IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 18, 2022, accepted June 5, 2022, date of publication June 17, 2022, date of current version June 23, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3183131

Orthogonal Design-Based Control Vector Parameterization Combined With Improved Seagull Optimization Algorithm for Dynamic Optimization Problems

HAIDONG GUO[1], YUANBIN MO[1,2], AND YUEDONG ZHANG[3]
[1]School of Artificial Intelligence, Guangxi Minzu University, Nanning 530006, China
[2]Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning 530006, China
[3]College of Electronic Information, Guangxi Minzu University, Nanning 530006, China

Corresponding author: Yuanbin Mo (moyuanbin2020@gxmzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 21466008, in part by the Guangxi Natural Science Foundation under Grant 2019GXNSFAA185017, and in part by the School Level Scientific Research Project under Grant 2021MDKJ004.

ABSTRACT This paper proposes an orthogonal design-based control vector parameterization (OCVP, for short) and a Gaussian distribution-based seagull optimization algorithm (GSOA) for dynamic optimization problems (DOPs). OCVP uses orthogonal experimental design to analyze the dynamic model to capture the fluctuation characteristics of the optimal control trajectory. Then using the ranges obtained in the orthogonal experiment to guide the construction of the time grid. Based on the seagull optimization algorithm (SOA), GSOA introduces the initialization idea based on Gaussian distribution and the dimension-order mutation operator based on Gaussian distribution. The initialization idea cleverly uses Gaussian distribution to generate the initial population that conforms to the chemical process. The mutation operator uses the dimension-order mutation method to improve the optimization performance of SOA. OCVP and GSOA are combined to form a new optimization method, named OCVP-GSOA. In the application of four typical chemical DOPs, the simulation results show that OCVP-GSOA can achieve similar or even higher solution accuracy. Furthermore, OCVP and control vector parameterization are compared, and GSOA and other meta-heuristic algorithms are compared. The results show that OCVP can achieve higher solution accuracy in most cases, and GSOA can achieve better performance.

INDEX TERMS Dynamic optimization, Gaussian distribution, nonuniform control vector parameterization, orthogonal design, seagull optimization algorithm.

I. INTRODUCTION

With the development of society, the energy demand is increasing. Seeking an excellent optimization method of the chemical control process has become a hot spot of social concern. The optimization of the chemical control process is usually expressed as a dynamic optimization problem [1], which improves the performance index of the chemical process by optimizing the control scheme. Therefore, dynamic optimization plays an increasingly important role in the chemical industry [2]. At present, the solution methods of dynamic optimization problems (DOPs) include solution method based on Pontryagin Principle, iterative dynamic programming (IDP) [3], control vector parameterization (CVP) [4], nonuniform control vector parameterization, and intelligent optimization algorithm.

The solution method based on Pontryagin Principle is an early solution method for DOPs. This method transforms DOP into the corresponding two-point boundary value problem with the help of the Hamilton system and the first-order necessary optimality condition. Istratie [5] used this method to study skip reentry trajectory optimization. This method has the advantage of high accuracy, but it is difficult to solve the complex high-dimensional DOP. IDP is a solution method with global convergence. It needs to discretize the time horizon and the control domain at the same time and uses

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

the iterative idea to gradually approach the optimal control strategy. As one of the methods that can effectively solve DOPs, IDP has been applied in many practical problems. Luus [3] combined IDP with the penalty function to determine the optimal control strategy of the fed-batch fermentor and achieved good results. Woinaroschy *et al.* [6] used IDP to optimize the fedbatch bioreactor, and further discussed the influence of IDP parameters on the optimal control strategy. Sundaralingam [7] used IDP to optimize DOP with state inequality constraints and introduced a two-step method to obtain the final solution. Numerical experiments show that this method can obtain a better performance index and reduce time cost. IDP has certain advantages in terms of computational cost and solution accuracy, but it also has limitations. IDP will become complicated with the increase of control variables and state variables [8].

With the development of computer technology, CVP has gradually become the mainstream method for solving DOPs. In CVP, the state variables maintain original continuity, while the control variables are discretized into finite control parameters over the time horizon. This can transform DOP into a nonlinear programming problem of control parameters. In recent years, CVP was used by many scholars to solve chemical DOPs. Zhou *et al.* [9] proposed an iteratively adaptive particle swarm optimization approach. The proposed approach combined with CVP achieved higher accuracy in solving several classical chemical DOPs. Zhang *et al.* [10] proposed a hybrid algorithm HAPSODSA-CVP based on adaptive particle swarm optimization (APSO) and differential search algorithm (DSA) and verified the effectiveness of the proposed algorithm on three nonlinear chemical DOPs. Tian *et al.* [11] combined the invasive weed optimization (IWO) algorithm with CVP to propose a new optimal approach IWO-CVP for chemical DOPs, and further proposed an optimal approach based on adaptive dispersion IWO, named ADIWO-CVP. Among them, the ADIWO-CVP approach showed better solution performance. As an effective method for solving DOPs, CVP also has certain limitations. CVP usually uses the uniform discrete strategy to divide the time horizon to create a uniform time grid. However, the uniform time grid lacks adaptability, which often makes it difficult for CVP to approximate the optimal control trajectory well. Therefore, many scholars have proposed nonuniform control vector parameterization method. Teo *et al.* [12] proposed a classic Time-Scaling method. This method regards the width of each grid and the control parameters as the object to be optimized to obtain a set of control parameters closer to the theoretical solution and a better time grid. Binder *et al.* [13] combined signal processing technology with CVP and proposed a CVP based on wavelet analysis. This method uses wavelets to evaluate potential mesh points to optimize the time grid obtained in the previous step. Li *et al.* [14] proposed a CVP with variable time nodes. In solving the optimal multivariable control problem, different control variables are parameterized using different time grids. This method showed better flexibility and universality. Xu *et al.* [15] proposed an adaptive CVP based on pseudo Wigner-Ville. With the help of pseudo Wigner-Ville and variable time node CVP method, this method can find the best time grid and achieve the purpose of reducing the calculation cost and improving the accuracy of the solution. Compared with CVP, nonuniform control vector parameterization can obtain higher solution accuracy, and its discrete strategy is also more flexible.

In recent years, intelligent optimization algorithms have been widely used to solve various optimization problems. It has the advantages of strong robustness and easy programming. CVP can transform DOPs into nonlinear programming problems, so intelligent optimization algorithms are also widely used to solve DOPs. Aiming at the multidimensional and nonlinear characteristics of DOPs, Sun *et al.* [16] proposed a hybrid improved genetic algorithm (HIGA), which improved the convergence speed and solution accuracy of the algorithm by introducing the simplex method, protecting the best individual and other operations. Chen *et al.* [17] combined the respective advantages of particle swarm optimization (PSO) and gradient-based algorithm (GBA) to propose a hybrid gradient particle swarm optimization (HGPSO) algorithm. In the solution of chemical DOPs, the HGPSO algorithm showed better solution accuracy and optimization performance. To develop the potential of artificial raindrop algorithm (ARA) to solve multi-objective optimization problems, Jiang *et al.* [18] proposed a multi-objective artificial raindrop algorithm (MOARA) and applied it to chemical DOPs. Tabassum *et al.* [19] proposed a differential gradient evolution plus (DGE+) algorithm. The algorithm combined differential evolution, gradient evolution, and jumping technique. It has strong global exploration performance and has achieved good results in the optimization of several bio-chemical reactors. In addition to the methods mentioned above, other excellent methods, such as environment sensitivity-based cooperative co-evolutionary algorithms [20] and adaptive fuzzy control [21], [22], can also provide references for proposing a new optimization method for DOPs. According to the research status of DOPs, nonuniform control vector parameterization and intelligent optimization algorithms are more competitive solution methods.

In this research, we propose a novel nonuniform control vector parameterization and an improved seagull optimization algorithm. The novel nonuniform control vector parameterization is named the orthogonal design-based control vector parameterization (OCVP). Based on CVP, OCVP introduces orthogonal experimental design to analyze the dynamic model and uses the range of each factor obtained in the orthogonal experiment to guide the construction of the time grid to obtain a better time grid. The improved seagull optimization algorithm is named the Gaussian distribution-based seagull optimization algorithm (GSOA). Aiming at the characteristics of chemical DOPs, this paper proposes an initialization idea based on Gaussian distribution and a dimension-order mutation operator based on Gaussian distribution to improve the seagull optimization algorithm

(SOA) [23]. Finally, OCVP and GSOA are combined to form a new optimization method, named OCVP-GSOA. In the simulation experiments of four typical chemical DOPs, OCVP-GSOA can construct a better time grid and achieve higher solution accuracy in most cases. Moreover, OCVP is compared with CVP, and GSOA is compared with other metaheuristic algorithms to show their superiority.

The main contributions of this research are as follows: (1) A novel nonuniform control vector parameterization (OCVP) is proposed, which provides a better control vector parameterization strategy for solving DOPs. (2) For chemical DOPs, an initialization idea based on Gaussian distribution is proposed. The initial population generated under the guidance of this idea will be more in line with the continuity of the chemical control process. (3) A dimension-order mutation operator based on Gaussian distribution is proposed. (4) A Gaussian distribution-based seagull optimization algorithm (GSOA) is proposed, which effectively improves the ability of SOA to solve chemical DOPs. (5) Combining OCVP and GSOA to form OCVP-GSOA, which provides a better solution method for researchers in this field.

The rest of this paper is organized as below. Section II depicts the dynamic optimization problem and the control vector parameterization method. Section III presents the orthogonal design-based control vector parameterization method. Section IV presents the Gaussian distribution-based seagull optimization algorithm. Section V verifies the performance of OCVP-GSOA on four typical chemical DOPs. The conclusion is summarized in the end.

II. PROBLEM DESCRIPTION AND CONTROL VECTOR PARAMETERIZATION

A. PROBLEM DESCRIPTION

The mathematical model of DOP can be stated as follows:

$$\max J = \Phi[\boldsymbol{x}(t_f), t_f] + \int_{t_0}^{t_f} \Psi[\boldsymbol{x}(t), \boldsymbol{u}(t), t]\mathrm{d}t,$$
$$\text{s.t.} \begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ g_i[\boldsymbol{x}(t), \boldsymbol{u}(t), t] = 0, i = 1, 2, \ldots, \upsilon \\ h_j[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \le 0, j = 1, 2, \ldots, \omega \\ \boldsymbol{u}_{\min} \le \boldsymbol{u}(t) \le \boldsymbol{u}_{\max} \\ t_0 \le t \le t_f, \end{cases} \tag{1}$$

where $[t_0, t_f]$ and $[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}]$ denote the time horizon and the control domain of the dynamic system, respectively;g_i and h_j denote equality constraints and inequality constraints, respectively; $\boldsymbol{x}(t) \in R^m$ is the state vector, and $\boldsymbol{x}_0$ is its initial state; $\boldsymbol{u}(t) \in R^n$ denotes the control vector and is also the object to be optimized; $\boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t]$ is the differential equation describing the dynamic system; J is the performance index for the dynamic process. $\Phi[\boldsymbol{x}(t_f), t_f]$ denotes the terminal value term at the terminal time t_f. $\int_{t_0}^{t_f} \Psi[\boldsymbol{x}(t), \boldsymbol{u}(t), t]\mathrm{d}t$ denotes the integral term over the time horizon $[t_0, t_f]$.

B. CONTROL VECTOR PARAMETERIZATION

CVP can convert an infinite dimension DOP into a finite dimension nonlinear programming problem, which approximates the optimal control trajectory with basis functions containing finite parameters. The detailed steps of CVP are as follows:

CVP usually uses the uniform discrete strategy to divide the time horizon $[t_0, t_f]$ into N equal-length time subintervals $[t_{i-1}, t_i](i = 1, 2, \ldots, N)$, as shown in Equation (2):

$$t_0 \le t_1 \le \ldots \le t_{N-1} \le t_N = t_f. \tag{2}$$

These time nodes $t_0, t_1, \ldots, t_N$, which divide the time horizon, are fixed throughout the solution cycle. For the control vector $\boldsymbol{u}(t) \in R^n$, the paper records its j-th control component as $u_j(t)$ $(j = 1, 2, \ldots, n)$. CVP uses Equation (3) to take a simple polynomial over each time subinterval to denote $u_j(t)$.

$$u_j(t) = \sum\nolimits_{i=1}^{N} u_j^i(t)\chi_i(t), \tag{3}$$

$$\chi_i(t) = \begin{cases} 1, & t \in [t_{i-1}, t_i) \\ 0, & t \notin [t_{i-1}, t_i), \end{cases} \tag{4}$$

where $\chi_i(t)$ denotes an indicator function, and Equation (4) gives its mathematical expression; $u_j^i(t)$ denotes the corresponding component of $u_j(t)$ over the time subinterval $[t_{i-1}, t_i]$. In Equation (5), $u_j^i(t)$ is approximated by a linear combination of a collection of basis functions.

$$u_j^i(t) \approx \sum\nolimits_{k=1}^{Q_{ij}+1} \sigma_{ijk}\phi_{ijk}^{Q_{ij}}(t), \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, n, \tag{5}$$

where σ_{ijk} denotes the coefficient of the linear combination, which is also considered as the control parameter; $\phi_{ijk}^{Q_{ij}}(t)$ denotes the Q_{ij}-order basis function. Equation (5) usually consists of some simple polynomials, such as the constant, the linear function, the quadratic polynomial, etc. Among them, the constant is the most common, that is, the piecewise constant parameterization ($Q_{ij} = 0$) is the most commonly used approximation strategy. The piecewise constant parameterization takes a control parameter σ_{ij1} $(i = 1, 2, \ldots, N)$ over each time subinterval to approximate the control trajectory of $u_j(t)$. Figure 1 shows the schematic diagram of the piecewise constant parameterization (taking $N = 10$ as an example).

