reused to maximize the efficiency.

by expanding its search space. That is, OCBA allocates the given number of fitness evaluations, T, efficiently to both $X_1^l, \ldots, X_m^l$ and $P_1^{l-1}, \ldots, P_m^{l-1}$ to maximize $P\{\text{CSG}\}$, which can be defined as follows depending on (3):

$$\begin{aligned} &\underset{\forall X: n_X^* }{\arg\max}\ P\left\{\bigcap_{X\in\Theta} f\left(G^l\right) \le f\left(X\right)\right\} \\ &\text{s.t.} \sum_{X\in\Theta} n_X^* = T \text{ and } n_X^* \ge 0, \\ &\quad \text{where } G^l = \underset{X\in\Theta}{\arg\min} \bar{f}\left(X\right), \\ &\quad \Theta = \left\{X_1^l, \ldots, X_m^l, P_1^{l-1}, \ldots, P_m^{l-1}\right\}. \end{aligned} \tag{9}$$

Algorithm 2 (i.e., the original OCBA) can be applied as it is to solve the problem in (9), but for one to maximize the efficiency of PSO, some steps for reusing the previous evaluation data of P_i^{l-1} need to be added. Because every X_i^l has never been evaluated, n_0 samples are collected for each X_i^l to obtain initial data $\bar{f}\left(X_i\right)$ and s_i^2 for the following iterative allocations. On the other hand, each P_i^{l-1} already has the evaluation data obtained via OCBA in the previous iteration of PSO. Thus, the unnecessary waste of a budget can be prevented by replacing the initial data of P_i^{l-1} with these data when applying the proposed adjustment. As shown in Fig. 3, $X_1^l, \ldots, X_m^l$ each receive five samples (i.e., $n_0 = 5$) through initial allocation, whereas $P_1^{l-1}, \ldots, P_m^{l-1}$ skip the initial allocation and reuse the previously collected samples. In this way, $5m$ samples can be saved. In addition, because the number of previously collected samples is typically larger then n_0, the precise initial data of P_i^{l-1} increases the efficiency of the allocation rules of OCBA in (4) and (5).

Algorithm 3 represents the adjusted OCBA for efficiently solving the problem in (9) based on Algorithm 2, including the reuse step. It reuses the previous evaluation data of P_i^{l-1} using lines 3-5 in Algorithm 3. Then, until given budget T is exhausted, additional Δ samples are allocated sequentially depending on the allocation rules of (4) and (5) to select the Gbest position correctly from $X_1^l, \ldots, X_m^l$ and $P_1^{l-1}, \ldots, P_m^{l-1}$. The gray area in Fig. 3 illustrates Algorithm 3, and the samples allocated via Algorithm 3 are indicated by a yellow and red bar. Note that $P_1^{l-1}, \ldots, P_m^{l-1}$ do not have the yellow bar that represents the initial allocation of n_0. Algorithm 4 shows the PSO with Algorithm 3. As shown in Fig. 3, depending on the fitness evaluation

Algorithm 4 Particle Swarm Optimization With the Adjusted OCBA for Stochastic Environments

generate m particles in the D-dimensional search space with an initial position $X_i^1 = \left[x_{i1}^1, \ldots, x_{iD}^1\right]$ and velocity $V_i^1 = \left[v_{i1}^1, \ldots, v_{iD}^1\right]$ randomly, where the particle index $i = 1, \ldots, m$.
set $P_i^0 \leftarrow X_i^1$ **for each** particle i
for $l = 1$ to l_{max} **do**
  evaluate $\bar{f}\left(X_i^l\right)$ and $\bar{f}\left(P_i^{l-1}\right)$ **for each** particle i using OCBA:
    $\left[\mathcal{M}_{\forall i}^l, \mathcal{N}_{\forall i}^{l-1}\right]$ = **Algorithm 3** $\left(l, X_{\forall i}^l, P_{\forall i}^{l-1}, \mathcal{N}_{\forall i}^{l-1}, n_0, \Delta, T\right)$,
    where $\mathcal{M}_i^l = \left[\bar{f}\left(X_i^l\right), s_i^2, n_i\right]$ and $\mathcal{N}_i^l = \left[\bar{f}\left(P_i^l\right), s_i^2, n_i\right]$ are the evaluation data of X_i^l and P_i^l.
  update P_i^l **for each** particle i: **if** $\mathcal{M}_i^l.\bar{f}\left(X_i^l\right) \le \mathcal{N}_i^{l-1}.\bar{f}\left(P_i^{l-1}\right)$;
    then **set** $P_i^l \leftarrow X_i^l$ and $\mathcal{N}_i^l \leftarrow \mathcal{M}_i^l$;
    else **set** $P_i^l \leftarrow P_i^{l-1}$ and $\mathcal{N}_i^l \leftarrow \mathcal{N}_i^{l-1}$
  update $G^l \leftarrow \text{argmin}_{P_i^l}\left(\mathcal{N}_i^l.\bar{f}\left(P_i^l\right)\right)$
  update V_i^{l+1} and X_i^{l+1} **for each** particle i with (1) and (2)
end
return G

TABLE 2. Benchmark problems.

