MULTI-VIEW PARALLEL COOPERATIVE OPTIMIZATION FOR DYNAMIC LAUNCH VEHICLE ATTITUDE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic Launch Vehicle Attitude Control (DLVAC) involves designing correction networks across dynamic flight stages. In practice, this poses two key challenges: (i) exploring strongly constrained design spaces and (ii) transferring reusable design knowledge. Several Dynamic Optimization Algorithms (DOAs) can address strongly constrained problems to some extent. However, due to the need for numerous iterations and their tendency to become trapped in local optima, they are somewhat inadequate when addressing strongly constrained optimization problems. To address the limitations of existing works, we propose a novel approach called Multi-View Parallel Cooperative Optimization (MPCO). MPCO constructs multiple subprocesses to carry out the optimization in parallel, enhancing optimization efficiency. In the iterative process, subprocesses gather problem-relevant information from multiple perspectives, adjusting search directions to enhance the exploratory capability of MPCO. Additionally, cooperation among subprocesses is facilitated through a knowledge transfer strategy, which helps reduce the issue of becoming trapped in local optima. Using a set of common benchmark functions and a set of practical launch vehicle control parameter design tasks, we compare the performance of MPCO with nine advanced algorithms. MPCO achieves superior and feasible solutions in less time, demonstrating excellent efficiency and stability.

1 INTRODUCTION

Dynamic Launch Vehicle Attitude Control (DLVAC) plays a critical role in ensuring flight stability and accurate trajectory tracking (Benedikter et al., 2022; Zhou et al., 2025; Pang et al., 2025; Yu et al., 2024). As illustrated in Figure 1, a launch vehicle flight process consists of multiple continuous flight stages, each represented by a set of transfer functions. The vehicle dynamics evolve substantially due to changes in mass properties and aerodynamic conditions. Consequently, a correction network designed for one flight stage may no longer remain effective at subsequent stages, making continuous adaptation of the control system indispensable.

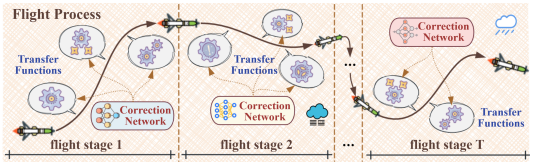


Figure 1: Illustration of DLVAC. DLVAC seeks correction networks that simultaneously optimizes all transfer functions at each flight stage.

While Dynamic Optimization Algorithms (DOAs) are widely used to solve constrained problems (Chung et al., 2008; Li & Li, 2020; Bian et al., 2018; 2019), they have limitations. One prominent challenge is their susceptibility to local optima. DOAs tend to explore solution spaces by iteratively improving candidate solutions. However, this iterative process can converge to infeasible or local-optimal solutions when faced with highly complex, constrained problems. Additionally, the iterative nature of DOAs often necessitates a large number of iterations to achieve satisfactory solutions. This requirement for numerous iterations can result in substantial time costs, making the optimization process inefficient, especially in real-world scenarios where time is a critical factor.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In response to these drawbacks, several improvements have been proposed. Firstly, to avoid getting stuck in local optima, DOAs have suggested various enhancements, including dynamically adjusting internal parameters based on iterative progress (Shi & Eberhart, 1998), integrating additional global and local search strategies (Mousa et al., 2012), and adopting hybrid optimization methods (Xia et al., 2018). Although these facilitate exploration in different directions and spaces, they often require a certain scale of population or number of iterations, potentially increasing the time cost.

Solving strongly constrained problems typically requires long iterative solving processes accompanied by a large number of evaluations. Particularly when conducting expensive evaluations, it can significantly increase solution time. To mitigate this issue, researchers have proposed the surrogate-assisted evolutionary algorithm (SAEA), which uses inexpensive surrogate models to replace costly function evaluations (Qian et al., 2022; Hao et al., 2024). However, surrogate models may face accuracy issues due to their reliance on the quality and quantity of training sample data, posing challenges to generalization. Therefore, relying solely on surrogate models still cannot fully address time-consuming optimization problems, highlighting the urgent need for DOAs to efficiently solve strongly constrained optimization problems. While DOAs provide valuable methods for solving constrained problems, their sensitivity to local optima and time-consuming nature underscore the need for further technological advancements to enhance their effectiveness and efficiency.