III. ORTHOGONAL DESIGN-BASED CONTROL VECTOR PARAMETERIZATION

CVP provides an effective solution method for solving DOPs, but it still has room for improvement. CVP usually uses the uniform discrete strategy to evenly divide the entire time horizon, which makes it difficult for CVP to approximate the optimal control trajectory well. In most cases, the optimal control trajectory has different fluctuation amplitudes over different time subintervals. Therefore, dividing the entire time horizon evenly lacks adaptability. For the time subinterval with large fluctuation amplitude, it should be finely divided

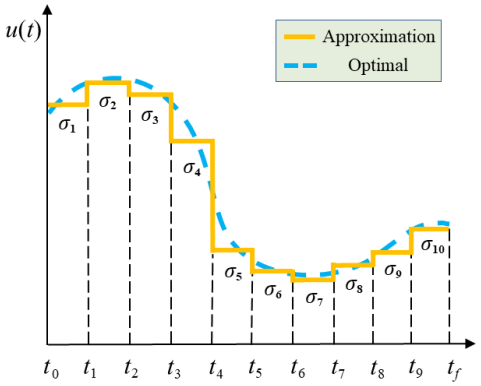


FIGURE 1. Schematic diagram of the piecewise constant parameterization.

to obtain higher approximation accuracy. For the time subinterval with small fluctuation amplitude, it should be roughly divided to reduce the number of control parameters, thereby reducing the computational cost. However, for DOP to be solved, how to know the fluctuation characteristics of the optimal control trajectory is a difficult problem. To solve this problem, inspired by the piecewise constant parameterization and orthogonal experimental design, this paper proposes an orthogonal design-based control vector parameterization (OCVP).

OCVP consists of three steps:

(1) Converting DOP to an 8-factor 7-level multi-factor experiment.

(2) Using orthogonal experimental design and selecting orthogonal table $L_{49}(7^8)$ to analyze this multi-factor experiment (MFE) to obtain the range of each factor.

(3) Using the ranges obtained in the orthogonal experiment to guide the construction of the time grid.

OCVP will be introduced in detail below.

A. CONVERTING DOP TO MFE

1) ORTHOGONAL DESIGN

Orthogonal design (also known as orthogonal experimental design) is a method for optimizing MFE. In orthogonal design, the orthogonal table can give an efficient experimental scheme, which uses some representative test points to replace the comprehensive test. Each row in the table is a level combination, representing a test point, and each column in the table is an experimental factor. The test points in the table are all strongly representative and evenly distributed among the comprehensive test. Orthogonal design combined with an orthogonal table can efficiently find the best level combination.

2) CONVERSION STEP

DOP and MFE are two different categories of problems, so the orthogonal experimental design cannot directly analyze the dynamic model. In this paper, OCVP selects the orthogonal table $L_{49}(7^8)$ to analyze the dynamic model. Therefore, the first step of OCVP is to convert DOP into an 8-factor 7-level multi-factor experiment.

The schematic diagram of converting DOP to MFE is shown in Figure 2. The solution space of DOP can be understood as a two-dimensional continuous search space composed of a time dimension and a control dimension, in which the time dimension has only one forward positive direction. The solving process of DOP can be understood as seeking the optimal control trajectory that changes continuously with time in the solution space. OCVP uses Equation (6) to evenly divide the entire time horizon $[t_0, t_f]$ into 8 equal-length time subintervals $[t_{i-1}, t_i](i = 1, 2, \ldots, 8)$.

$$t_0 < t_1 < \ldots < t_7 < t_8 = t_f, \tag{6}$$

where the time subinterval $[t_{i-1}, t_i]$ corresponds to the factor i in the orthogonal table $L_{49}(7^8)$. For the control domain $[u_{i,\min}, u_{i,\max}]$ over the time subinterval $[t_{i-1}, t_i]$, it is evenly divided into 7 equal-length control subintervals $[u_{i,j-1}, u_{i,j}](i = 1, 2, \ldots, 8; j = 1, 2, \ldots, 7)$ using Equation (7).

$$u_{i,\min} = u_{i,0} < u_{i,1} < \ldots < u_{i,6} < u_{i,7} = u_{i,\max}. \tag{7}$$

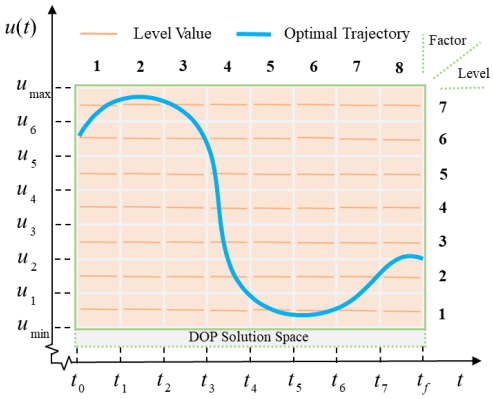


FIGURE 2. Schematic diagram of converting DOP to MFE.

Finally, the median value of the control subinterval $[u_{i,j-1}, u_{i,j}]$ is taken as the value $v_{i,j}$ of level j of factor i using Equation (8).

$$v_{i,j} = (u_{i,j-1} + u_{i,j})/2. \tag{8}$$

B. ORTHOGONAL EXPERIMENT AND RANGE ANALYSIS

1) ORTHOGONAL EXPERIMENT

In this subsection, we will introduce the steps of the orthogonal experiment in detail.

First, each level in the orthogonal table $L_{49}(7^8)$ is converted into the value of the level of the corresponding factor to obtain the orthogonal experiment scheme in Table 1. This orthogonal experiment scheme has a total of 49 test points. In OCVP, a test point corresponds to a set of control parameters. Taking the first test point as an example, OCVP takes the values of 8 levels in the first test point as a set of control parameters $\boldsymbol{\sigma}_1$

TABLE 1. The scheme of the orthogonal experiment.

Number	Experimental factors								Experimental result
	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8	
1	2 $v_{1,2}$	6 $v_{2,6}$	3 $v_{3,3}$	6 $v_{4,6}$	7 $v_{5,7}$	5 $v_{6,5}$	1 $v_{7,1}$	6 $v_{8,6}$	Result 1
2	2 $v_{1,2}$	4 $v_{2,4}$	7 $v_{3,7}$	5 $v_{4,5}$	1 $v_{5,1}$	3 $v_{6,3}$	3 $v_{7,3}$	3 $v_{8,3}$	Result 2
3	7 $v_{1,7}$	7 $v_{2,7}$	5 $v_{3,5}$	5 $v_{4,5}$	7 $v_{5,7}$	1 $v_{6,1}$	4 $v_{7,4}$	2 $v_{8,2}$	Result 3
4	5 $v_{1,5}$	5 $v_{2,5}$	6 $v_{3,6}$	6 $v_{4,6}$	5 $v_{5,5}$	1 $v_{6,1}$	3 $v_{7,3}$	4 $v_{8,4}$	Result 4
5	7 $v_{1,7}$	6 $v_{2,6}$	7 $v_{3,7}$	1 $v_{4,1}$	4 $v_{5,4}$	7 $v_{6,7}$	5 $v_{7,5}$	4 $v_{8,4}$	Result 5
6	4 $v_{1,4}$	4 $v_{2,4}$	3 $v_{3,3}$	3 $v_{4,3}$	4 $v_{5,4}$	1 $v_{6,1}$	6 $v_{7,6}$	5 $v_{8,5}$	Result 6
7	1 $v_{1,1}$	2 $v_{2,2}$	6 $v_{3,6}$	5 $v_{4,5}$	4 $v_{5,4}$	2 $v_{6,2}$	7 $v_{7,7}$	6 $v_{8,6}$	Result 7
8	3 $v_{1,3}$	3 $v_{2,3}$	7 $v_{3,7}$	7 $v_{4,7}$	3 $v_{5,3}$	1 $v_{6,1}$	2 $v_{7,2}$	6 $v_{8,6}$	Result 8
9	7 $v_{1,7}$	1 $v_{2,1}$	3 $v_{3,3}$	2 $v_{4,2}$	3 $v_{5,3}$	2 $v_{6,2}$	3 $v_{7,3}$	7 $v_{8,7}$	Result 9
10	5 $v_{1,5}$	2 $v_{2,2}$	5 $v_{3,5}$	1 $v_{4,1}$	3 $v_{5,3}$	5 $v_{6,5}$	6 $v_{7,6}$	3 $v_{8,3}$	Result 10
11	4 $v_{1,4}$	7 $v_{2,7}$	4 $v_{3,4}$	1 $v_{4,1}$	6 $v_{5,6}$	4 $v_{6,4}$	3 $v_{7,3}$	6 $v_{8,6}$	Result 11
12	3 $v_{1,3}$	1 $v_{2,1}$	4 $v_{3,4}$	6 $v_{4,6}$	4 $v_{5,4}$	6 $v_{6,6}$	4 $v_{7,4}$	3 $v_{8,3}$	Result 12
13	4 $v_{1,4}$	1 $v_{2,1}$	2 $v_{3,2}$	5 $v_{4,5}$	2 $v_{5,2}$	5 $v_{6,5}$	2 $v_{7,2}$	4 $v_{8,4}$	Result 13
14	5 $v_{1,5}$	1 $v_{2,1}$	7 $v_{3,7}$	4 $v_{4,4}$	7 $v_{5,7}$	4 $v_{6,4}$	7 $v_{7,7}$	5 $v_{8,5}$	Result 14
15	2 $v_{1,2}$	7 $v_{2,7}$	1 $v_{3,1}$	3 $v_{4,3}$	3 $v_{5,3}$	6 $v_{6,6}$	7 $v_{7,7}$	4 $v_{8,4}$	Result 15
16	6 $v_{1,6}$	7 $v_{2,7}$	7 $v_{3,7}$	6 $v_{4,6}$	2 $v_{5,2}$	2 $v_{6,2}$	6 $v_{7,6}$	1 $v_{8,1}$	Result 16
17	1 $v_{1,1}$	7 $v_{2,7}$	3 $v_{3,3}$	4 $v_{4,4}$	5 $v_{5,5}$	7 $v_{6,7}$	2 $v_{7,2}$	3 $v_{8,3}$	Result 17
18	4 $v_{1,4}$	3 $v_{2,3}$	5 $v_{3,5}$	6 $v_{4,6}$	1 $v_{5,1}$	7 $v_{6,7}$	7 $v_{7,7}$	7 $v_{8,7}$	Result 18
19	6 $v_{1,6}$	2 $v_{2,2}$	3 $v_{3,3}$	7 $v_{4,7}$	1 $v_{5,1}$	4 $v_{6,4}$	4 $v_{7,4}$	4 $v_{8,4}$	Result 19
20	5 $v_{1,5}$	3 $v_{2,3}$	3 $v_{3,3}$	5 $v_{4,5}$	6 $v_{5,6}$	6 $v_{6,6}$	5 $v_{7,5}$	1 $v_{8,1}$	Result 20
21	2 $v_{1,2}$	2 $v_{2,2}$	4 $v_{3,4}$	4 $v_{4,4}$	2 $v_{5,2}$	1 $v_{6,1}$	5 $v_{7,5}$	7 $v_{8,7}$	Result 21
22	1 $v_{1,1}$	5 $v_{2,5}$	7 $v_{3,7}$	3 $v_{4,3}$	6 $v_{5,6}$	5 $v_{6,5}$	4 $v_{7,4}$	7 $v_{8,7}$	Result 22
23	1 $v_{1,1}$	4 $v_{2,4}$	2 $v_{3,2}$	6 $v_{4,6}$	3 $v_{5,3}$	4 $v_{6,4}$	5 $v_{7,5}$	2 $v_{8,2}$	Result 23
24	2 $v_{1,2}$	1 $v_{2,1}$	6 $v_{3,6}$	7 $v_{4,7}$	6 $v_{5,6}$	7 $v_{6,7}$	6 $v_{7,6}$	2 $v_{8,2}$	Result 24
25	3 $v_{1,3}$	6 $v_{2,6}$	1 $v_{3,1}$	5 $v_{4,5}$	5 $v_{5,5}$	4 $v_{6,4}$	6 $v_{7,6}$	7 $v_{8,7}$	Result 25
26	7 $v_{1,7}$	3 $v_{2,3}$	6 $v_{3,6}$	3 $v_{4,3}$	2 $v_{5,2}$	4 $v_{6,4}$	1 $v_{7,1}$	3 $v_{8,3}$	Result 26
27	6 $v_{1,6}$	3 $v_{2,3}$	1 $v_{3,1}$	4 $v_{4,4}$	4 $v_{5,4}$	5 $v_{6,5}$	3 $v_{7,3}$	2 $v_{8,2}$	Result 27
28	1 $v_{1,1}$	1 $v_{2,1}$	1 $v_{3,1}$	1 $v_{4,1}$	1 $v_{5,1}$	1 $v_{6,1}$	1 $v_{7,1}$	1 $v_{8,1}$	Result 28
29	5 $v_{1,5}$	6 $v_{2,6}$	4 $v_{3,4}$	3 $v_{4,3}$	1 $v_{5,1}$	2 $v_{6,2}$	2 $v_{7,2}$	2 $v_{8,2}$	Result 29
30	7 $v_{1,7}$	2 $v_{2,2}$	1 $v_{3,1}$	6 $v_{4,6}$	6 $v_{5,6}$	3 $v_{6,3}$	2 $v_{7,2}$	5 $v_{8,5}$	Result 30
31	2 $v_{1,2}$	5 $v_{2,5}$	5 $v_{3,5}$	2 $v_{4,2}$	4 $v_{5,4}$	4 $v_{6,4}$	2 $v_{7,2}$	1 $v_{8,1}$	Result 31
32	4 $v_{1,4}$	2 $v_{2,2}$	7 $v_{3,7}$	2 $v_{4,2}$	5 $v_{5,5}$	6 $v_{6,6}$	1 $v_{7,1}$	2 $v_{8,2}$	Result 32
33	3 $v_{1,3}$	2 $v_{2,2}$	2 $v_{3,2}$	3 $v_{4,3}$	7 $v_{5,7}$	7 $v_{6,7}$	3 $v_{7,3}$	1 $v_{8,1}$	Result 33
34	7 $v_{1,7}$	5 $v_{2,5}$	2 $v_{3,2}$	4 $v_{4,4}$	1 $v_{5,1}$	6 $v_{6,6}$	6 $v_{7,6}$	6 $v_{8,6}$	Result 34
35	4 $v_{1,4}$	6 $v_{2,6}$	6 $v_{3,6}$	4 $v_{4,4}$	3 $v_{5,3}$	3 $v_{6,3}$	4 $v_{7,4}$	1 $v_{8,1}$	Result 35
36	6 $v_{1,6}$	1 $v_{2,1}$	5 $v_{3,5}$	3 $v_{4,3}$	5 $v_{5,5}$	3 $v_{6,3}$	5 $v_{7,5}$	6 $v_{8,6}$	Result 36
37	6 $v_{1,6}$	4 $v_{2,4}$	6 $v_{3,6}$	1 $v_{4,1}$	7 $v_{5,7}$	6 $v_{6,6}$	2 $v_{7,2}$	7 $v_{8,7}$	Result 37
38	7 $v_{1,7}$	4 $v_{2,4}$	4 $v_{3,4}$	7 $v_{4,7}$	5 $v_{5,5}$	5 $v_{6,5}$	7 $v_{7,7}$	1 $v_{8,1}$	Result 38
39	4 $v_{1,4}$	5 $v_{2,5}$	1 $v_{3,1}$	7 $v_{4,7}$	7 $v_{5,7}$	2 $v_{6,2}$	5 $v_{7,5}$	3 $v_{8,3}$	Result 39
40	3 $v_{1,3}$	7 $v_{2,7}$	6 $v_{3,6}$	2 $v_{4,2}$	1 $v_{5,1}$	5 $v_{6,5}$	5 $v_{7,5}$	5 $v_{8,5}$	Result 40
41	5 $v_{1,5}$	4 $v_{2,4}$	1 $v_{3,1}$	2 $v_{4,2}$	2 $v_{5,2}$	7 $v_{6,7}$	4 $v_{7,4}$	6 $v_{8,6}$	Result 41
42	6 $v_{1,6}$	5 $v_{2,5}$	4 $v_{3,4}$	5 $v_{4,5}$	3 $v_{5,3}$	7 $v_{6,7}$	1 $v_{7,1}$	5 $v_{8,5}$	Result 42
43	1 $v_{1,1}$	3 $v_{2,3}$	4 $v_{3,4}$	2 $v_{4,2}$	7 $v_{5,7}$	3 $v_{6,3}$	6 $v_{7,6}$	4 $v_{8,4}$	Result 43
44	6 $v_{1,6}$	6 $v_{2,6}$	2 $v_{3,2}$	2 $v_{4,2}$	6 $v_{5,6}$	1 $v_{6,1}$	7 $v_{7,7}$	3 $v_{8,3}$	Result 44
45	3 $v_{1,3}$	4 $v_{2,4}$	5 $v_{3,5}$	4 $v_{4,4}$	6 $v_{5,6}$	2 $v_{6,2}$	1 $v_{7,1}$	4 $v_{8,4}$	Result 45
46	3 $v_{1,3}$	5 $v_{2,5}$	3 $v_{3,3}$	1 $v_{4,1}$	2 $v_{5,2}$	3 $v_{6,3}$	7 $v_{7,7}$	2 $v_{8,2}$	Result 46
47	2 $v_{1,2}$	3 $v_{2,3}$	2 $v_{3,2}$	1 $v_{4,1}$	5 $v_{5,5}$	2 $v_{6,2}$	4 $v_{7,4}$	5 $v_{8,5}$	Result 47
48	5 $v_{1,5}$	7 $v_{2,7}$	2 $v_{3,2}$	7 $v_{4,7}$	4 $v_{5,4}$	3 $v_{6,3}$	1 $v_{7,1}$	7 $v_{8,7}$	Result 48
49	1 $v_{1,1}$	6 $v_{2,6}$	5 $v_{3,5}$	7 $v_{4,7}$	2 $v_{5,2}$	6 $v_{6,6}$	3 $v_{7,3}$	5 $v_{8,5}$	Result 49