No.	Name	Objective function	D	$f(\boldsymbol{X}_{opt})$	Search space[a]
1	Sphere	$f(\boldsymbol{X}) = \sum_{i=1}^{D} x_i^2$	30	$f([0,\ldots,0]) = 0$	$[-5,5]^D$
2	Schwefel's Problem 1.2	$f(\boldsymbol{X}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	30	$f([1,2,\ldots,30]) = 0$	$[-30,30]^D$
3	Rotated HCE[b]	$f(\boldsymbol{X}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	30	$f([0,\ldots,0]) = 0$	$[-0.1,0.1]^D$
4	Beale	$f(\boldsymbol{X}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	$f([3,0.5]) = 0$	$[-4.5,4.5]^D$
5	Booth	$f(\boldsymbol{X}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	2	$f([1,3]) = 0$	$[-10,10]^D$
6	Matyas	$f(\boldsymbol{X}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	$f([0,0]) = 0$	$[-10,10]^D$
7	Levy	$f(\boldsymbol{X}) = \sin^2(\pi\omega_1) + \sum_{i=1}^{D-1}(\omega_i - 1)^2 [1 + 10\sin^2(\pi\omega_i + 1)] + (\omega_D - 1)^2[1 + \sin^2(2\pi\omega_D)]$, where $\forall i$: $\omega_i = 1 + (x_i - 1)/4$	30	$f([1,\ldots,1]) = 0$	$[-5,5]^D$
8	Rosenbrock	$f(\boldsymbol{X}) = \sum_{i=1}^{D-1}[100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2]$	30	$f([1,\ldots,1]) = 0$	$[-1,1]^D$
9	Goldstein -Price	$f(\boldsymbol{X}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$f([0,-1]) = 3$	$[-2,2]^D$
10	Camel-Six Hump	$f(\boldsymbol{X}) = 2 + (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + 4(x_2^2 - 1)x_2^2$	2	$f([0.0898,-0.7126]) = f([-0.0898,0.7126]) = 0.9684$	$[-5,5]^D$
11	Ackley	$f(\boldsymbol{X}) = -20\exp\left(-0.2\sqrt{\sum_{i=1}^{D}(x_i^2/D)}\right) - \exp(\sum_{i=1}^{D}(\cos(cx_i)/D)) + 20 - e$	30	$f([0,\ldots,0]) = 0$	$[-20,20]^D$
12	Alpine N. 1	$f(\boldsymbol{X}) = \sum_{i=1}^{D}\lvert x_i\sin x_i + 0.1x_i\rvert$	30	$f([0,\ldots,0]) = 0$	$[-5,5]^D$
13	Griewank	$f(\boldsymbol{X}) = \sum_{i=1}^{D}(x_i^2/4000) - \prod_{i=1}^{D}\cos(x_i/\sqrt{i}) + 1$	30	$f([0,\ldots,0]) = 0$	$[-100,100]^D$
14	Rastrigin	$f(\boldsymbol{X}) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)]$	30	$f([0,\ldots,0]) = 0$	$[-2,2]^D$
15	Salomon	$f(\boldsymbol{X}) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{D}x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D}x_i^2}$	30	$f([0,\ldots,0]) = 0$	$[-50,50]^D$
16	Cross-in-Tray	$f(\boldsymbol{X}) = -0.0001\left[1 + \left\lvert\sin x_1 \sin x_2 \exp\left(\left\lvert 100 - \sqrt{x_1^2 + x_2^2}/\pi\right\rvert\right)\right\rvert\right]^{0.1}$	2	$f([\pm 1.3491,\pm 1.3491]) = -2.0626$	$[-10,10]^D$
17	Drop-Wave	$f(\boldsymbol{X}) = -\left(1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)\right)/(0.5(x_1^2 + x_2^2) + 2)$	2	$f([0,0]) = -1$	$[-5.12,5.12]^D$
18	Egg Crate	$f(\boldsymbol{X}) = x_1^2 + x_2^2 + 25(\sin^2 x_1 + \sin^2 x_2)$	2	$f([0,0]) = 0$	$[-5,5]^D$
19	Happy Cat	$f(\boldsymbol{X}) = (\lVert\boldsymbol{X}\rVert^2 - D)^{1/4} + (\lVert\boldsymbol{X}\rVert^2/2 + \sum_{i=1}^{D}x_i)/2 + 0.5$	2	$f([-1,-1]) = 0$	$[-2,2]^D$
20	Levy N. 13	$f(\boldsymbol{X}) = \sin^2(3\pi x_1) + (x_1 - 1)^2(1 + \sin^2(3\pi x_2)) + (x_2 - 1)^2(1 + \sin^2(3\pi x_2))$	2	$f([1,1]) = 0$	$[-10,10]^D$
21	Schaffer N. 2	$f(\boldsymbol{X}) = 0.5 + (\sin^2(x_1^2 - x_2^2) - 0.5)/\left(1 + 0.001(x_1^2 + x_2^2)\right)^2$	2	$f([0,0]) = 0$	$[-50,50]^D$
22	Schaffer N. 4	$f(\boldsymbol{X}) = 0.5 + (\cos(\sin^2(x_1^2 - x_2^2)) - 0.5)/\left(1 + 0.001(x_1^2 + x_2^2)\right)^2$	2	$f([0,1.2531]) = 0.2926$	$[-50,50]^D$
23	Shubert	$f(\boldsymbol{X}) = \left[\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right] \times \left[\sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)\right]$	2	$f(\boldsymbol{X}_{opt}) \cong -186.7309$[c]	$[-5.12,5.12]^D$

[a] Search space is adjusted such that the "without noise" version can converge based on the parameter setting and the added Gaussian noise is meaningful compared with the fitness value.

[b] High conditioned elliptic.

[c] Subject to the search space, $\boldsymbol{X}_{opt}$ in Shubert problem can be $(-1.4251,-0.8003)$, $(-0.8003,-1.4251)$, $(-0.8003,4.8580)$, or $(4.8580,-0.8003)$.

data of $\boldsymbol{X}_1^l,\ldots,\boldsymbol{X}_m^l$ and $\boldsymbol{P}_1^{l-1},\ldots,\boldsymbol{P}_m^{l-1}$ via Algorithm 3, Pbest and Gbest positions are selected, and particles' velocities and positions are updated. Here, the evaluation data of $\boldsymbol{P}_1^l,\ldots,\boldsymbol{P}_m^l$ (i.e., newly selected Pbest positions) are stored for reuse in Algorithm 3 at the next iteration of PSO.

Meanwhile, the computational complexity of the original OCBA (i.e., Algorithm 2) is calculated as $O(k)$, were k is the number of positions in its search space [10]. Because the previous studies integrated OCBA to select the best position from only $\boldsymbol{X}_1^l,\ldots,\boldsymbol{X}_m^l$ or $\boldsymbol{P}_1^{l-1},\ldots,\boldsymbol{P}_m^{l-1}$, their complexity of the integrated OCBA is $O(m)$ (i.e., $k = m$). On the other hand, the proposed adjustment makes OCBA double its search space by considering both $\boldsymbol{X}_1^l,\ldots,\boldsymbol{X}_m^l$ and $\boldsymbol{P}_1^{l-1},\ldots,\boldsymbol{P}_m^{l-1}$ to find the Gbest position correctly. That is, k increases to $2m$ in Algorithm 3. Accordingly, the amount of computation increases, but it does not affect the complexity of Algorithm 3. From the perspective of the big-O notation, the complexity of Algorithm 3 is still $O(m)$, as in the integrated OCBA of the previous studies. In addition, this increase in computation can be negligible compared with the fitness evaluation costs in practical situations where the objective function is usually a stochastic simulation model.