and uses Equation (9) to express the control trajectory $u_1(t)$.

$$\begin{aligned} u_1(t) &= \sum_{i=1}^{8} \boldsymbol{\sigma}_1(i)\chi_i(t), \\ \boldsymbol{\sigma}_1 &= [v_{1,2}, v_{2,6}, v_{3,3}, v_{4,6}, v_{5,7}, v_{6,5}, v_{7,1}, v_{8,6}], \\ \chi_i(t) &= \begin{cases} 1, & t \in [t_{i-1}, t_i) \\ 0, & t \notin [t_{i-1}, t_i). \end{cases} \end{aligned} \tag{9}$$

Second, the control trajectory $u_1(t)$ is brought into the mathematical model of DOP, as shown in Equation (10), and the test result Result 1 is calculated by the Runge-Kutta method.

$$\text{Result } 1 = \Phi[\boldsymbol{x}(t_f), t_f] + \int_{t_0}^{t_f} \Psi[\boldsymbol{x}(t), \sum_{i=1}^{8} \boldsymbol{\sigma}_1(i)\chi_i(t), t]\mathrm{d}t,$$

$$\text{s.t.} \begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \sum_{i=1}^{8} \boldsymbol{\sigma}_1(i)\chi_i(t), t] \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ g_i[\boldsymbol{x}(t), \sum_{i=1}^{8} \boldsymbol{\sigma}_1(i)\chi_i(t), t] = 0, \quad i = 1, 2, \ldots, \upsilon \\ h_j[\boldsymbol{x}(t), \sum_{i=1}^{8} \boldsymbol{\sigma}_1(i)\chi_i(t), t] \le 0, \quad j = 1, 2, \ldots, \omega \\ \boldsymbol{u}_{\min} \le \sum_{i=1}^{8} \boldsymbol{\sigma}_1(i)\chi_i(t) \le \boldsymbol{u}_{\max} \\ t_0 \le t \le t_f. \end{cases} \tag{10}$$

OCVP uses the same steps to complete all the test points in this orthogonal experiment scheme and calculate all test results Result k ($k = 1, 2, \ldots, 49$). Finally, the ranges of all factors are calculated based on all test results. The formula for calculating range Range i of factor i is as follows:

$$\begin{aligned} \text{Range } i = &\ \text{Max}(K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}, K_{i,7}) \\ &- \text{Min}(K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}, K_{i,7}), \end{aligned} \tag{11}$$

where $K_{i,j}$ ($i = 1, 2, \ldots, 8; j = 1, 2, \ldots, 7$) takes the average value of test results of all test points, containing level j of factor i, in this orthogonal experiment scheme. Taking $K_{1,1}$ as an example, the detailed calculation process is as follows:

$$\begin{aligned} K_{1,1} = &\ (\text{Result } 7 + \text{Result } 17 + \text{Result } 22 + \text{Result } 23 \\ &+ \text{Result } 28 + \text{Result } 43 + \text{Result } 49)/7. \end{aligned} \tag{12}$$

2) RANGE ANALYSIS

In OCVP, the range Range i is large, which indicates that the optimal control trajectory has a small fluctuation amplitude over the time subinterval $[t_{i-1}, t_i]$. On the contrary, the range Range i is small, which indicates that the optimal control trajectory has a large fluctuation amplitude over the time subinterval $[t_{i-1}, t_i]$. This conclusion is explained in detail below.

Orthogonal experimental design has symmetrical comparability. When analyzing the range Range i of factor i, it can be considered that the effects of the other factors $r (r \neq i)$ on $K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}$, and $K_{i,7}$ are roughly the same. The differences between $K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}$, and $K_{i,7}$ are mainly caused by the fact that factor i takes different values of level.

As shown in Figure 2, factor i corresponds to the time subinterval $[t_{i-1}, t_i]$, and each factor has the same levels. A value of level corresponds to a horizontal control trajectory in the solution space of DOP. If the fluctuation amplitude of the optimal control trajectory over the time subinterval $[t_{i-1}, t_i]$ is small, then the contour of the optimal control trajectory is similar to that of the horizontal control trajectory. Among the 7 horizontal control trajectories corresponding to $v_{i,1}, v_{i,2}, v_{i,3}, v_{i,4}, v_{i,5}, v_{i,6}$, and $v_{i,7}$, the horizontal control trajectory, which is closer to the optimal control trajectory or has more parts that overlap with the optimal control trajectory, will obtain a better K value. Conversely, the horizontal control trajectory, which is farther from the optimal control trajectory or has fewer parts that overlap with the optimal control trajectory, will obtain a worse K value. There will be an excellent K value and a very poor K value in $K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}$, and $K_{i,7}$, which makes the range Range i large. On the contrary, if the fluctuation amplitude of the optimal control trajectory over the time subinterval $[t_{i-1}, t_i]$ is large, then the contour of the optimal control trajectory is very different from that of the horizontal control trajectory, which leads to $K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}$, and $K_{i,7}$ are all poor. There is no excellent K value in $K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}$, and $K_{i,7}$, and the gap between $K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}, K_{i,6}$, and $K_{i,7}$ won't be huge. This makes the range Range i small.

3) SELECTION OF ORTHOGONAL TABLE

For the orthogonal table selected by OCVP, the number of factors and levels should not be too large or too small. If the number of factors is too large, the length of each time subinterval will be too small. The fluctuation amplitude that can be captured over each time subinterval will be very weak, which makes it difficult for OCVP to effectively obtain the fluctuation characteristics of the optimal control trajectory. If the number of factors is too small, the length of each time subinterval will be too large. In a single time subinterval, the optimal control trajectory may have the characteristics of large fluctuation amplitude and small fluctuation amplitude at the same time, which makes OCVP cannot accurately distinguish the part with large fluctuation amplitude and the part with small fluctuation amplitude on the optimal control trajectory. If the number of levels is too large, the number of test points in the orthogonal experiment scheme will increase dramatically. If the number of levels is too small, it will reduce the accuracy of OCVP to capture the fluctuation characteristics of the optimal control trajectory. Therefore, this paper uses the orthogonal table $L_{49}(7^8)$, which has an appropriate number of factors and levels, to design the orthogonal experiment scheme. We use SPSS software to generate the orthogonal table $L_{49}(7^8)$. The orthogonal table $L_{49}(7^8)$ has a variety of experiment schemes, and the time grids constructed

by different schemes may vary. In this paper, the experiment scheme with the best effect is selected in Table 1.

C. CONSTRUCTION OF TIME GRID

OCVP uses ranges Range 1,Range 2,...,Range 8 obtained in the orthogonal experiment to guide the construction of the time grid. The detailed steps are as follows:

Step 1: Set the number of time subintervals N over the time horizon $[t_0, t_f]$; initialize the amplifier $y = (e^k)^x$, set $k = 1$; and get ranges Range 1,Range 2,...,Range 8.

Step 2: Use Equation (13) to normalize the eight ranges Range $i (i = 1, 2, \ldots, 8)$ to obtain $\text{Range}_{\text{nor}}\ i$; Record the largest of $\text{Range}_{\text{nor}}\ 1$,$\text{Range}_{\text{nor}}\ 2$,...,$\text{Range}_{\text{nor}}\ 8$ as $\text{Max}_{\text{value}}$; Then, use Equation (14) to calculate $\text{Range}_{\text{pos}}\ 1$, $\text{Range}_{\text{pos}}\ 2, \ldots, \text{Range}_{\text{pos}}\ 8$.

$$\text{Range}_{\text{nor}}\ i = \frac{\text{Range } i}{(\text{Range } 1 + \text{Range } 2 + \ldots + \text{Range } 8)}, \tag{13}$$

$$\text{Range}_{\text{pos}}\ i = \text{Max}_{\text{value}} - \text{Range}_{\text{nor}}\ i + 1. \tag{14}$$

Step 3: Use Equation (15) to normalize the eight $\text{Range}_{\text{pos}}\ i$ to obtain the weight coefficient of each time subinterval Weight i; Then, use the amplifier $y = (e^k)^x$ to amplify Weight i to obtain the new weight coefficient $\text{Weight}_{\text{amp}}\ i$, as shown in Equation (16); Next, use Equation (17) to normalize the eight $\text{Weight}_{\text{amp}}\ i$ to obtain $\text{Weight}_{\text{amp\_nor}}\ i$.

$$\text{Weight } i = \frac{\text{Range}_{\text{pos}}\ i}{(\text{Range}_{\text{pos}}\ 1 + \text{Range}_{\text{pos}}\ 2 + \ldots + \text{Range}_{\text{pos}}\ 8)}, \tag{15}$$

$$\text{Weight}_{\text{amp}}\ i = (e^k)^{\text{Weight } i}, \tag{16}$$

$$\text{Weight}_{\text{amp\_nor}}\ i = \frac{\text{Weight}_{\text{amp}}\ i}{(\text{Weight}_{\text{amp}}\ 1 + \text{Weight}_{\text{amp}}\ 2 + \ldots + \text{Weight}_{\text{amp}}\ 8)}. \tag{17}$$

Step 4: Subtract 8 quotas from N to get N^*, that is,$N^* = N - 8$. The purpose is to reserve 1 quota in advance for each time subinterval to ensure that each time subinterval has at least 1 quota; Then, multiply $\text{Weight}_{\text{amp\_nor}}\ i$ corresponding to each time subinterval by N^* and round up to obtain the quota of each time subinterval $\text{Quota}^*\ i$, as shown in Equation (18).

$$\text{Quota}^*\ i = \left\lfloor (\text{Weight}_{\text{amp\_nor}}\ i \cdot N^*) + \frac{1}{2} \right\rfloor. \tag{18}$$

Step 5: Calculate the total number of quotas $\text{Quota}^*_{\text{sum}}$, that is, $\text{Quota}^*_{\text{sum}} = \text{Quota}^*\ 1 + \text{Quota}^*\ 2 + \cdots + \text{Quota}^*\ 8$; If $\text{Quota}^*_{\text{sum}} > N^*$, subtract 1 quota from the time subinterval with the most quota, subtract 1 quota from the time subinterval with the second most quota, and so on, until $\text{Quota}^*_{\text{sum}} = N^*$. If $\text{Quota}^*_{\text{sum}} < N^*$, add 1 quota to the time subinterval with the most quota, add 1 quota to the time subinterval with the second most quota, and so on, until $\text{Quota}^*_{\text{sum}} = N^*$.

Step 6: Allocate 1 quota reserved for each time subinterval to each time subinterval, that is, Quota i = $\text{Quota}^*\ i + 1$. At this point, the allocation of quotas for each time subinterval is completed.

Step 7: Record the largest of Quota 1,Quota 2,...,Quota 8 as $\text{Quota}_{\text{Max}}$, and the smallest as $\text{Quota}_{\text{Min}}$; If $(\text{Quota}_{\text{Max}} - \text{Quota}_{\text{Min}}) > 0.1N$, set $k = k - 1$ and go to Step 8. If $(\text{Quota}_{\text{Max}} - \text{Quota}_{\text{Min}}) \leq 0.1N$, set $k = k + 1$ and go to Step 3.

Step 8: At this point, a suitable amplifier has been found. Execute Step 3, Step 4, Step 5, and Step 6, which uses the found amplifier to complete the allocation of quotas for each time subinterval.

Step 9: Construct a time grid over the time horizon $[t_0, t_f]$. Divide the time subinterval $[t_{i-1}, t_i]$ $(i = 1, 2, \ldots, 8)$ into Quota i equal parts. Finally, output the time grid.

The algorithm flow chart of OCVP is shown in Figure 3.

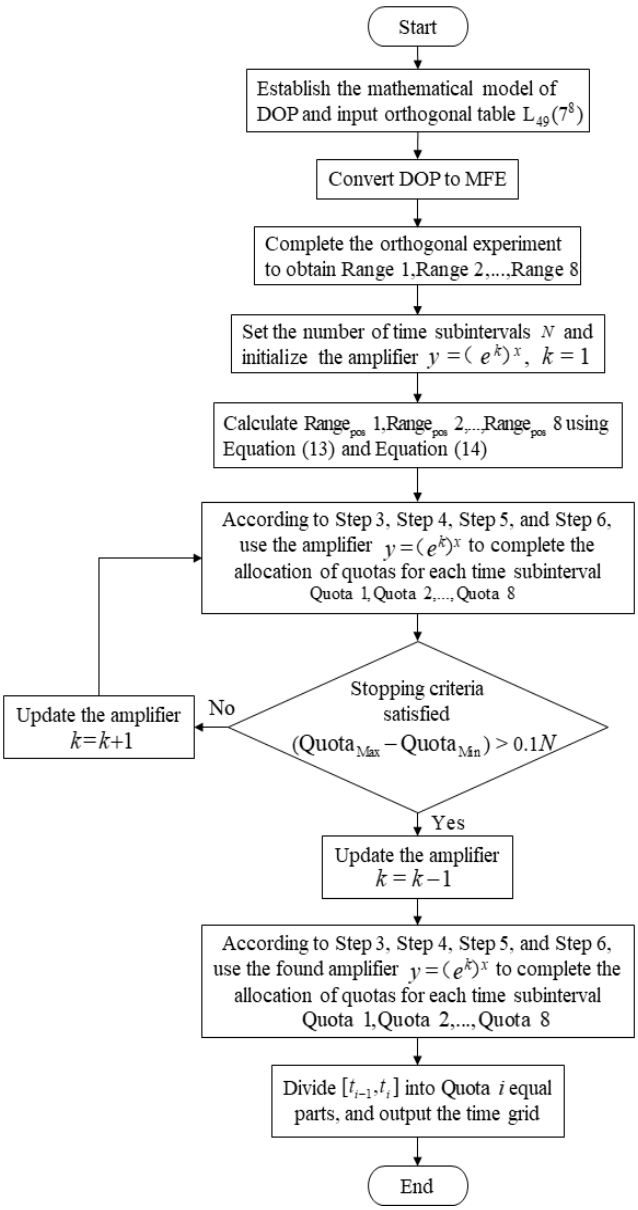


FIGURE 3. The algorithm flow chart of OCVP.

IV. GAUSSIAN DISTRIBUTION-BASED SEAGULL OPTIMIZATION ALGORITHM

A. SEAGULL OPTIMIZATION ALGORITHM

In recent years, SOA has been studied by many scholars [24]–[26]. In the search space, a search agent represents a seagull. Each search agent gradually approaches the global optimal solution by simulating migration behavior and attacking behavior.

Migration behavior: migration behavior helps SOA comprehensively explore the entire search space. In this phase, a search agent should satisfy the following three conditions:

- Avoiding the collisions: The equations for avoiding the collisions are shown in Equation (19) and Equation (20), which can increase the distance between adjacent search agents to avoid collisions.

$$\boldsymbol{C}_l = A\boldsymbol{P}_l^{\text{iter}}, \tag{19}$$

$$A = f_c - (\text{iter}(f_c/\text{Max}_{\text{iteration}})), \tag{20}$$

where $l = 1, 2, \ldots,$ pop, pop denotes the population size; iter indicates the current iteration; $\boldsymbol{P}_l^{\text{iter}}$ represents the current position of search agent; $\boldsymbol{C}_l$ represents the position of search agent after avoiding the collision; $\text{Max}_{\text{iteration}}$ denotes the maximum number of iterations; f_c denotes a constant parameter; A denotes the movement behavior of search agent. During the iteration process, A linearly decreases from f_c to 0.

- The direction of the best search agent: After avoiding the collision, the search agents move in the direction of the best search agent, as given by Equation (21) and Equation (22).

$$\boldsymbol{M}_l = B(\boldsymbol{P}_{\text{best}}^{\text{iter}} - \boldsymbol{P}_l^{\text{iter}}), \tag{21}$$

$$B = 2A^2\text{rd}, \tag{22}$$

where $\boldsymbol{P}_{\text{best}}^{\text{iter}}$ represents the best search agent in the population; $\boldsymbol{M}_l$ represents the direction of the best search agent;B is responsible for balancing exploration and exploitation; rd is a random number in [0, 1].

- Approaching the best search agent: The search agent updates its position according to the best search agent.

$$\boldsymbol{D}_l = |\boldsymbol{C}_l + \boldsymbol{M}_l|, \tag{23}$$

where $\boldsymbol{D}_l$ denotes the distance between the search agent and the best search agent.

Attacking behavior: when seagulls attack their prey, the flight trajectory of seagulls approximates a spiral curve. In x, y, and z planes, the attacking behavior can be described as follows:

$$x' = r\cos(k), \tag{24}$$

$$y' = r\sin(k), \tag{25}$$

$$z' = kr, \tag{26}$$

$$r = ue^{kv}, \tag{27}$$

$$\boldsymbol{P}_l^{\text{iter}} = \boldsymbol{D}_l x' y' z' + \boldsymbol{P}_{\text{best}}^{\text{iter}}, \tag{28}$$

where k is a random number in $[0, 2\pi]$, representing the angle of attack; r is the radius of the spiral flight trajectory; u and v are constants to define the shape of the spiral flight trajectory; $\boldsymbol{P}_l^{\text{iter}}$ saves the best solution and updates the position of other search agents.

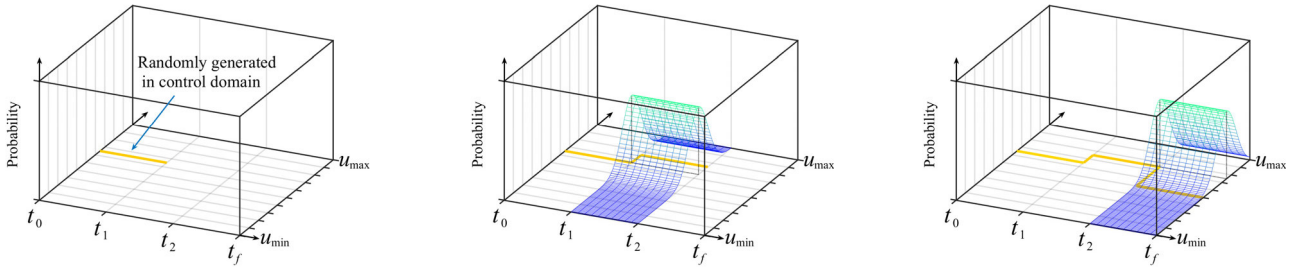
B. GAUSSIAN DISTRIBUTION-BASED SEAGULL OPTIMIZATION ALGORITHM

SOA is an effective optimizer for solving challenging large-scale constrained problems. But in the solution of chemical DOPs, SOA is often difficult to approximate the optimal control trajectory. The No Free Lunch (NFL) theorem [27] tells us that no one optimization algorithm can achieve excellent results on all optimization problems. For chemical DOPs, this paper proposes a Gaussian distribution-based seagull optimization algorithm (GSOA). Based on SOA, GSOA introduces the initialization idea based on Gaussian distribution and the dimension-order mutation operator based on Gaussian distribution, which effectively improves the ability of SOA to solve chemical DOPs.

1) INITIALIZATION IDEA BASED ON GAUSSIAN DISTRIBUTION

In practical production, the control scheme should have continuity, and the control scheme with small fluctuation is more in line with the characteristics of the chemical dynamic process [28], [29]. SOA adopts the traditional random initialization idea to generate the initial population, which makes each region in the search space has a certain probability to generate the initial individual. However, the random initialization idea is not suitable for solving chemical DOPs. The random initialization idea has certain blindness and uncertainty. The individuals generated by this idea are usually chaotic, and the difference between adjacent dimensions in the individual is often large. Such individuals are not usually in line with the continuity of the chemical dynamic process. To improve the quality of the initial population, this paper proposes an initialization idea based on Gaussian distribution. The initialization idea cleverly uses the characteristic of Gaussian distribution to generate the initial population, which can greatly improve the quality of the initial population. Figure 4 shows the schematic diagram of the initialization idea based on Gaussian distribution (taking $N = 3$ as an example). The detailed steps are as follows:

Initialization of search agent $\boldsymbol{P}_l = (p_{l,1}, p_{l,2}, \ldots, p_{l,N})$: First,$p_{l,1}$ is randomly generated in the control domain $[u_{\min}, u_{\max}]$ using Equation (29). Next, $p_{l,2}$ is generated using Equation (30). $\varsigma_{l,2}$ is a random number generated from a Gaussian distribution with mean $\mu = p_{l,1}$ and standard deviation $\sigma = (u_{\max} - u_{\min})/10$. If $\varsigma_{l,2} \notin [u_{\min}, u_{\max}]$, Equation (30) is used again to generate $p_{l,2}$. $\varsigma_{l,2}$ is still a random number generated from a Gaussian distribution with mean $\mu = p_{l,1}$ and standard deviation $\sigma = (u_{\max} - u_{\min})/10$. Until $\varsigma_{l,2} \in [u_{\min}, u_{\max}]$. Similarly, $p_{l,3}, p_{l,4}, \ldots, p_{l,N}$ are



(a) Generating $p_{l,1}$ (b) Generating $p_{l,2}$ (c) Generating $p_{l,3}$

FIGURE 4. Schematic diagram of the initialization idea based on Gaussian distribution.

generated in sequence.

$$p_{l,1} = (u_{\max} - u_{\min})\mathrm{rd} + u_{\min}, \tag{29}$$

$$\begin{cases} p_{l,I} = \varsigma_{l,I}, \\ \varsigma_{l,I} \sim N(p_{l,I-1}, ((u_{\max} - u_{\min})/10)^2), \end{cases} \tag{30}$$

where $l = 1, 2, \ldots,$ pop, pop denotes the population size; $I = 2, 3, \ldots, N$, N is the dimension of search space; $p_{l,1}$ is the value of the 1-th dimension of search agent $\boldsymbol{P}_l = (p_{l,1}, p_{l,2}, \ldots, p_{l,N})$; $p_{l,I}$ is the value of the I-th dimension of search agent $\boldsymbol{P}_l = (p_{l,1}, p_{l,2}, \ldots, p_{l,N})$; $u_{\max}$ and $u_{\min}$ are the upper and lower bounds of the control domain, respectively; rd is a random number in $[0, 1]$; $\varsigma_{l,I}$ is a random number generated from a Gaussian distribution with mean $\mu = p_{l,I-1}$ and standard deviation $\sigma = (u_{\max} - u_{\min})/10$. A random number generated from a Gaussian distribution $N(\mu, \sigma^2)$ has about 99.7% probability to lie in the range of $[\mu-3\sigma, \mu+3\sigma]$. Setting $\sigma = (u_{\max} - u_{\min})/10$, the initialization idea based on Gaussian distribution can not only avoid generating a large number of chaotic initial individuals but also has a small probability to generate individuals with large fluctuation to avoid missing the potential best individual with large fluctuation.

2) DIMENSION-ORDER MUTATION OPERATOR BASED ON GAUSSIAN DISTRIBUTION

In the solution of chemical DOPs, SOA is easy to fall into local optimum. Because in SOA, the evolution of the population is guided by the best search agent. In the high-dimensional and complex search space, the best search agent is easy to fall into local optimum, which leads to poor quality of the population. To improve the ability of SOA to solve chemical DOPs, this paper proposes a dimension-order mutation operator based on Gaussian distribution. The mutation strategy is a common improvement strategy in intelligent optimization algorithms, which can effectively improve the ability of algorithms to jump out of the local optimum and solution accuracy [30]–[32]. Aiming at the characteristics of chemical DOPs, the dimension-order mutation operator based on Gaussian distribution performs Gaussian mutation on the best search agent dimension by dimension, according to the order of dimensions, to improve the global search performance of SOA. Taking dynamic optimization problem $\max J$ as an example, the steps of the dimension-order mutation operator based on Gaussian distribution are as follows:

(1) $\boldsymbol{P}_{\text{best}}^{\text{iter}} = (p_{\text{best},1}^{\text{iter}}, p_{\text{best},2}^{\text{iter}}, \ldots, p_{\text{best},N}^{\text{iter}})$ represents the best search agent at the iter-th iteration, and its performance index is denoted as $J_{\text{best}}^{\text{iter}}$. $\boldsymbol{P}_{\text{nb}}^{\text{iter}} = (p_{\text{nb},1}^{\text{iter}}, p_{\text{nb},2}^{\text{iter}}, \ldots, p_{\text{nb},N}^{\text{iter}})$ represents the new best search agent.

(2) For $\boldsymbol{P}_{\text{best}}^{\text{iter}}$, its 1-th dimension mutates to generate the mutated search agent $\boldsymbol{P}_{\text{mutant}}^{\text{iter},1} = (p_{\text{mutant},1}^{\text{iter}}, p_{\text{best},2}^{\text{iter}}, \ldots, p_{\text{best},N}^{\text{iter}})$. The value $p_{\text{mutant},1}^{\text{iter}}$ of the $I = 1$-th dimension of $\boldsymbol{P}_{\text{mutant}}^{\text{iter},1}$ is calculated using Equation (31). If $p_{\text{mutant},1}^{\text{iter}} > u_{\max}(p_{\text{mutant},1}^{\text{iter}} < u_{\min})$, set $p_{\text{mutant},1}^{\text{iter}} = u_{\max}(p_{\text{mutant},1}^{\text{iter}} = u_{\min})$. The values of the remaining dimensions of $\boldsymbol{P}_{\text{mutant}}^{\text{iter},1}$ are all equal to the values of the corresponding dimensions of $\boldsymbol{P}_{\text{best}}^{\text{iter}}$.