IV. EXPERIMENTS

This section exhibits the experimental results on both benchmark and practical problems to demonstrate the effectiveness of the proposed adjustment.

A. BENCHMARK PROBLEMS

Algorithm 4 with the proposed adjustment was compared with the recent existing works in the resampling approach: PSO_OCBA [18], PSO_UE [14], and PSO_HYL [16]. In addition, PSO_ER and its first variant in Table 1 (i.e., without noise) were used as a control group. PSO_HYL applied OCBA as is to correctly select the best position among the particles' current positions, whereas PSO_UE used UE for the same purpose. However, compared with PSO_HYL, PSO_UE considers $\boldsymbol{G}^{l-1}$ as well as $\boldsymbol{X}_1^l,\ldots,\boldsymbol{X}_m^l$ in the allocation to find the Gbest position more correctly. PSO_OCBA combined an OCBA-based variant that focuses on finding both Pbest and Gbest positions correctly. Because PSO_OCBA showed the best performance among the competitors, the experimental environments were designed based on [18].

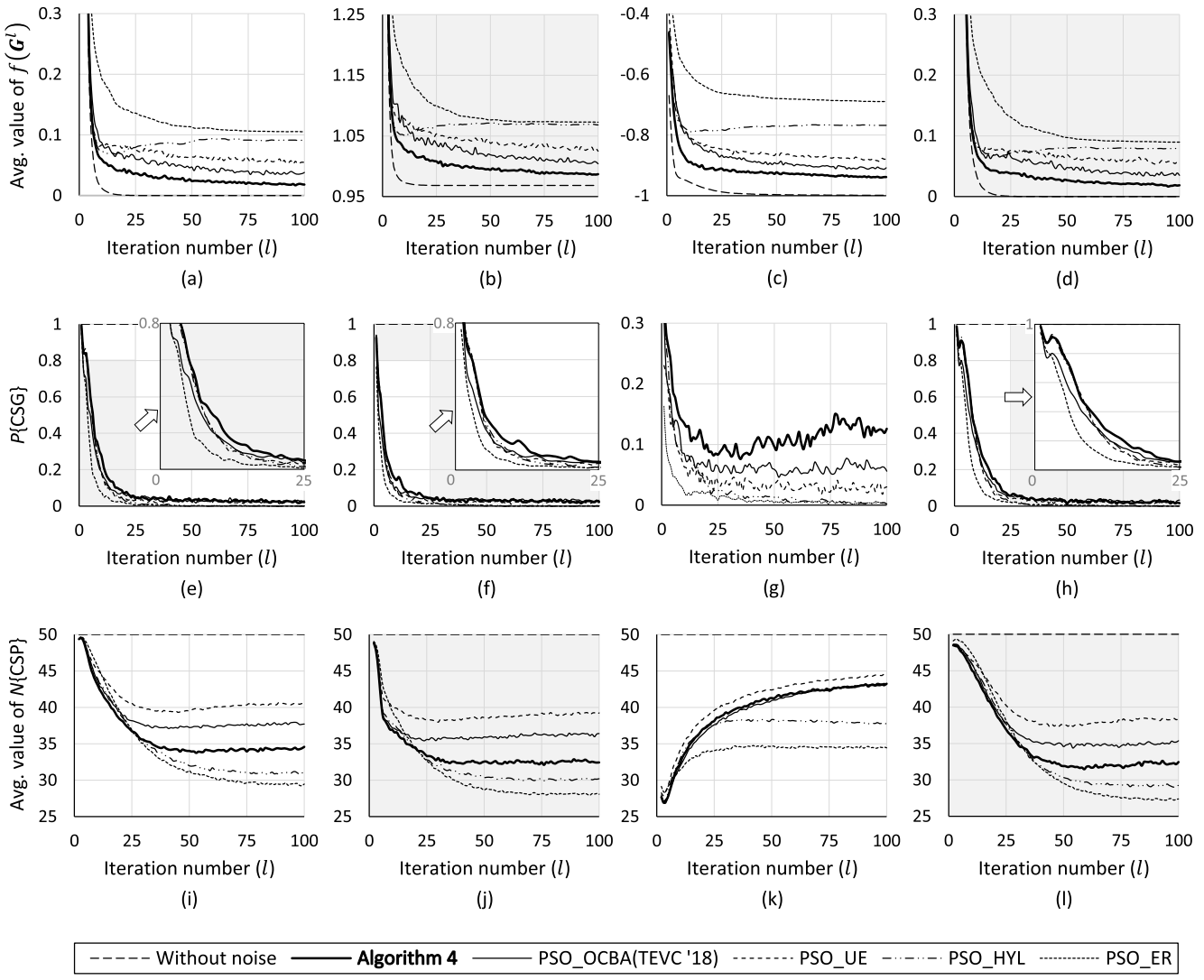

FIGURE 4. Graphs (a)-(l) represent the average value of $f\left(\boldsymbol{G}^l\right)$, P{CSG}, and the average value of N{CSP} of each competitor for the selected four problems: the first col. [(a), (e), (i)] Booth, the second col. [(b), (f), (j)] Camel-Six Hump, the third col. [(c), (g), (k)] Drop-Wave, and the last col. [(d), (h), (l)] Egg Crate problems.