(3) Calculate the performance index $J_{\text{mutant}}^{\text{iter},1}$ of $\boldsymbol{P}_{\text{mutant}}^{\text{iter},1}$. If $J_{\text{mutant}}^{\text{iter},1} > J_{\text{best}}^{\text{iter}}$, set $p_{\text{nb},1}^{\text{iter}} = p_{\text{mutant},1}^{\text{iter}}$. If $J_{\text{mutant}}^{\text{iter},1} \le J_{\text{best}}^{\text{iter}}$, set $p_{\text{nb},1}^{\text{iter}} = p_{\text{best},1}^{\text{iter}}$.

(4) For $\boldsymbol{P}_{\text{best}}^{\text{iter}}$, its 2-th dimension mutates to generate the mutated search agent $\boldsymbol{P}_{\text{mutant}}^{\text{iter},2} = (p_{\text{best},1}^{\text{iter}}, p_{\text{mutant},2}^{\text{iter}}, \ldots, p_{\text{best},N}^{\text{iter}})$. The value $p_{\text{mutant},2}^{\text{iter}}$ of the $I = 2$-th dimension of $\boldsymbol{P}_{\text{mutant}}^{\text{iter},2}$ is calculated using Equation (31). If $p_{\text{mutant},2}^{\text{iter}} > u_{\max}(p_{\text{mutant},2}^{\text{iter}} < u_{\min})$, set $p_{\text{mutant},2}^{\text{iter}} = u_{\max}(p_{\text{mutant},2}^{\text{iter}} = u_{\min})$. The values of the remaining dimensions of $\boldsymbol{P}_{\text{mutant}}^{\text{iter},2}$ are all equal to the values of the corresponding dimensions of $\boldsymbol{P}_{\text{best}}^{\text{iter}}$.

(5) Calculate the performance index $J_{\text{mutant}}^{\text{iter},2}$ of $\boldsymbol{P}_{\text{mutant}}^{\text{iter},2}$. If $J_{\text{mutant}}^{\text{iter},2} > J_{\text{best}}^{\text{iter}}$, set $p_{\text{nb},2}^{\text{iter}} = p_{\text{mutant},2}^{\text{iter}}$. If $J_{\text{mutant}}^{\text{iter},2} \le J_{\text{best}}^{\text{iter}}$, set $p_{\text{nb},2}^{\text{iter}} = p_{\text{best},2}^{\text{iter}}$.

(6) Similarly, according to the order of dimensions, complete the mutation operations of the remaining dimensions of $\boldsymbol{P}_{\text{best}}^{\text{iter}}$. Finally, the new best search agent $\boldsymbol{P}_{\text{nb}}^{\text{iter}} = (p_{\text{nb},1}^{\text{iter}}, p_{\text{nb},2}^{\text{iter}}, \ldots, p_{\text{nb},N}^{\text{iter}})$ can be obtained.

$$\begin{cases} p_{\text{mutant},I}^{\text{iter}} = \xi_I^{\text{iter}}, \\ \xi_I^{\text{iter}} \sim N(p_{\text{best},I}^{\text{iter}}, ((u_{\max} - u_{\min})/G)^2), \\ G = 500 - (490 - (\text{iter}(490/\text{Max}_{\text{iteration}}))), \end{cases} \tag{31}$$

where $I = 1, 2, \ldots, N$, $N$is the dimension of search space; $u_{\max}$ and $u_{\min}$ are the upper and lower bounds of the control domain, respectively; $p_{\text{mutant},I}^{\text{iter}}$ is the value of the I-th dimension of the mutated search agent $\boldsymbol{P}_{\text{mutant}}^{\text{iter},I}$; $p_{\text{best},I}^{\text{iter}}$ is the value

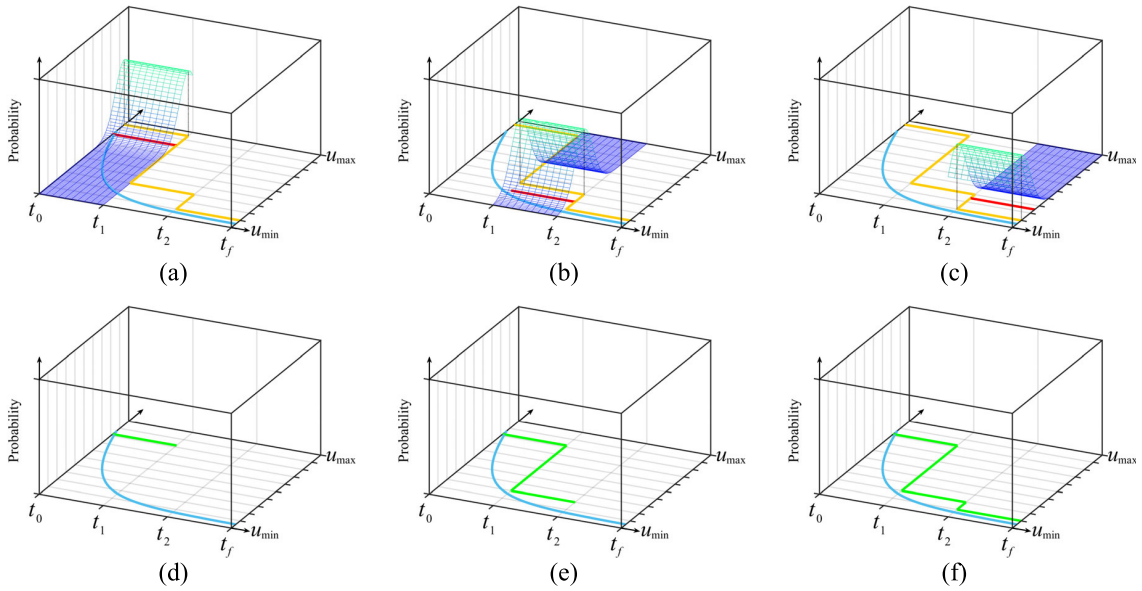


FIGURE 5. Schematic diagram of the dimension-order mutation operator based on Gaussian distribution. The blue trajectory represents the theoretically optimal trajectory, the yellow trajectory represents the best search agent, the red trajectory represents the mutated value, and the green trajectory represents the new best search agent. (a), (b), (c) show the mutation process of the best search agent. (d), (e), (f) show the generation process of the new best search agent.

of the I-th dimension of the best search agent $\boldsymbol{P}_{\text{best}}^{\text{iter}}$; iter indicates the current iteration; $\text{Max}_{\text{iteration}}$denotes the maximum number of iterations; ξ_I^{iter} is a random number generated from a Gaussian distribution with mean $\mu = p_{\text{best},I}^{\text{iter}}$ and standard deviation $\sigma = (u_{\max} - u_{\min})/G$. Figure 5 shows the schematic diagram of the dimension-order mutation operator based on Gaussian distribution (taking $N = 3$ as an example).

V. OPTIMIZATION PROBLEMS AND SIMULATION EXPERIMENTS

In this section, OCVP and GSOA are combined to form a new optimization method for DOPs, named OCVP-GSOA. Simulation experiments are carried out to test the feasibility and effectiveness of OCVP-GSOA. First, we state the execution procedures of OCVP-GSOA. Second, four classic DOPs are described. These DOPs are widely used as benchmark problems, and their control trajectories have different shapes. Finally, OCVP-GSOA is applied to these DOPs, and the simulation results are compared with optimization methods in the related literature. Moreover, OCVP is compared with CVP, and GSOA is compared with other meta-heuristic algorithms.

All simulation experiments are carried out on a desktop computer with Intel (R) Core (TM) i7-10700/2.90GHz, 16.0GB memory. The experimental software is MATLAB R2020a, and the operating system is Windows 10 Education 64-bit.

A. EXECUTION PROCEDURE OF OCVP-GSOA

The execution procedures of solving DOPs using the proposed OCVP-GSOA is as follows:

Step 1: Determine the necessary computational parameters, including the number of time subintervals N, the population size pop, the mutation times $\text{Mutation}_{\text{times}}$, and the maximum number of iterations $\text{Max}_{\text{iteration}}$. Initialize the basic parameters of SOA, including f_c, u, and v. Input the orthogonal table $L_{49}(7^8)$.

Step 2: Use OCVP in Section III to construct the time grid.

Step 3: Use the initialization idea based on Gaussian distribution in Section IV.B.1 to generate the initial population. The control trajectory is approximated using the piecewise constant parameterization. A search agent represents a control trajectory.

Step 4: Use the SOA algorithm in Equations (19)-(28) to update the positions of all search agents.

Step 5: Calculate the performance indexes of all search agents using the Runge-Kutta method. Update the position of the best search agent.

Step 6: The best search agent is mutated $\text{Mutation}_{\text{times}}$times using the dimension-order mutation operator based on Gaussian distribution.

Step 7: If the maximum number of iterations is reached, output the best search agent; otherwise, turn to Step 4.

The pseudocode of OCVP-GSOA is given in Table 2.

B. TEST PROBLEMS

1) PROBLEM 1: BATCH REACTOR CONSECUTIVE REACTION

The reaction process in a batch reactor is A $\rightarrow$ B $\rightarrow$ C. The mathematical model of this problem is as follows [11], [28], [33]–[36]:

$$\begin{aligned} &\max J(t_f) = C_{\text{B}}(t_f), \\ &\text{s.t.} \begin{cases} \frac{dC_{\text{A}}}{dt} = -k_1 C_{\text{A}}^2 \\ \frac{dC_{\text{B}}}{dt} = k_1 C_{\text{A}}^2 - k_2 C_{\text{B}} \\ 298 \le T \le 398, C_{\text{A}}(0) = 1, C_{\text{B}}(0) = 0, t_f = 1 \\ k_1 = 4 \times 10^3 e^{-2500/T}, k_2 = 6.2 \times 10^5 e^{-5000/T}, \end{cases} \end{aligned} \tag{32}$$

TABLE 2. Pseudocode of OCVP-GSOA.

Algorithm: OCVP-GSOA

Input: The number of time subintervals N, the population size pop, the mutation times $\text{Mutation}_{\text{times}}$, the maximum number of iterations $\text{Max}_{\text{iteration}}$, and the orthogonal table $\text{L}_{49}(7^8)$

Output: The best search agent $\boldsymbol{P}_{\text{best}}$

procedure OCVP-GSOA
Initialize the basic parameters of SOA (i.e., set $f_c \leftarrow 2$, $u \leftarrow 1$, and $v \leftarrow 1$)
$TimeGrid \leftarrow$ **OCVP** (N) /* Use **OCVP** function to construct the time grid */
for $l = 1 : \text{pop}$ /* Generate the initial population on $TimeGrid$ */
  Search agent $\boldsymbol{P}_l$ is generated using the initialization idea based on Gaussian distribution
end for
Calculate the performance indexes of all search agents using the Runge-Kutta method
Find the best search agent $\boldsymbol{P}_{\text{best}}$
while ($\text{iter} < \text{Max}_{\text{iteration}}$) **do**
  for $l = 1 : \text{pop}$ /* SOA optimizer */
    Update the position of search agent $\boldsymbol{P}_l$ by Equations (19) - (28)
    Check if search agent $\boldsymbol{P}_l$ goes beyond the given search space and then adjusts it
    Calculate the performance index of $\boldsymbol{P}_l$ using the Runge-Kutta method
  end for
  Update $\boldsymbol{P}_{\text{best}}$ if there is a better search agent than previous best search agent
  for $x = 1 : \text{Mutation}_{\text{times}}$
    $\boldsymbol{P}_{\text{best}}$ is mutated using the dimension-order mutation operator based on Gaussian distribution to obtain a new best search agent $\boldsymbol{P}_{\text{nb}}$
    Set $\boldsymbol{P}_{\text{best}} \leftarrow \boldsymbol{P}_{\text{nb}}$
  end for
end while
return $\boldsymbol{P}_{\text{best}}$
end procedure

procedure OCVP (N)
Convert DOP to MFE by Equations (6) - (8)
Complete the 49 test points in the orthogonal table $\text{L}_{49}(7^8)$, such as Equations (9) - (10), to obtain $\text{Result}\ 1, \text{Result}\ 2, ..., \text{Result}\ 49$
for $i = 1 : 8$
  Calculate $\text{Range}\ i$ of factor i using Equation (11)
end for
Set $k \leftarrow 1$ /* Initialize the amplifier $y = (e^k)^x$ */
Calculate $\text{Range}_{\text{pos}}\ 1, \text{Range}_{\text{pos}}\ 2, ..., \text{Range}_{\text{pos}}\ 8$ using Equations (13) - (14)
while (1) **do** /* Find a suitable amplifier */
  Set $N^* \leftarrow N - 8$ /* Reserve 1 quota in advance for each time subinterval */
  Calculate $\text{Quota}^*\ 1, \text{Quota}^*\ 2, ..., \text{Quota}^*\ 8$ using Equations (15) - (18)
  Set $\text{Quota}^*_{\text{sum}} \leftarrow \text{Quota}^*\ 1 + \text{Quota}^*\ 2 + ... + \text{Quota}^*\ 8$
  Adjust $\text{Quota}^*_{\text{sum}}$ until $\text{Quota}^*_{\text{sum}} = N^*$, if $\text{Quota}^*_{\text{sum}} \neq N^*$
  for $i = 1 : 8$ /*Allocate 1 quota reserved for each time subinterval to each time subinterval*/
    Set $\text{Quota}\ i \leftarrow \text{Quota}^*\ i + 1$
  end for

TABLE 2. ***(Continued.)*** **Pseudocode of OCVP-GSOA.**

```
if (Quota_Max − Quota_Min) > 0.1N
  Set k ← k − 1 and break while
end if
Set k ← k + 1
end while
Set N* ← N − 8          /* Use the found amplifier to allocate quotas to each time subinterval */
Calculate Quota* 1, Quota* 2, ..., Quota* 8 using Equations (15) - (18)
Set Quota*_sum ← Quota* 1 + Quota* 2 + ... + Quota* 8
Adjust Quota*_sum until Quota*_sum = N*, if Quota*_sum ≠ N*
for i = 1 : 8
  Set Quota i ← Quota* i + 1
end for
for i = 1 : 8           /* Construct TimeGrid over the time horizon [t_0, t_f] */
  Divide the time subinterval [t_{i−1}, t_i] into Quota i equal parts
end for
return TimeGrid
end procedure
```

where T denotes the temperature (i.e., control variable); C_{A} and C_{B} denote the concentrations of A and B, respectively; $C_{\mathrm{A}}(0)$ and $C_{\mathrm{B}}(0)$ denote the initial concentrations of A and B, respectively; k_1 and k_2 are model parameters; t_f denotes the final time; J is the performance index.

2) PROBLEM 2: PLUG FLOW REACTOR CATALYST BLEND PROBLEM

In the reactor, the reaction process is A $\leftrightarrow$ B $\rightarrow$ C. The mathematical model of problem 2 is as follows [34], [36]–[39]:

$$\begin{aligned} &\max J(z_f) = 1 - x_{\mathrm{A}}(z_f) - x_{\mathrm{B}}(z_f), \\ &\text{s.t.} \begin{cases} \frac{dx_{\mathrm{A}}}{dz} = u(z)[10x_{\mathrm{B}}(z) - x_{\mathrm{A}}(z)] \\ \frac{dx_{\mathrm{B}}}{dz} = -u(z)[10x_{\mathrm{B}}(z) - x_{\mathrm{A}}(z)] - [1 - u(z)]x_{\mathrm{B}}(z) \\ 0 \le u(z) \le 1 \\ x_{\mathrm{A}}(0) = 1, x_{\mathrm{B}}(0) = 0, z_f = 12, \end{cases} \end{aligned} \tag{33}$$

where $u(z)$ denotes the fraction of the type 1 catalyst at position z in the reactor (i.e., control variable); x_{A} and x_{B} denote the mole fractions of A and B, respectively; $x_{\mathrm{A}}(0)$ and $x_{\mathrm{B}}(0)$ denote the initial mole fractions of A and B, respectively; z_f is the length of the reactor; J is the performance index.

3) PROBLEM 3: TUBULAR REACTOR PARALLEL REACTION PROBLEM

In the tubular reactor, the parallel reaction: A $\rightarrow$ B, A $\rightarrow$ C takes place. The mathematical model of problem 3 is as follows [34]–[36], [40], [41]:

$$\begin{aligned} &\max J(t_f) = x_2(t_f), \\ &\text{s.t.} \begin{cases} \frac{dx_1}{dt} = -[u(t) + 0.5u^2(t)]x_1(t) \\ \frac{dx_2}{dt} = u(t)x_1(t) \\ x_1(0) = 1, x_2(0) = 0 \\ 0 \le u(t) \le 5, t_f = 1, \end{cases} \end{aligned} \tag{34}$$

TABLE 3. **Parameters of OCVP-GSOA for Problem 1.**

parameter	value
population size pop	30
mutation times $\mathrm{Mutation}_{\mathrm{times}}$	15
maximum number of iterations $\mathrm{Max}_{\mathrm{iteration}}$	20

where $u(t)$ denotes the control variable;x_1 and $x_2$denote the dimensionless concentration of A and B, respectively; $x_1(0)$ and $x_2(0)$ denote the initial dimensionless concentrations of A and B, respectively; $t_f$denotes the final time; J is the performance index.

4) PROBLEM 4: PLUG-FLOW TUBULAR REACTOR

The mathematical model of this problem is as follows [15], [42]–[44]:

$$\begin{aligned} &\max J = x_1(t_f), \\ &\text{s.t.} \begin{cases} \frac{dx_1}{dt} = (1 - x_1)k_1 - x_1k_2 \\ \frac{dx_2}{dt} = 300[(1 - x_1)k_1 - x_1k_2] - u(x_2 - 290) \\ k_1 = 1.7536 \times 10^5 e^{(-1.1374 \times 10^4 / 1.9872x_2)} \\ k_2 = 2.4885 \times 10^{10} e^{(-2.2748 \times 10^4 / 1.9872x_2)} \\ x_1(0) = 0, x_2(0) = 410 \\ 0 \le u \le 0.5, t_f = 5, \end{cases} \end{aligned} \tag{35}$$

TABLE 4. Simulation results for Problem 1.

	methods	N	best	mean	worst	std	time/s
Problem 1 (max)	OCVP-GSOA	50	0.61076309	0.61075373	0.61060504	3.38710530E-05	4.01
		150	0.61080037	0.61078903	0.61054798	4.51313832E-05	32.01
		350	0.61080242	0.61077526	0.61055237	5.06561239E-05	165.92
	PSO-CVP [33]	25	0.6105359	-[a]	-	-	193
	IKBCA [28]	20	0.610454	-	-	-	-
		100	0.610787	-	-	-	-
	ADIWO-CVP [11]	80	0.61079218	-	-	-	-
	ISOA [34]	30	0.610794203	-	-	-	118
	CP-APSO [35]	-	0.6107850	-	-	-	-
	IDP [36]	10	0.610070	-	-	-	105
		80	0.610775	-	-	-	-

[a] The symbol "-" means not reported.

where u denotes the control variable; k_1 and k_2 are rate constants; x_1 and $x_2$denote the normalized concentration of the desired product and the temperature, respectively; $x_1(0)$ and $x_2(0)$ denote the initial normalized concentration of the desired product and the initial temperature, respectively; t_f denotes the final time; J is the performance index.

C. RESULTS AND DISCUSSIONS

1) ANALYSIS OF THE SIMULATION RESULTS OF PROBLEM 1

For problem 1, the time horizon [0, 1] is divided into $N = 50, 150, 350$ time subintervals; and OCVP-GSOA is run 50 times independently. Table 3 presents the parameters of OCVP-GSOA for problem 1. Table 4 shows results obtained by OCVP-GSOA and other methods. Shi *et al.* used a hybrid strategy combined with particle swarm optimization and control vector parameterization method (PSO-CVP) to obtain a value of 0.6105359 ($N = 25$) [33]. Liu *et al.* used improved knowledge-based cultural algorithm (IKBCA) to obtain values of 0.610454 ($N = 20$) and 0.610787 ($N = 100$) [28]. Tian *et al.* reached a value of 0.61079218 ($N = 80$) using control vector parameterization based adaptive dispersion invasive weed optimization (ADIWO-CVP) [11]. Xu *et al.* achieved a value of 0.610794203 ($N = 30$) with improved seagull optimization algorithm (ISOA) [34]. Zhou and Liu obtained a value of 0.6107850 through a control parameterization-based adaptive particle swarm optimization (CP-APSO) approach [35]. Dadebo and McAuley applied IDP to obtain values of 0.610070 ($N = 10$) and 0.610775 ($N = 80$) [36].

Our OCVP-GSOA divides the time horizon into different numbers of time subintervals: 50, 150, and 350, respectively. When$N$is 350, the best result of OCVP-GSOA 0.61080242 is better than all results in the literature. When N is 150, the best result of OCVP-GSOA 0.61080037 is also better than all results in the literature, and the mean result of OCVP-GSOA 0.61078903 is close to the result 0.61079218 in [11]. Figure 6(c) shows the box plot of OCVP-GSOA of the different numbers of time subintervals (Problem 1). The results show that when N is 150, OCVP-GSOA is stable and can obtain a good result. In Table 4, time/s, which denotes the running time, reflects the optimization efficiency of optimization methods.

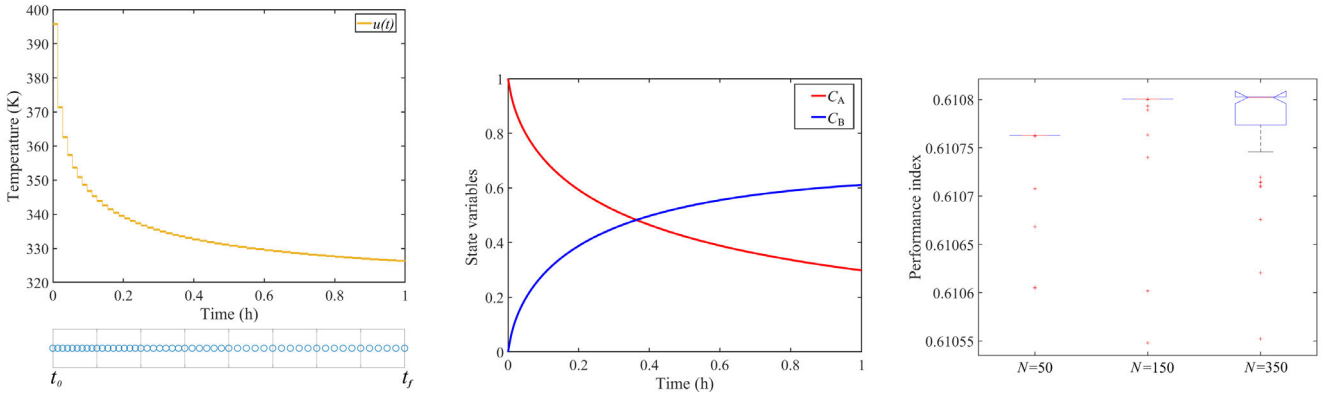
Figure 6(a) shows the optimal control trajectory and time grid obtained by OCVP-GSOA when $N = 50$. As can be seen, the fluctuation amplitude of the optimal control trajectory gradually decreases as time goes on. The time grid constructed by OCVP gradually changes from fine to coarse as time goes on. This is because OCVP can accurately capture the fluctuation characteristics of the optimal control trajectory of this problem, which enables OCVP to construct a more reasonable time grid. The state variables of this problem are plotted in Figure 6(b).

TABLE 5. Parameters of OCVP-GSOA for Problem 2.

parameter	value
population size pop	30
mutation times $\text{Mutation}_{\text{times}}$	20
maximum number of iterations $\text{Max}_{\text{iteration}}$	30

2) ANALYSIS OF THE SIMULATION RESULTS OF PROBLEM 2

For problem 2, the control variable is discretized into $N = 50, 100, 120$ control parameters; and OCVP-GSOA is run 50 times independently. The parameters of OCVP-GSOA for this problem are shown in Table 5. Table 6 presents results of OCVP-GSOA and other methods. Pham *et al.* obtained a value of 0.477444 ($N = 20$) with the Bees Algorithm (BA) [37]. Xu *et al.* used an improved seagull optimization algorithm (ISOA) and an unequal division method to obtain a value of 0.47770 ($N = 40$) [34]. On this problem, Dadebo and McAuley used IDP to obtain values of 0.475272 ($N = 20$) and 0.476946 ($N = 40$) [36]. Huang *et al.* obtained values of 0.473630 ($N = 10$) and 0.474531 ($N = 15$) using an improved state transition algorithm (STA) [38]. Chen *et al.* integrated nonuniform discretization-based CVP into hybrid gradient PSO to obtain a value of 0.47771 ($N = 15$) [39].



(a) Optimal control trajectory and time grid ($N = 50$)

(b) State variables

(c) Box plot

FIGURE 6. Results obtained by OCVP-GSOA. (Problem 1.)

TABLE 6. Simulation results for Problem 2.

	methods	N	best	mean	worst	std	time/s
Problem 2 (max)	OCVP-GSOA	50	0.47714685	0.47714685	0.47714685	1.79304095E-10	7.51
		100	0.47768634	0.47768629	0.47768619	4.41032846E-08	28.04
		120	0.47769527	0.47769517	0.47769496	7.45582891E-08	40.14
	BA [37]	20	0.477444	-	-	-	-
	ISOA [34]	40	0.47770	-	-	-	285
	IDP [36]	20	0.475272	-	-	-	595.6
		40	0.476946	-	-	-	-
	Improved STA [38]	10	0.473630	-	-	-	15.329338
		15	0.474531	-	-	-	33.564853
	ndCVP-HGPSO [39]	15	0.47771	-	-	-	-

From Table 6, when the number of control parameters N is 100, the best result of OCVP-GSOA 0.47768634 is better than the best results reported in [37], [36], and [38]. When N is 120, the best result of OCVP-GSOA is 0.47769527, very close to the best result of ISOA 0.47770 and the best result of ndCVP-HGPSO 0.47771, where ISOA and ndCVP-HGPSO also use nonuniform CVP to approximate the optimal control trajectory. Figure 7(c) shows the box plot of OCVP-GSOA of the different numbers of control parameters (Problem 2). The results show that OCVP-GSOA has excellent stability in solving problem 2.

The optimal control trajectory and time grid of problem 2 for $N = 100$ are plotted in Figure 7(a). The optimal control trajectory of this problem has two switch points. OCVP can accurately capture the first switch point and construct a finer time grid near the first switch point, which can better approximate the optimal control trajectory. The state variables of this problem are plotted in Figure 7(b).

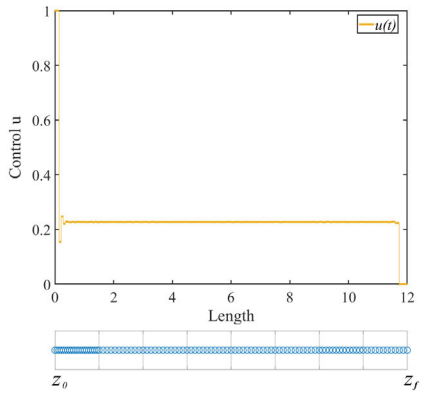
3) ANALYSIS OF THE SIMULATION RESULTS OF PROBLEM 3

For problem 3, the time horizon $[0, 1]$ is divided into $N = 50, 70, 150$ time subintervals; and OCVP-GSOA is run 50 times independently. Table 7 presents the parameters of OCVP-GSOA for problem 3. Table 8 lists a comparison of results obtained by OCVP-GSOA and other methods. Dadebo and McAuley obtained values of 0.57348 ($N = 40$) and 0.57353 ($N = 80$) using IDP [36]. Xu *et al.* solved problem 3 using an improved seagull optimization algorithm (ISOA) and obtained a value of 0.573535 ($N = 40$) [34]. Zhou and Liu got a value of 0.573544 using a control parameterization-based adaptive particle swarm optimization (CP-APSO) approach [35]. Biegler obtained values of 0.57349 and 0.56910 using control vector iteration (CVI) and control vector parameterization (CVP) respectively [40]. In the framework of ant colony optimization (ACO), Rajesh *et al.* obtained a value of 0.57284 [41].