Twenty-three benchmark problems were chosen from [18], [24], and [26], and Table 2 summarizes them. The objective functions in problems 1–6 are simple unimodal, whereas problems 7–10 in the shaded area have multimodal functions with few local minima. In the remaining problems 11–23, the objective functions are highly complex multimodal functions with many local minima. Gaussian noise with the variance of 25 was added to each objective function to create a stochastic environment. All competitors have the same typical settings of PSO: $m = 50$, $c_1 = c_2 = 2.05$, and $\chi = 0.72984$. In addition, the parameters for the R&S methods were set as follows: $n_0 = 10$, $\Delta = 100$, and $T^l = 5,000 + 100\,(l-1)$, where $l = 1, \ldots, l_{max}$. The average value of $f\left(\boldsymbol{G}^l\right)$, the average value of $N\{\mathrm{CSP}\}$, and $P\{\mathrm{CSG}\}$ until reaching 100 iterations of PSO (i.e., $l_{max} = 100$) were estimated via 1,000 independently repeated experiments for each competitor. The results for the four selected problems are shown in Fig. 4. Table 3 represents the average value of $f\left(\boldsymbol{G}^{100}\right)$ and the standard error of each competitor for the 23 problems, and Fig. 5 normalizes these average values in Table 3 to between 0 and 1 for easy comparison.

The experimental results clearly indicate the effectiveness of the proposed adjustment. As shown in Table 3 and Fig. 5, Algorithm 4 yielded a good solution close to the solution of the "without noise" version compared with the competitors after 100 iterations of PSO. In addition, it converged to a good solution quickly as shown in Figs. 4(a)-(d). This is because it selects the Gbest position most correctly under the limited budget. In the case of PSO_OCBA, the budget is distributed to select not only the Gbest position but also the Pbest positions, which decreases $P\{\mathrm{CSG}\}$. Because PSO_HYL and PSO_UE do not take into account $\boldsymbol{P}_1^{l-1}, \ldots, \boldsymbol{P}_m^{l-1}$, they cannot select the Gbest position correctly, like the fourth variant of PSO_ER in Table 1. On the other hand, Algorithm 4,

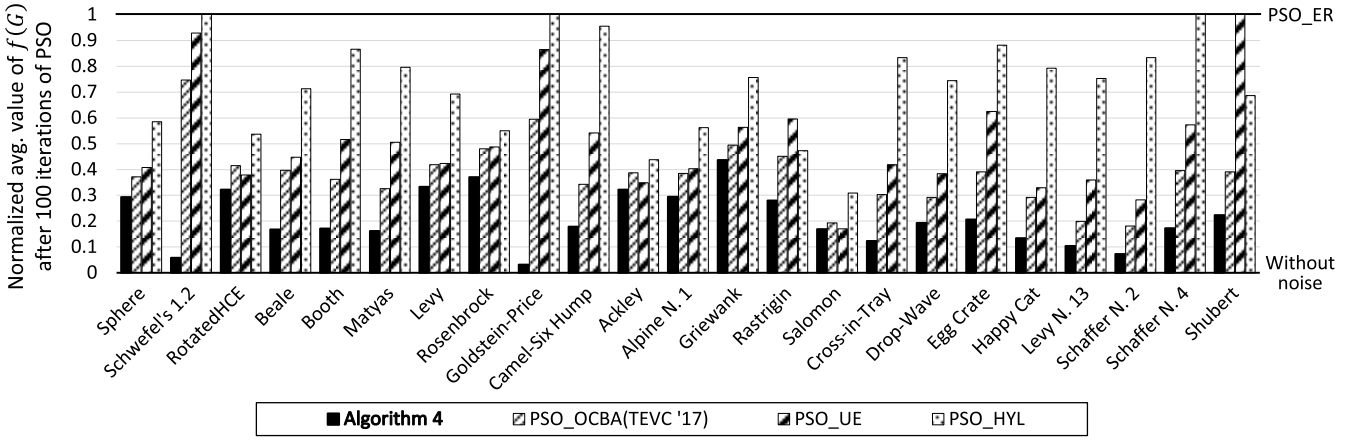

FIGURE 5. Average values of $f\left(G^{100}\right)$ of each competitor for the 23 benchmark problems normalized to between 0 and 1 based on the results of the first variant of PSO_ER (i.e., without noise) in Table 1 and the results of PSO_ER, where the error bars were not depicted due to their tiny values, as shown in Table 3.

which concentrates the budget on selecting the Gbest position correctly by applying the proposed adjustment, achieved a higher $P\{\text{CSG}\}$ than the competitors did, although $P\{\text{CSG}\}$ decreased overall due to the increasing similar positions as shown in Figs. 4(e)-(h). Especially in the early stages of the iteration of PSO, the correctly selected Gbest position via the adjusted OCBA in Algorithm 3 led the entire swarm in a better direction for convergence. In the case of $N\{\text{CSP}\}$, Algorithm 4 had a lower value than PSO_OCBA did, but its value was quite large and especially larger than PSO_HYL as shown in Figs. 4(i)-(l). As mentioned in Section III, $P\{\text{CSG}\}$ maximization of Algorithm 3 secures a certain value of $N\{\text{CSP}\}$ as well, thus further mitigating the effect of noise in Algorithm 4.

Meanwhile, Figs. 4(g) and (k) show slightly different patterns compared with the other results. That is, as the iteration of PSO increases, the average values of $P\{\text{CSG}\}$ and $N\{\text{CSP}\}$ also increase. Among 23 benchmarks, problems of 17 (illustrated in Figs. 4), 19, 21, and 22 presented such patterns. It is speculated that this is due to the characteristics of their objective functions, but more detailed analysis will be needed, which is left as future study.

If a given budget is large enough (e.g., $T^l \rightarrow \infty$), PSO_OCBA, which finds both Pbest and Gbest positions correctly, may outperform Algorithm 4. However, the proposed adjustment is effective in terms of efficiency to improve the performance of PSO under a limited evaluation budget. This efficiency is necessary for practical optimization problems of which the objective function is a stochastic simulation model. To demonstrate the necessity of the proposed adjustment, Algorithm 4 was applied to optimize the fighter's evasion flight.