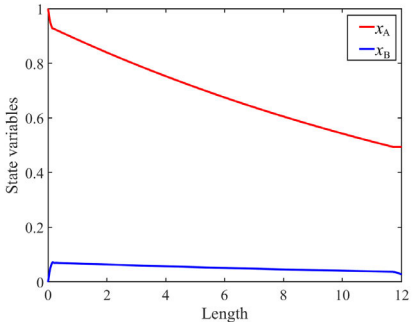
TABLE 7. Parameters of OCVP-GSOA for Problem 3.

parameter	value
population size pop	30
mutation times $\text{Mutation}_{\text{times}}$	15
maximum number of iterations $\text{Max}_{\text{iteration}}$	20

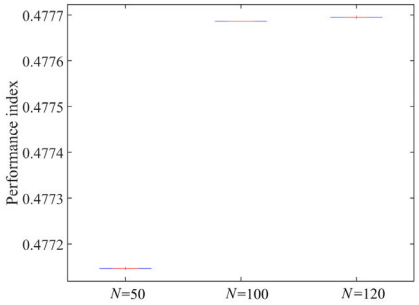
Our OCVP-GSOA can obtain a value of 0.57354134 ($N = 150$), better than the best results reported in [34], [36], [40], and [41]. And the best result obtained by OCVP-GSOA



(a) Optimal control trajectory and time grid ($N = 100$)



(b) State variables



(c) Box plot

FIGURE 7. Results obtained by OCVP-GSOA. (Problem 2.)

TABLE 8. Simulation results for Problem 3.

	methods	N	best	mean	worst	std	time/s
Problem 3 (max)	OCVP-GSOA	50	0.57351086	0.57349120	0.57304872	8.06832053E-05	3.90
		70	0.57352870	0.57350805	0.57314534	6.80400536E-05	7.34
		150	0.57354134	0.57348466	0.57306586	1.15501187E-04	30.58
	IDP [36]	40	0.57348	-	-	-	-
		80	0.57353	-	-	-	-
	ISOA [34]	40	0.573535	-	-	-	302
	CP-APSO [35]	-	0.573544	-	-	-	-
	CVI [40]	-	0.57349	-	-	-	-
	CVP [40]	-	0.56910	-	-	-	-
	ACO [41]	-	0.57284	-	-	-	-

0.57354134 is very close to the best result of CP-APSO 0.573544. Figure 8(c) shows the box plot of OCVP-GSOA of the different numbers of time subintervals (Problem 3). From the box plot and the standard deviation results of Table 8, when N is 70, OCVP-GSOA is stable and can obtain a good result.

Figure 8(a) shows the optimal control trajectory and time grid obtained by OCVP-GSOA when $N = 70$. The fluctuation amplitude of the optimal control trajectory of problem 3 gradually increases as time goes on. The time grid constructed by OCVP gradually changes from coarse to fine as time goes on, which indicates OCVP can also accurately capture the fluctuation characteristics of the optimal control trajectory of this problem. Figure 8(b) shows the state variables of problem 3.

4) ANALYSIS OF THE SIMULATION RESULTS OF PROBLEM 4

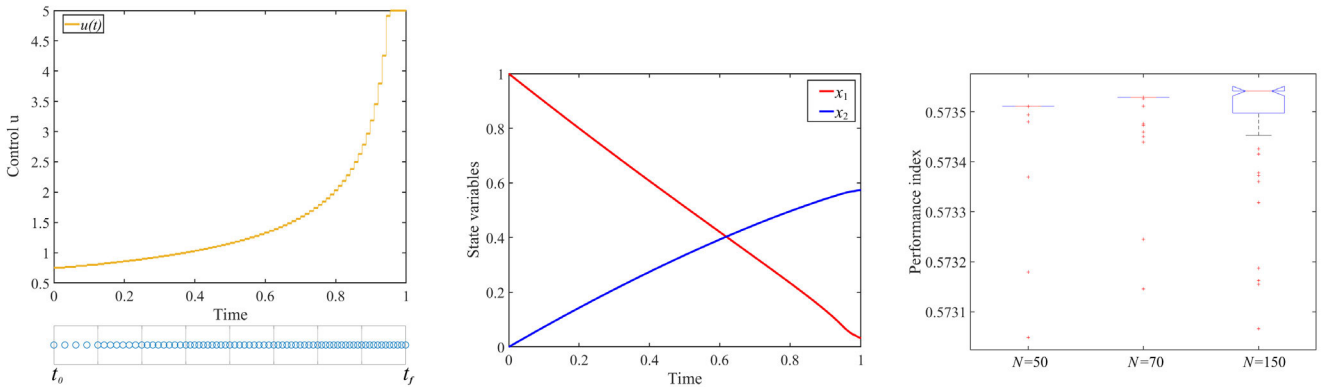
For problem 4, the time horizon $[0, 5]$ is divided into $N = 25, 45$ time subintervals; and OCVP-GSOA is run 50 times independently. Table 9 lists the parameters of OCVP-GSOA for problem 4. Table 10 presents results obtained by OCVP-GSOA and other methods. Reddy and

TABLE 9. Parameters of OCVP-GSOA for Problem 4.

parameter	value
population size pop	50
mutation times $\text{Mutation}_{\text{times}}$	100
maximum number of iterations $\text{Max}_{\text{iteration}}$	200

Husain used the conjugate gradient method (CGM) to obtain a value of 0.7227 [42]. Ko and Stevens got a value of 0.7226 using the combined modes method (CMM) [43]. Zhang and Mo reached values of 0.7226987 ($N = 10$) and 0.7234724 ($N = 20$) using a modified sailfish optimizer (MSFO) and an equal division method [44]. Xu *et al.* obtained values of 0.7238900 ($N = 150$) and 0.7238990 ($N = 32$) using a uniform discretization CVP (UD-CVP) method and a CVP method based on pseudo Wigner-Ville (PWV-CVP) respectively [15].

From Table 10, when the number of time subintervals N is 25, OCVP-GSOA can obtain a value of 0.72361395, better than the best results reported in [42], [43], and [44]. When N is 45, the best result of OCVP-GSOA is 0.72374838, very



(a) Optimal control trajectory and time grid ($N = 70$)

(b) State variables

(c) Box plot

FIGURE 8. Results obtained by OCVP-GSOA. (Problem 3.)

TABLE 10. Simulation results for Problem 4.

	methods	N	best	mean	worst	std	time/s
Problem 4 (max)	OCVP-GSOA	25	0.72361395	0.72361343	0.72361181	4.08383953E-07	71.69
		45	0.72374838	0.72374781	0.72374575	4.41232205E-07	212.15
	CGM [42]	-	0.7227	-	-	-	-
	CMM [43]	-	0.7226	-	-	-	-
	MSFO [44]	10	0.7226987	-	-	-	-
		20	0.7234724	-	-	-	-
	UD-CVP [15]	150	0.7238900	-	-	-	2499.6
	PWV-CVP [15]	32	0.7238990	-	-	-	121.9

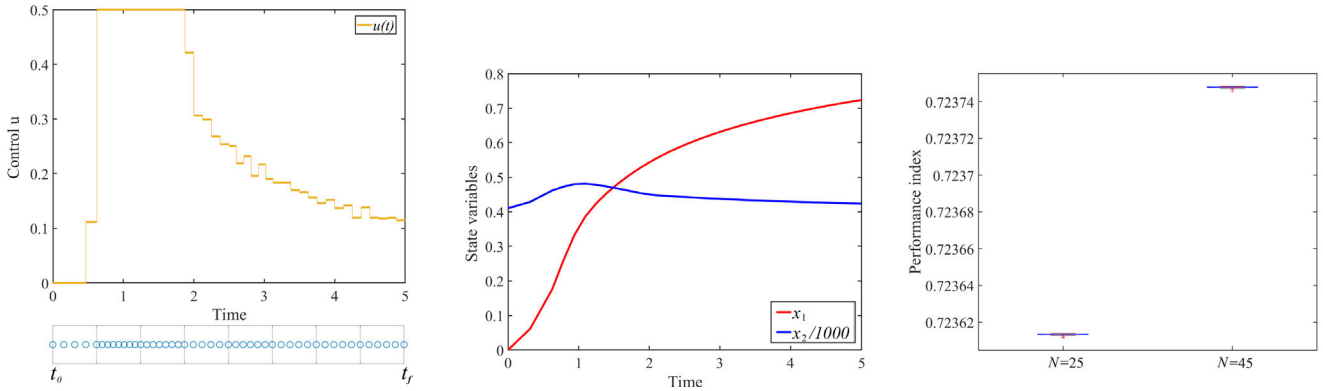
close to the best result 0.7238990 reported in [15]. Figure 9(c) shows the box plot of OCVP-GSOA of the different numbers of time subintervals (Problem 4). As can be seen, OCVP-GSOA also has excellent stability in solving problem 4. Figure 9(a) shows the optimal control trajectory and time grid obtained by OCVP-GSOA when $N = 45$. The optimal control trajectory of this problem has two switch points. OCVP constructs a relatively fine time grid near these two switch points. The state variables of this problem are plotted in Figure 9(b).

From the above analyses for four problems, it can be concluded that OCVP has adaptability and can construct a reasonable time grid. OCVP-GSOA can achieve similar or higher solution accuracy compared with other methods. The control trajectory obtained by OCVP-GSOA has a good continuous smooth shape, which is more in line with the characteristics of the chemical dynamic process. On balance, OCVP-GSOA is an excellent optimization method of chemical DOP.

5) FURTHER DISCUSSIONS

In this subsection, simulation experiments are carried out to compare OCVP and CVP, compare GSOA and SOA, and compare GSOA and other meta-heuristic algorithms. Three well-known meta-heuristic algorithms are chosen for comparison. These are Arithmetic Optimization Algorithm (AOA) [45], Whale Optimization Algorithm (WOA) [46], and Particle Swarm Optimization (PSO) [47].

For problem 1, problem 2, problem 3, and problem 4, the number of time subintervals N is set as 50, 50, 50, and 45, respectively. CVP and GSOA are combined to form CVP-GSOA. For each problem, the parameters of CVP-GSOA are the same as those of OCVP-GSOA; CVP-GSOA and OCVP-GSOA are run 50 times independently. The simulation results obtained by OCVP-GSOA and CVP-GSOA are presented in Table 11. In problem 1, when N is 50, the best (0.61076309) and mean (0.61075373) results of OCVP-GSOA are better than the best (0.61070776) and mean (0.61070755) results of CVP-GSOA, respectively. In problem 2, the worst result (0.47714685) of OCVP-GSOA when N is 50 is better than the best result (0.47631338) of CVP-GSOA when N is 50. In problem 3, when N is 50, the best (0.57351086), mean (0.57349120), and worst (0.57304872) results of OCVP-GSOA are better than the best (0.57349880), mean (0.57346547), and worst (0.57253734) results of CVP-GSOA, respectively. In problem 4, when N is 45, the best (0.72374838), mean (0.72374781), and worst (0.72374575) results of OCVP-GSOA are very close to the best (0.72383440), mean (0.72383400), and worst (0.72383292) results of CVP-GSOA, respectively. From



(a) Optimal control trajectory and time grid ($N = 45$) (b) State variables (c) Box plot

FIGURE 9. Results obtained by OCVP-GSOA. (Problem 4.)

TABLE 11. Comparison of OCVP-GSOA, CVP-GSOA, OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO.

	methods	N	best	mean	worst	std	time/s
Problem 1 (max)	OCVP-GSOA	50	**0.61076309**[b]	**0.61075373**	0.61060504	3.38710530E-05	4.01
	CVP-GSOA	50	0.61070776	0.61070755	**0.61069702**	**1.50339737E-06**	3.90
	OCVP-SOA	50	0.58635172	0.55587852	0.53551746	1.07381510E-02	0.15
	OCVP-AOA	50	0.58016791	0.55893425	0.54382767	9.92738918E-03	0.15
	OCVP-WOA	50	0.60481029	0.60024115	0.59487724	2.40979596E-03	0.14
	OCVP-PSO	50	0.60680189	0.60472517	0.60163765	9.49772283E-04	0.15
Problem 2 (max)	OCVP-GSOA	50	**0.47714685**	**0.47714685**	**0.47714685**	**1.79304095E-10**	7.51
	CVP-GSOA	50	0.47631338	0.47631338	0.47631338	2.87466455E-10	7.41
	OCVP-SOA	50	0.46930932	0.45185832	0.40495190	1.62197031E-02	0.22
	OCVP-AOA	50	0.46281730	0.45497584	0.44587733	3.92015148E-03	0.22
	OCVP-WOA	50	0.46799052	0.46075619	0.45197030	3.76205325E-03	0.21
	OCVP-PSO	50	0.45026957	0.43842136	0.42194480	7.40374724E-03	0.21
Problem 3 (max)	OCVP-GSOA	50	**0.57351086**	**0.57349120**	**0.57304872**	**8.06832053E-05**	3.90
	CVP-GSOA	50	0.57349880	0.57346547	0.57253734	1.50571826E-04	3.75
	OCVP-SOA	50	0.52506021	0.47998735	0.42154454	2.74977888E-02	0.15
	OCVP-AOA	50	0.49725411	0.47964386	0.46539427	7.76331096E-03	0.15
	OCVP-WOA	50	0.53674606	0.51607454	0.48404753	1.39856311E-02	0.14
	OCVP-PSO	50	0.49390280	0.47262617	0.44264568	1.15184903E-02	0.15
Problem 4 (max)	OCVP-GSOA	45	0.72374838	0.72374781	0.72374575	4.41232205E-07	212.15
	CVP-GSOA	45	**0.72383440**	**0.72383400**	**0.72383292**	**3.14525661E-07**	204.19
	OCVP-SOA	45	0.70853981	0.70305537	0.69613805	3.07525819E-03	2.15
	OCVP-AOA	45	0.72348740	0.72275368	0.72106937	5.24050365E-04	2.12
	OCVP-WOA	45	0.71943620	0.71250247	0.70290817	3.71196021E-03	2.05
	OCVP-PSO	45	0.72291988	0.72052589	0.71741381	1.49704692E-03	2.12

[b] The bold symbol denotes best results.

these comparisons, it can be concluded that OCVP has certain advantages compared to CVP. In problem 1, problem 2, and problem 3, OCVP can obtain better results than CVP with the same number of time subintervals. In problem 4, OCVP is slightly worse than CVP, but the results obtained by OCVP are also competitive.