B. PRACTICAL PROBLEM

The evasion flight of a fighter against an approaching missile consists of descending and soaring steps. When a fighter detects a missile approaching from 20,000 m away, the fighter descends in maintaining a constant angle, called the leading angle. Then, the missile also descends along with the fighter. When the distance between the fighter and missile is under a certain value, called the soaring distance, the fighter radically soars upward using the maximum G-factor. Because missiles are usually several times faster than fighters are, the missile cannot catch up with the fighter's movement and finally overshoots (i.e., the fighter survives). However, if the leading angle is too small, or if the soaring distance is too short or too long, the missile can continue to chase the fighter without overshooting and shoot it down.

Dominating the air is one of the most important factors in winning a modern battlefield, so it is necessary to determine the optimal values of the leading angle and the soaring distance to maximize a fighter's survival rate. Leading angle l ranges from 0 to 90 degrees, and soaring distance s ranges from 1,000 m to 15,000 m. Given $\boldsymbol{X} = [l, s]$, a combination of the two variables, the survival rate of the fighter can be evaluated using the six degrees of freedom (6 DoF) aircraft simulation model [27], denoted by $f(\boldsymbol{X})$. Fig. 6 illustrates the evasion flight of a fighter simulated by the model. Because the model has many random variables for describing uncertainty in the complex real world, it returns the fighter's survival:$f(\boldsymbol{X}) = 1$ or destruction:$f(\boldsymbol{X}) = 0$ (i.e., $f(\boldsymbol{X})$ follows Bernoulli distribution) per simulation run of $\boldsymbol{X}$. That is, the survival rate for $\boldsymbol{X}$ is defined as the expected value of $f(\boldsymbol{X})$, $E[f(\boldsymbol{X})]$. To sum up, this optimization problem can be defined as follows:

$$\begin{aligned} \boldsymbol{X}_{opt} &= \arg\max_{\boldsymbol{X}\in\Theta} E[f(\boldsymbol{X})], \text{ where} \\ \Theta &= \{\boldsymbol{X} = [l, s] \,|\, l \in [0, 90] \quad \text{and } s = [1000, 15000]\}. \end{aligned} \tag{10}$$

To solve the problem of (4), Algorithm 4 applied with the proposed adjustment was used. In addition, the other competitors in the benchmark problems were applied together to check the improved efficiency of PSO via the proposed

TABLE 3. Average value of $f\left(\boldsymbol{G}^{100}\right)$ and standard error for 23 benchmarks in Table 2.

Problems	Without noise	**Algorithm 4**	PSO_OCBA (TEVC '17)	PSO_UE	PSO_HYL	PSO_ER
Sphere	0.0359 (±0.0)[a]	**0.3680** (±0.0001)	0.4571 (±0.0001)	0.4970 (±0.0002)	0.6984 (±0.0004)	1.1679 (±0.0004)
Schwefel's Prob. 1.2	1390.2 (±1.1137)	**1395.3** (±0.9633)	1455.4 (±1.0964)	1471.3 (±1.0681)	1490.2 (±1.0891)	1457.5 (±1.1168)
Rotated-HCE[b]	0.1932 (±0.0002)	**0.6002** (±0.0003)	0.7182 (±0.0004)	0.6717 (±0.0003)	0.8715 (±0.0004)	1.4576 (±0.0009)
Beale	0.0389 (±0.0002)	**0.0593** (±0.0002)	0.0873 (±0.0002)	0.0935 (±0.0002)	0.1259 (±0.0002)	0.1609 (±0.0003)
Booth	0.0 (±0.0)	**0.0181** (±0.0)	0.0381 (±0.0001)	0.0544 (±0.0001)	0.0913 (±0.0002)	0.1503 (±0.0001)
Matyas	0.0 (±0.0)	**0.0192** (±0.0)	0.0387 (±0.0001)	0.0602 (±0.0001)	0.0947 (±0.0001)	0.1189 (±0.0001)
Levy	0.7071 (±0.0007)	**1.2125** (±0.0008)	1.3417 (±0.0007)	1.3499 (±0.0007)	1.7583 (±0.0007)	2.2238 (±0.0008)
Rosen-brock	28.1537 (±0.0007)	**28.6804** (±0.0007)	28.8363 (±0.0007)	28.846 (±0.0007)	28.9357 (±0.0007)	29.5741 (±0.0008)
Goldstein -Price	3.027 (±0.0009)	**3.0184** (±0.0)	3.0624 (±0.0009)	3.0835 (±0.0009)	3.1050 (±0.0009)	3.0941 (±0.0001)
Camel-Six Hump	0.9684 (±0.0)	**0.9871** (±0.0)	1.0041 (±0.0001)	1.0249 (±0.0001)	1.0681 (±0.0002)	1.0728 (±0.0001)
Ackley	1.8803 (±0.0005)	**2.8489** (±0.0005)	3.0474 (±0.0004)	2.9294 (±0.0004)	3.1996 (±0.0008)	4.8886 (±0.0010)
Alpine N. 1	0.0904 (±0.0001)	**0.4340** (±0.0001)	0.5403 (±0.0002)	0.5608 (±0.0002)	0.7481 (±0.0004)	1.2592 (±0.0004)
Griewank	0.4942 (±0.0002)	**1.2420** (±0.0001)	1.3406 (±0.0001)	1.4571 (±0.0001)	1.7872 (±0.0004)	2.2043 (±0.0004)
Rastrigin	17.0349 (±0.0054)	**17.3203** (±0.0052)	17.4951 (±0.0052)	17.6437 (±0.0052)	17.5174 (±0.0051)	18.0554 (±0.005)
Salomon	0.9523 (±0.0001)	**1.3372** (±0.0002)	1.3941 (±0.0002)	1.3421 (±0.0002)	1.6585 (±0.0005)	3.2346 (±0.0008)
Cross-in-Tray	-2.0626 (±0.0)	**-2.0430** (±0.0)	-2.0146 (±0.0001)	-1.9963 (±0.0001)	-1.9303 (±0.0002)	-1.0940 (±0.0001)
Drop-Wave	-0.9983 (±0.0)	**-0.9386** (±0.0)	-0.9082 (±0.0001)	-0.8798 (±0.0001)	-0.7683 (±0.0002)	-0.6894 (±0.0002)
Egg Crate	0.0 (±0.0)	**0.0185** (±0.0)	0.0350 (±0.0001)	0.0560 (±0.0001)	0.0791 (±0.0002)	0.0896 (±0.0001)
Happy Cat	0.0 (±0.0)	**0.0169** (±0.0)	0.0368 (±0.0001)	0.0415 (±0.0)	0.1000 (±0.0002)	0.1262 (±0.0001)
Levy N. 13	0.0 (±0.0)	**0.0170** (±0.0)	0.0324 (±0.0001)	0.0586 (±0.0001)	0.1226 (±0.0002)	0.1628 (±0.0002)
Schaffer N. 2	0.0 (±0.0)	**0.0216** (±0.0)	0.0542 (±0.0001)	0.0843 (±0.0001)	0.2494 (±0.0002)	0.2991 (±0.0002)
Schaffer N. 4	0.2926 (±0.0)	**0.3229** (±0.0)	0.3622 (±0.0001)	0.3935 (±0.0001)	0.4735 (±0.0001)	0.4687 (±0.0001)
Shubert	-186.7309 (±0.0)	**-186.7131** (±0.0)	-186.7000 (±0.0001)	-186.5740 (±0.0001)	-186.6760 (±0.0001)	-186.6510 (±0.0001)

[a] The value in parentheses is the standard error of the average of $f(\boldsymbol{G}^{100})$, where the value of 0.0 is not actually zero, but is a very small value closed to zero.

[b] High Conditioned Elliptic.

adjustment. All parameters except for T^l were set the same as those in the benchmarks, and T^l was set to $800+200\,(l-1)$. Unlike previous benchmark problems, the 6 DoF aircraft simulation model has high evaluation cost; thus, each competitor was applied only 10 times independently to solve the problem of (4), and 10 solutions were obtained per each. For each obtained solution $\boldsymbol{G}^{100}$, $E\left[f\left(\boldsymbol{G}^{100}\right)\right]$ was precisely estimated with many independently replicated simulations. Table 4 represents the average value and the standard error of 10 precisely estimated $E\left[f\left(\boldsymbol{G}^{100}\right)\right]$s for each competitor.

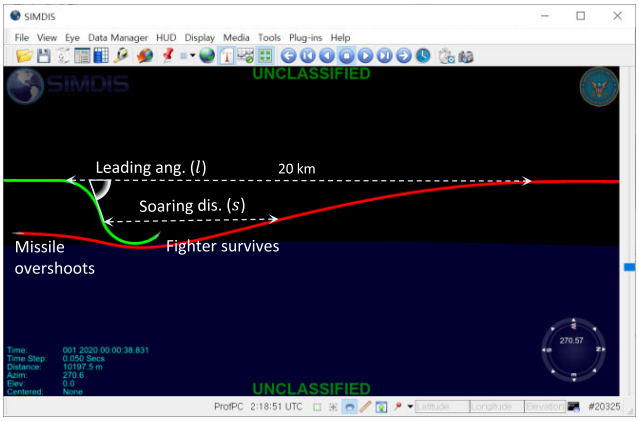

FIGURE 6. A screenshot of simulating a successful evasion flight using the 6 DoF aircraft simulation model [27].

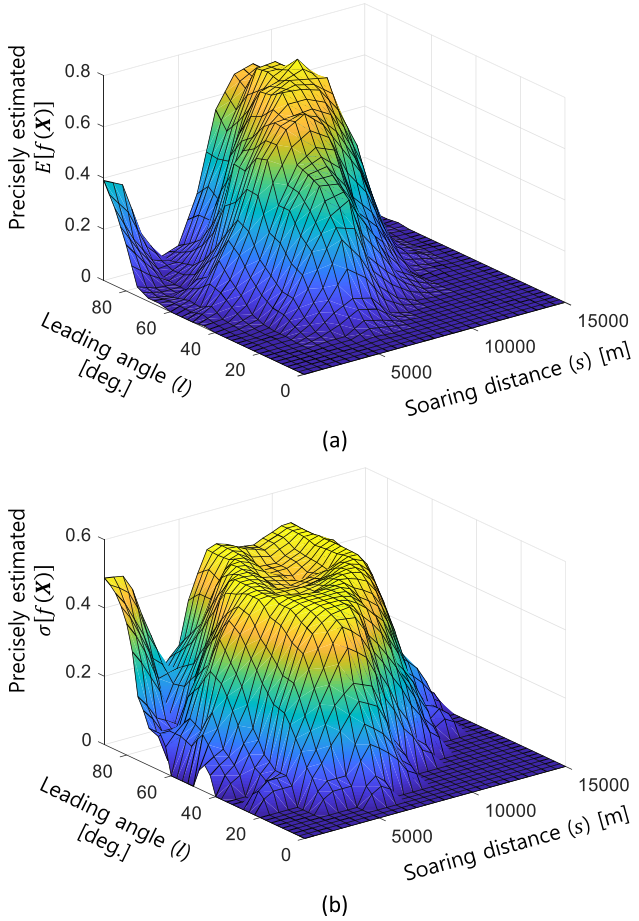

FIGURE 7. Graphs (a) and (b) represent precisely estimated $E\,[f\,(\boldsymbol{X})]$ and $\sigma\,[f\,(\boldsymbol{X})]$ with many independently replicated simulations of the 6 DoF aircraft model for finely discretized $\boldsymbol{X}$ within the search space.