SOA, AOA, WOA, and PSO combine with OCVP to form OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO, respectively. Table 12 lists the parameter values for AOA, WOA, and PSO. For each problem, OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO have the same population size and maximum number of iterations as OCVP-GSOA; OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO are run 50 times independently. The simulation results obtained by these methods are presented in Table 11. In problem 1, when N is 50, the worst result

TABLE 12. Parameter values for AOA, WOA, and PSO.

algorithm	parameter	value
AOA	α	5
	μ	0.5
	MOA	Linear increase from 0.2 to 0.9
WOA	$\vec{a}$	Linear reduction from 2 to 0
	b	1
PSO	Inertia weight	0.9
	c_1	2
	c_2	2
	Velocity limit	[-20,20]

(0.61060504) of OCVP-GSOA is better than the best results (0.58635172, 0.58016791, 0.60481029, and 0.60680189) of OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO. In problem 2, when N is 50, the worst result (0.47714685) of OCVP-GSOA is better than the best results (0.46930932, 0.46281730, 0.46799052, and 0.45026957) of OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO. In problem 3, the worst result (0.57304872) of OCVP-GSOA when N is 50 is better than the best results (0.52506021, 0.49725411, 0.53674606, and 0.49390280) of OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO when$N$is 50. In problem 4, when N is 45, the worst result (0.72374575) of OCVP-GSOA is better than the best results (0.70853981, 0.72348740, 0.71943620, and 0.72291988) of OCVP-SOA, OCVP-AOA, OCVP-WOA, and OCVP-PSO. From these comparisons, it is clear that GSOA can consistently obtain better results than SOA, AOA, WOA, and PSO. Compared with SOA, AOA, WOA, and PSO, GSOA has obvious advantages in solving chemical DOP.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel nonuniform control vector parameterization method OCVP and an improved seagull optimization algorithm GSOA. By integrating OCVP into GSOA, a new optimization method, named OCVP-GSOA, can be formed. OCVP uses orthogonal experimental design to analyze the dynamic model to obtain the range of each factor. The range of each factor can measure the fluctuation amplitude of the optimal control trajectory. Then, under the guidance of the ranges, a reasonable time grid can be constructed. GSOA cleverly uses Gaussian distribution to generate the initial population that conforms to the chemical process and performs Gaussian mutation on the best search agent dimension by dimension to improve its ability to solve chemical DOPs. By application in four classic DOPs, the simulation results show that OCVP has adaptability and can construct a reasonable time grid. OCVP-GSOA can achieve similar or higher solution accuracy compared with other methods. Further comparison results show OCVP can achieve higher solution accuracy compared with CVP in most cases. GSOA has obvious advantages in solving chemical DOP compared with other meta-heuristic algorithms. On balance, OCVP-GSOA is an excellent optimization method.

For future works, OCVP that can handle dynamic optimization problem with multiple control variables can be developed. For more complex dynamic optimization problems, this can provide a better optimization method. And it would be also interesting to integrate OCVP into other intelligent optimization algorithm.

REFERENCES

[1] P. Tian, X. Chen, W. Zhao, and W. Du, "Dynamic optimization of chemical processes using symbiotic organisms search algorithm," in *Proc. Chin. Autom. Congr. (CAC)*, Hangzhou, China, Nov. 2019, pp. 1052–1058, doi: 10.1109/CAC48633.2019.8996822.

[2] J. Zhu, X. Yan, and W. Zhao, "Chemical process dynamic optimization based on the differential evolution algorithm with an adaptive scheduling mutation strategy," *Eng. Optim.*, vol. 45, no. 10, pp. 1205–1221, Oct. 2013.

[3] R. Luus, "Optimization of fed-batch fermentors by iterative dynamic programming," *Biotechnol. Bioeng.*, vol. 41, no. 5, pp. 599–602, Mar. 1993.

[4] G. P. Pollard and R. W. H. Sargent, "Off line computation of optimum controls for a plate distillation column," *Automatica*, vol. 6, no. 1, pp. 59–76, Jan. 1970.

[5] V. Istratie, "Optimal entry into atmosphere with minimum heat and constraints," in *Proc. AIAA Atmos. Flight Mech. Conf. Exhibit*, Providence, RI, USA, Aug. 2004, p. 5282, doi: 10.2514/6.2004-5282.

[6] A. Woinaroschy, I. D. Ofiteru, and A. Nica, "Optimal control of fed-batch bioreactors by iterative dynamic programming," *Control*, vol. 10, pp. 89–94, May 2010.

[7] R. Sundaralingam, "Two-step method for dynamic optimization of inequality state constrained systems using iterative dynamic programming," *Ind. Eng. Chem. Res.*, vol. 54, no. 31, pp. 7658–7667, Aug. 2015.

[8] X. Chen, C. Mei, B. Xu, Y. Ding, and G. Liu, "Biogeography-based learning particle swarm optimization method for solving dynamic optimization problems in chemical processes," *CIESC J.*, vol. 68, no. 8, pp. 3161–3167, 2017, doi: 10.11949/j.issn.0438-1157.20161786.

[9] Y. Zhou, C. Zhao, and X. Liu, "An iteratively adaptive particle swarm optimization approach for solving chemical dynamic optimization problems," *CIESC J.*, vol. 65, no. 4, pp. 1296–1302, 2014, doi: 10.3969/j.issn.0438-1157.2014.04.020.

[10] P. Zhang, X. Liu, and L. Ma, "Optimal control vector parameterization approach with a hybrid intelligent algorithm for nonlinear chemical dynamic optimization problems," *Chem. Eng. Technol.*, vol. 38, no. 11, pp. 2067–2078, Nov. 2015.

[11] J. Tian, P. Zhang, Y. Wang, X. Liu, C. Yang, J. Lu, W. Gui, and Y. Sun, "Control vector parameterization-based adaptive invasive weed optimization for dynamic processes," *Chem. Eng. Technol.*, vol. 41, no. 5, pp. 964–974, May 2018.

[12] K. L. Teo, L. S. Jennings, H. W. J. Lee, and V. Rehbock, "The control parameterization enhancing transform for constrained optimal control problems," *J. Austral. Math. Soc. B, Appl. Math.*, vol. 40, no. 3, pp. 314–335, Jan. 1999.

[13] T. Binder, A. Cruse, C. A. C. Villar, and W. Marquardt, "Dynamic optimization using a wavelet based adaptive control vector parameterization strategy," *Comput. Chem. Eng.*, vol. 24, nos. 2–7, pp. 1201–1207, Jul. 2000.

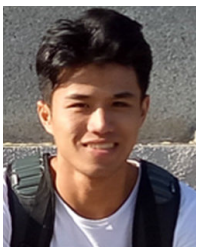
[14] G. Li, P. Liu, and X. Liu, "A control parameterization approach with variable time nodes for optimal control problems," *Asian J. Control*, vol. 18, no. 3, pp. 976–984, May 2016.

[15] W. F. Xu, A. P. Jiang, H. K. Wang, E. H. Jiang, Q. Ding, and H. H. Gao, "A grid reconstruction strategy based on pseudo Wigner–Ville analysis for dynamic optimization problem," *CIESC J.*, vol. 70, no. S1, pp. 158–167, 2019.

[16] F. Sun, W. Du, R. Qi, F. Qian, and W. Zhong, "A hybrid improved genetic algorithm and its application in dynamic optimization problems of chemical processes," *Chin. J. Chem. Eng.*, vol. 21, no. 2, pp. 144–154, Feb. 2013.

[17] X. Chen, W. Du, R. Qi, F. Qian, and H. Tianfield, "Hybrid gradient particle swarm optimization for dynamic optimization problems of chemical processes," *Asia–Pacific J. Chem. Eng.*, vol. 8, no. 5, pp. 708–720, Sep. 2013.

[18] Q. Jiang, L. Wang, Y. Lin, X. Hei, G. Yu, and X. Lu, "An efficient multi-objective artificial raindrop algorithm and its application to dynamic optimization problems in chemical processes," *Appl. Soft Comput.*, vol. 58, pp. 354–377, Sep. 2017.
[19] M. F. Tabassum, M. Saeed, A. Akgül, M. Farman, and S. Akram, "Solution of chemical dynamic optimization systems using novel differential gradient evolution algorithm," *Phys. Scripta*, vol. 96, no. 3, Mar. 2021, Art. no. 035212.
[20] B. Xu, Y. Zhang, D. Gong, Y. Guo, and M. Rong, "Environment sensitivity-based cooperative co-evolutionary algorithms for dynamic multi-objective optimization," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 15, no. 6, pp. 1877–1890, Nov./Dec. 2018.
[21] L. Fang, S. Ding, J. H. Park, and L. Ma, "Adaptive fuzzy control for stochastic high-order nonlinear systems with output constraints," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 9, pp. 2635–2646, Sep. 2021.
[22] L. Fang, S. Ding, J. H. Park, and L. Ma, "Adaptive fuzzy control for non-triangular stochastic high-order nonlinear systems subject to asymmetric output constraints," *IEEE Trans. Cybern.*, vol. 52, no. 2, pp. 1280–1291, Feb. 2022.
[23] G. Dhiman and V. Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," *Knowl.-Based Syst.*, vol. 165, pp. 169–196, Feb. 2019.
[24] Y. Cao, Y. Li, G. Zhang, K. Jermsittiparsert, and N. Razmjooy, "Experimental modeling of PEM fuel cells using a new improved seagull optimization algorithm," *Energy Rep.*, vol. 5, pp. 1616–1625, Nov. 2019.
[25] H. Jia, Z. Xing, and W. Song, "A new hybrid seagull optimization algorithm for feature selection," *IEEE Access*, vol. 7, pp. 49614–49631, 2019.
[26] H. Jiang, Y. Yang, W. Ping, and Y. Dong, "A novel hybrid classification method based on the opposition-based seagull optimization algorithm," *IEEE Access*, vol. 8, pp. 100778–100790, 2020.
[27] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[28] Z. Liu, W. L. Du, R. Qi, and F. Qian, "Dynamic optimization in chemical processes using improved knowledge-based cultural algorithm," *CIESC J.*, vol. 61, no. 11, pp. 2889–2895, Nov. 2010.
[29] X. Peng, R. Qi, W. Du, and F. Qian, "An improved knowledge evolution algorithm and its application to chemical process dynamic optimization," *CIESC J.*, vol. 63, no. 3, pp. 841–850, Mar. 2012, doi: 10.3969/j.issn.0438-1157.2012.03.024.
[30] Z.-K. Feng, W.-J. Niu, S. Liu, B. Luo, S.-M. Miao, and K. Liu, "Multiple hydropower reservoirs operation optimization by adaptive mutation sine cosine algorithm based on neighborhood search and simplex search strategies," *J. Hydrol.*, vol. 590, Nov. 2020, Art. no. 125223.
[31] Y. Zhang, G. Cui, J. Wu, W.-T. Pan, and Q. He, "A novel multi-scale cooperative mutation fruit fly optimization algorithm," *Knowl.-Based Syst.*, vol. 114, pp. 24–35, Dec. 2016.
[32] Y. Feng, J. Yang, C. Wu, M. Lu, and X.-J. Zhao, "Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation," *Memetic Comput.*, vol. 10, no. 2, pp. 135–150, Jun. 2018.
[33] B. Shi, Y. Yin, and F. Liu, "Optimal control strategies combined with PSO and control vector parameterization for batchwise chemical process," *CIESC J.*, vol. 70, no. 3, pp. 979–986, 2019, doi: 10.11949/j.issn.0438-1157.20181140.
[34] L. Xu, Y. Mo, Y. Lu, and J. Li, "Improved seagull optimization algorithm combined with an unequal division method to solve dynamic optimization problems," *Processes*, vol. 9, no. 6, Jun. 2021, Art. no. 1037.
[35] Y. Zhou and X. Liu, "Control parameterization-based adaptive particle swarm approach for solving chemical dynamic optimization problems," *Chem. Eng. Technol.*, vol. 37, no. 4, pp. 692–702, Apr. 2014.
[36] S. A. Dadebo and K. B. Mcauley, "Dynamic optimization of constrained chemical engineering problems using dynamic programming," *Comput. Chem. Eng.*, vol. 19, no. 5, pp. 513–525, May 1995.
[37] D. T. Pham, Q. T. Pham, A. Ghanbarzadeh, and M. Castellani, "Dynamic optimisation of chemical engineering processes using the bees algorithm," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 6100–6105, 2008.
[38] M. Huang, X. Zhou, T. Huang, C. Yang, and W. Gui, "Dynamic optimization based on state transition algorithm for copper removal process," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 2827–2839, Jul. 2019.
[39] X. Chen, W. Du, H. Tianfield, R. Qi, W. He, and F. Qian, "Dynamic optimization of industrial processes with nonuniform discretization-based control vector parameterization," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 4, pp. 1289–1299, Oct. 2014.
[40] L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Comput. Chem. Eng.*, vol. 8, nos. 3–4, pp. 243–247, 1984.
[41] J. Rajesh, K. Gupta, H. S. Kusumakar, V. K. Jayaraman, and B. D. Kulkarni, "Dynamic optimization of chemical processes using ant colony framework," *Comput. Chem.*, vol. 25, no. 6, pp. 583–595, Nov. 2001.
[42] K. V. Reddy and A. Husain, "Computation of optimal control policy with singular subarc," *Can. J. Chem. Eng.*, vol. 59, no. 4, pp. 557–559, Aug. 1981.
[43] D. Y. C. Ko and W. F. Stevens, "Studies of singular solutions in dynamic optimization: II. Optimal singular design of a plug-flow tubular reactor," *AIChE J.*, vol. 17, no. 1, pp. 160–166, Jan. 1971.
[44] Y. Zhang and Y. Mo, "Dynamic optimization of chemical processes based on modified sailfish optimizer combined with an equal division method," *Processes*, vol. 9, no. 10, Oct. 2021, Art. no. 1806.
[45] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Comput. Methods Appl. Mech. Eng.*, vol. 376, Apr. 2021, Art. no. 113609.
[46] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.
[47] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, Perth, WA, Australia, Nov./Dec. 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.



HAIDONG GUO received the B.S. degree in electronic and information engineering from Ningde Normal University, Ningde, China, in 2020. He is currently pursuing the master's degree with the School of Artificial Intelligence, Guangxi Minzu University, Nanning, China. His current research interests include swarm intelligent optimization, numerical simulation of control systems, process dynamic optimization, and computational intelligence.

YUANBIN MO received the M.S. degree in mathematics from Guizhou University, Guiyang, China, in 2001, and the Ph.D. degree in control theory and control engineering from Zhejiang University, Hangzhou, China, in 2007. He is currently a Professor with Guangxi Minzu University. His research interests include computation intelligence, system optimization, and numerical simulation of control systems.

YUEDONG ZHANG was born in Shandong, China, in 1998. He received the B.S. degree in information and computing science from Qufu Normal University, in 2020. He is currently pursuing the M.Sc. degree with the College of Electronic Information, Guangxi Minzu University, Nanning, China. His research interests include intelligent information processing and process dynamic optimization.