Among 10 solutions of Algorithm 4, the representative solution of which the $E\left[f\left(\boldsymbol{G}^{100}\right)\right]$ is closest to the average value (i.e., 0.7831) in Table 4 consists of 70.15 degrees and 8,990 m (i.e., $\boldsymbol{G}^{100} = [70.15, 8990]$). When designing the evasion flight with this solution, fighters can achieve a survival rate of about 3.6% higher than the solutions of

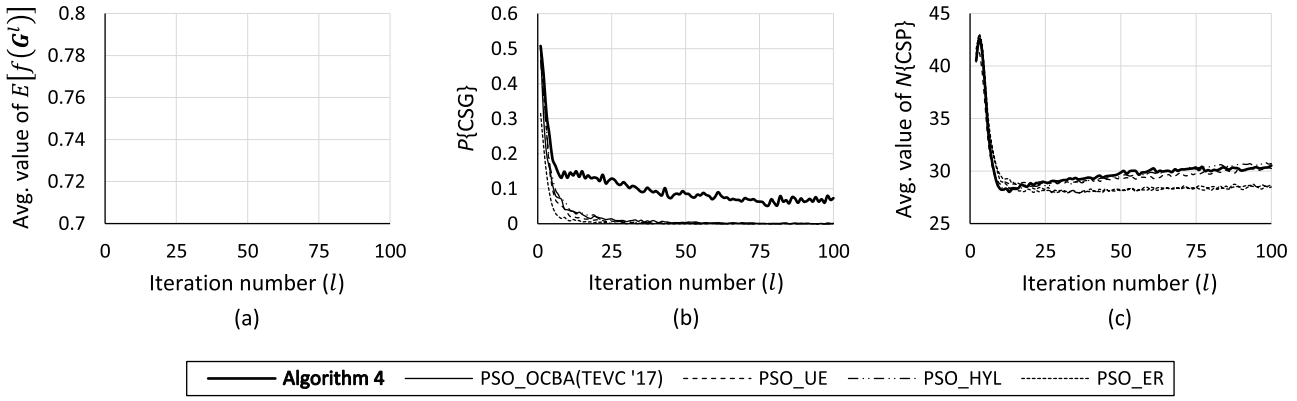

FIGURE 8. Graphs (a)-(c) represent the experimental results of the competitors for the flight evasion optimization problems based on the results of $E[f(X)]$ and $\sigma[f(X)]$ in Fig. 7: (a) the average value of $E\left[f\left(G^l\right)\right]$, (b) P{CSG}, and (c) the average value of N{CSP}.

TABLE 4. Average value of the precisely estimated $E\left[f\left(G^{100}\right)\right]$ and standard error for the evasion flight optimization problem.

	Average value of $E[f(G^{100})]$	Standard error
Algorithm 4	**0.7831**	**0.0011**
PSO_OCBA(TEVC `17)	0.7463	0.0059
PSO_UE	0.7451	0.0086
PSO_HYL	0.7455	0.0055
PSO_ER	0.7213	0.0095

PSO_OCBA, PSO_UE, and PSO_ER, and about 6.1% higher than the solution of PSO_ER as shown in Table 4. Considering that it is crucial to increase the fighter's survival rate even if the increment is just 1%, this is a huge improvement. If a greater evaluation budget is given (i.e., T^l increases), the other competitors may also find a better solution like Algorithm 4. However, as shown in Fig. 6, the cost per run of the 6 DoF simulation model is expensive and time-consuming, which may lead to prohibitively high computational costs for the optimization.

The improved efficiency of Algorithm 4 is attributed to the fact that Algorithm 3 with the proposed adjustment can select the Gbest position correctly. Fig. 7 represents the precisely estimated values of $E[f(X)]$ and $\sigma[f(X)]$ with many independently replicated simulations of the 6 DoF model for finely discretized X within the search space. As shown in Fig. 7, although the absolute value of $\sigma[f(X)]$ is small, it is relatively large compared with the small difference of $E[f(X)]$ between neighbors, which acts as a large noise effect. Based on the results in Fig. 7, the average value of $E\left[f\left(G^l\right)\right]$, the average value of N{CSP}, and P{CSG} until reaching 100 iterations of PSO were estimated through 100 independently repeated experiments for each competitor. As shown in the results in Fig. 8, Algorithm 4 achieved a higher P{CSG} than the competitors; the correctly selected Gbest position could mitigate the large noise effect and lead the entire swarm to a better solution.

Like this practical problem, the proposed adjustment is effective for solving simulation-based optimization problems that have large amounts of stochastic noise and high evaluation costs using PSO. In particular, simulation models, such as the digital twins, which are in the spotlight in the Fourth Industrial Revolution era, are continuously updated to maintain synchronization with the corresponding real systems [28]. In light of this, optimal control settings should be found quickly based on the updated models to maximize the performance of the real systems. In this case, the proposed adjustment is essential for the use of PSO.

V. CONCLUSION

This article proposed a simple yet effective adjustment to the integration of OCBA to further improve the efficiency of PSO in stochastic environments. The proposed adjustment causes OCBA to correctly select the Gbest position that has a significant influence on the performance of PSO compared with the correctly found Pbest positions. Although the previous studies select the best one from only the particles' current positions or their previous Pbest positions using OCBA, the proposed adjustment extends the search space of OCBA to consider both the current positions and the previous Pbest positions. Thereby, OCBA can correctly select the Gbest position that is defined as the best position among the extended space. In addition, the proposed adjustment prevents OCBA from wasting the evaluation budget by reusing the existing evaluation data of the previous Pbest positions. The results of comparative experiments with the latest studies on various benchmark problems demonstrated the improved efficiency of PSO due to the proposed adjustment. The PSO applied with the proposed adjustment could find the Gbest position correctly compared with the competitors, and this exact Gbest position led the entire swarm in a better direction; thus, the adjusted PSO yielded a superior solution within a limited evaluation budget. In particular, the results of the evasion flight optimization problem demonstrated the need for the proposed adjustment when using PSO to solve practical optimization problems that typically have large amounts of stochastic noise and high evaluation costs. The proposed adjustment is expected to enable PSO to be used more

effectively in the Fourth Industrial Revolution era, which requires fast and accurate optimization based on the digital twins.

ACKNOWLEDGMENT

The authors would like to thank the Editor, an Associate Editor, and the anonymous reviewers for their valuable comments and constructive suggestions.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov./Dec. 1995, pp. 1942–1948.
[2] M. Elbes, S. Alzubi, T. Kanan, A. Al-Fuqaha, and B. Hawashin, "A survey on particle swarm optimization with emphasis on engineering and network applications," *Evol. Intell.*, vol. 12, no. 2, pp. 113–129, Feb. 2019.
[3] Y. Xue, T. Tang, and A. X. Liu, "Large-scale feedforward neural network optimization by a self-adaptive strategy and parameter based particle swarm optimization," *IEEE Access*, vol. 7, pp. 52473–52483, 2019.
[4] A. Ghosh, S. Das, R. Mallipeddi, A. K. Das, and S. S. Dash, "A modified differential evolution with distance-based selection for continuous optimization in presence of noise," *IEEE Access*, vol. 5, pp. 26944–26964, 2017.
[5] J. L. Fernandez-Marquez and J. L. Arcos, "An evaporation mechanism for dynamic and noisy multimodal optimization," in *Proc. 11th Annu. Conf. Genetic Evol. Comput. (GECCO)*, Montreal, QC, Canada, 2009, pp. 17–24.
[6] J. L. Fernandez-Marquez and J. L. Arcos, "Adapting particle swarm optimization in dynamic and noisy environments," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, Jul. 2010, pp. 1–8.
[7] J. Rada-Vilela, M. Johnston, and M. Zhang, "Population statistics for particle swarm optimization: Single-evaluation methods in noisy optimization problems," *Soft Comput.*, vol. 19, no. 9, pp. 2691–2716, Sep. 2015.
[8] J. Rada-Vilela, M. Johnston, and M. Zhang, "Population statistics for particle swarm optimization: Resampling methods in noisy optimization problems," *Swarm Evol. Comput.*, vol. 17, pp. 37–59, Aug. 2014.
[9] S.-H. Kim and B. L. Nelson, "A fully sequential procedure for indifference-zone selection in simulation," *ACM Trans. Model. Comput. Simul.*, vol. 11, no. 3, pp. 251–273, Jul. 2001.
[10] C.-H. Chen and L. H. Lee, *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*, vol. 1. Singapore: World Scientific, 2011.
[11] S. H. Choi and T. G. Kim, "Efficient ranking and selection for stochastic simulation model based on hypothesis test," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 9, pp. 1555–1565, Sep. 2018.
[12] A. W. Mohamed, H. Zaher, and M. Korshid, "A particle swarm approach for solving stochastic optimization problems," *Appl. Math. Inf. Sci.*, vol. 5, no. 3, pp. 379–401, 2011.
[13] T. Bartz-Beielstein, D. Blum, and J. Branke, "Particle swarm optimization and sequential sampling in noisy environments," in *Metaheuristics* (Operations Research/Computer Science Interfaces Series), vol. 39. New York, NY, USA: Springer, 2007, pp. 261–273.
[14] S. H. Choi, "Incorporation of the uncertainty evaluation procedure into particle swarm optimization in a noisy environment," in *Proc. Winter Simulation Conf.*, National Harbor, MD, USA, Dec. 2019, pp. 2918–2919.
[15] H. Pan, L. Wang, and B. Liu, "Particle swarm optimization for function optimization in noisy environment," *Appl. Math. Comput.*, vol. 181, no. 2, pp. 908–919, Oct. 2006.
[16] S.-C. Horng, F.-Y. Yang, and S.-S. Lin, "Applying PSO and OCBA to minimize the overkills and re-probes in wafer probe testing," *IEEE Trans. Semicond. Manuf.*, vol. 25, no. 3, pp. 531–540, Aug. 2012.
[17] J. Rada-Vilela, M. Zhang, and M. Johnston, "Optimal computing budget allocation in particle swarm optimization," in *Proc. 15th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, Amsterdam, The Netherlands, 2013, pp. 81–88.
[18] S. Zhang, J. Xu, L. H. Lee, E. P. Chew, W. P. Wong, and C.-H. Chen, "Optimal computing budget allocation for particle swarm optimization in stochastic optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 206–219, Apr. 2017.
[19] J. Rada-Vilela, M. Johnston, and M. Zhang, "Population statistics for particle swarm optimization: Hybrid methods in noisy optimization problems," *Swarm Evol. Comput.*, vol. 22, pp. 15–29, Jun. 2015.
[20] S. Taghiyeh and J. Xu, "A new particle swarm optimization algorithm for noisy optimization problems," *Swarm Intell.*, vol. 10, no. 3, pp. 161–192, Jul. 2016.
[21] H. Xiao, L. H. Lee, and K. M. Ng, "Optimal computing budget allocation for complete ranking," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 516–524, Apr. 2014.
[22] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part I: Background and development," *Natural Comput.*, vol. 6, no. 4, pp. 467–484, Oct. 2007.
[23] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Comput.*, vol. 7, no. 1, pp. 109–124, Mar. 2008.
[24] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, Honolulu, HI, USA, Apr. 2007, pp. 120–127.
[25] M. Clerc and J. Kennedy, "The particle swarm–explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[26] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no 2, pp. 150–194, 2013.
[27] S. H. Choi, J. H. Lee, S. H. Lee, H. D. Yoo, J. Koo, and T. G. Kim, "6 DOF aircraft simulation model capable of handling maneuver events (WIP)," in *Proc. Summer Comput. Simulation Conf.*, Jul. 2016, Art. no. 54.
[28] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in CPS-based production systems," *Procedia Manuf.*, vol. 11, pp. 939–948, Sep. 2017.

SEON HAN CHOI (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2012, 2014, and 2018, respectively.

He was a Teaching Assistant for computer programming with the School of Electrical Engineering, from 2013 to 2017. In 2018, he was a Postdoctoral Researcher with the Information and Electronics Research Institute, KAIST. From 2018 to 2019, he was a Senior Researcher with the Korea Institute of Industrial Technology. In 2019, he joined the Department of IT Convergence and Application Engineering, Pukyong National University, Busan, South Korea, as an Assistant Professor. His current research interests include the modeling and simulation of discrete-event systems, efficient simulation optimization under stochastic noise, evolutionary computing, and machine learning.

JANG WON BAE (Member, IEEE) received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2001, and the M.S. degree in electrical engineering and the Ph.D. degree in industrial and systems engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2007 and 2015, respectively.

From 2015 to 2020, he was a Senior Researcher with the Electronics and Telecommunications Research Institute, Daejeon. Since 2020, he has been an Assistant Professor with the School of Industrial Management, Korea University of Technology and Education, Cheonan, South Korea. His research interests include agent-based modeling and simulation, simulation-based optimization, and interplay between simulation and AI techniques.