To overcome the limitations of existing methods, we propose a novel approach called Multi-View Parallel Collaborative Optimization (MPCO). MPCO establishes multiple parallel subprocesses using different optimizers. A key feature of MPCO is its utilization of multiple views during iterations to adjust the search direction, thereby enabling a comprehensive understanding of the problem. Although the parallel subprocesses operate independently, MPCO employs knowledge transfer to facilitate collaboration among the subprocesses, sharing experiences to cooperatively search for the global optimum. MPCO offers several advantages. (i) The parallel processing in MPCO accelerates the search speed and reduces the overall optimization time. (ii) The use of multiple views increases the chances of finding the global optimum from different search perspectives. (iii) The knowledge transfer collaboration among subprocesses helps to accelerate convergence to better solutions. By introducing MPCO as an innovative approach, our aim is to solve highly constrained problems more quickly (particularly those that are time-consuming) while enhancing the performance of the final optimization results. Our main contributions can be summarized as follows:

- We propose the MPCO framework to efficiently address the strongly constrained characteristic of dynamic launch vehicle attitude control.
- MPCO exhibits flexibility with various optional optimizers, while the multi-coordinate view and knowledge transfer strategies provide MPCO with strong versatility, enabling it to adapt to various scenarios and explore more valuable solution spaces.
- Through comparative analyses with several classical DOAs, involving a set of benchmark test functions and practical parameter design tasks in the launch vehicle attitude control system, our method demonstrates an exceptional ability to identify feasible solutions more quickly, ultimately enhancing the efficiency of solving such problems.

2 PRELIMINARIES

Dynamic Flight Process. The dynamic flight process $\{\mathcal{E}^t\}_{t=1}^{T}$ involves a sequence of T flight stages. In each stage $\mathcal{E}^t$, the launch vehicle state is represented by a set of transfer functions $\mathcal{G}^t = \{G_k^t(s)\}_{k=1}^{K_t}$, s represents the complex frequency-domain variable and K_t is the number of transfer functions in stage t. Both K_t and the settings of $G_k^t(s)$ may vary over time, for the k-th transfer function in stage t, its general form is defined as

$$G_k^t(s) = \kappa_k^t \cdot \frac{\prod_{a=1}^{N_{1,k}^t}(O_{a,k}^t s+1)\prod_{b=1}^{N_{2,k}^t}\left[\left(\frac{s}{\omega_{b,k}^t}\right)^2+\frac{2\xi_{b,k}^t}{\omega_{b,k}^t}s+1\right]}{\prod_{c=1}^{N_{3,k}^t}(O_{c,k}^t s+1)\prod_{d=1}^{N_{4,k}^t}\left[\left(\frac{s}{\omega_{d,k}^t}\right)^2+\frac{2\xi_{d,k}^t}{\omega_{d,k}^t}s+1\right]}, \tag{1}$$

where κ defines the gain of the transfer function, $N_{1,k}^t$, $N_{2,k}^t$, $N_{3,k}^t$, and $N_{4,k}^t$ represent the numbers of first-order numerator terms, second-order numerator terms, first-order denominator terms, and

second-order denominator terms. $O^t_{a,k}$ and $O^t_{c,k}$ are the time variation, ($\omega_{b,k}$,$\omega_{d,k}$) and ($\xi_{b,k}$,$\xi_{d,k}$) are the frequency and damping ratios of the second-order differential and oscillation element.

Variable-Structure Correction Network. The structure of the correction network $\mathbf{x}^t$ at the t-th stage follows the identical form of the transfer functions. However, the correction network can adopt variable-structures, let N_1, N_2, N_3, and N_4 denote the number of numerator and denominator terms in $G^t_k(s)$, the structure of $\mathbf{x}^t$ is defined as $D_{\mathbf{x}^t} = N_1 + 2N_2 + N_3 + 2N_4$, each second-order term contributes two parameters and the structural orders are bounded by domain expert knowledge as $1 \le N_r \le N_{\text{exp}}$, $r \in \{1, 2, 3, 4\}$. For the t-th stage, DLVAC searches for a control parameter $\mathbf{x}^t \in \mathbb{R}^{D_{\mathbf{x}^t}}$ with a variable-structure to regulate all the transfer functions in $\mathcal{G}^t$. In practice, the frequency-domain behavior of each transfer function $G^t_k(s)$ is characterized by its Bode plot, comprising the magnitude response $A^t_k(\omega)$ and phase response $\varphi^t_k(\omega)$ as functions of frequency:

$$\begin{cases} A^t_k(\omega) = 20\lg\left|\kappa^t_k\right| + \sum_{a=1}^{N^t_{1,k}} 20\lg\sqrt{(O^t_{a,k}\omega)^2+1} + \sum_{b=1}^{N^t_{2,k}} 20\lg\sqrt{f^{A,t}_{b,k}(\omega)} - \sum_{c=1}^{N^t_{3,k}} 20\lg\sqrt{(O^t_{c,k}\omega)^2+1} - \sum_{d=1}^{N^t_{4,k}} 20\lg\sqrt{f^{A,t}_{d,k}(\omega)}, \\ \varphi^t_k(\omega) = \sum_{a=1}^{N^t_{1,k}} \tan^{-1}(O^t_{a,k}\omega) + \sum_{b=1}^{N^t_{2,k}} \tan^{-1} f^{\varphi,t}_{b,k}(\omega) - \sum_{c=1}^{N^t_{3,k}} \tan^{-1}(O^t_{c,k}\omega) - \sum_{d=1}^{N^t_{4,k}} \tan^{-1} f^{\varphi,t}_{d,k}(\omega). \end{cases} \tag{2}$$

Here, $f^{A,t}_{b,k}(\omega)$ and $f^{A,t}_{d,k}(\omega)$ denote the magnitude-related terms of the numerator- and denominator-side second-order factors, while $f^{\varphi,t}_{b,k}(\omega)$ and $f^{\varphi,t}_{d,k}(\omega)$ denote their corresponding phase-related terms. Based on the Bode plots of $\mathcal{G}^t$, we extract the flight attitude indicators and formulate them as multiple objectives to be optimized. The detailed rules for indicators are presented in **Appendix A**. The unified maximization objective vector for the k-th transfer function $G^t_k(s)$ is defined as

$$\mathbf{f}^t_k(\mathbf{x}^t) = \left[\omega^t_{0,k}, A^t_{d,k}, -A^t_{g,k}, -A^t_{h,k}, -A^t_{e,k}, \varphi^t_{d,k}, \varphi^t_{h,k}, \varphi^t_{e,k}\right]^\top. \tag{3}$$

Task Formulation. We define the objectives for each stage as the worst objective value over the transfer function set, each of the eight objectives is aggregated at the stage level as $F^t_j(\mathbf{x}^t) = \min_{1\le k\le K_t} f^t_{j,k}(\mathbf{x}^t)$, where $f^t_{j,k}(\mathbf{x}^t)$ denotes the j-th component of $\mathbf{f}^t_k(\mathbf{x}^t)$ and the aggregation guides the control parameter to improve the most difficult transfer function for each objective. The DLVAC task is then formulated to search for a sequence of control parameters $\{\mathbf{x}^t\}^T_{t=1}$:

$$\max_{\mathbf{x}^t\in\mathbb{R}^{D_{\mathbf{x}^t}}} \{\mathcal{F}^t(\mathbf{x}^t)\}^T_{t=1} = \{F^t_1(\mathbf{x}^t), F^t_2(\mathbf{x}^t), \ldots, F^t_8(\mathbf{x}^t)\}^T_{t=1}. \tag{4}$$

We highlight two key characteristics of DLVAC task: *(i)* the correction network $\mathbf{x}^t$ as the decision vector, has a dimension $D_{\mathbf{x}^t}$ that varies with its parameter structure; *(ii)* each objective $F^t_j(\mathbf{x}^t)$ is constructed by aggregating the same flight attitude indicators across multiple transfer functions.

3 Framework of MPCO

Aiming at solving strongly constrained optimization problems with higher efficiency, we propose a novel framework named Multi-view Parallel Cooperative Optimization (MPCO). The whole framework is depicted in Fig. 2. MPCO employs a parallel cooperative framework where multiple sub-processes concurrently execute optimization tasks using various selected base optimizers. Leveraging the parallel model, sufficient population information can be gathered to establish a robust Eigen coordinate view. As for utilizing multi-coordinate views, we propose a strategy based on the directional difference of the offsets between the old and new positions, which can effectively select new positions without increasing evaluation costs. Finally, we introduced a knowledge transfer strategy to help individual threads escape local optima from an external perspective, accelerating convergence to better solutions. The entire procedure of MPCO is illustrated in Algorithm 1.

MPCO consists of four components: ***(i) initialization for sub-processes***; ***(ii) generating the multi-coordinate views***; ***(iii) Iterative parallel search with multi-views;*** and ***(iv) knowledge transfer among the sub-processes***. Subsequently, we will elaborate on these components.

Initialization for sub-processes. Ensuring non-overlapping individual positions across sub-processes enhances solution space exploration. When sub-processes initialize independently, positions may overlap. Therefore, the main process handles initialization to evenly distribute individuals among sub-processes. We utilize orthogonal initialization via Latin Hypercube Sampling

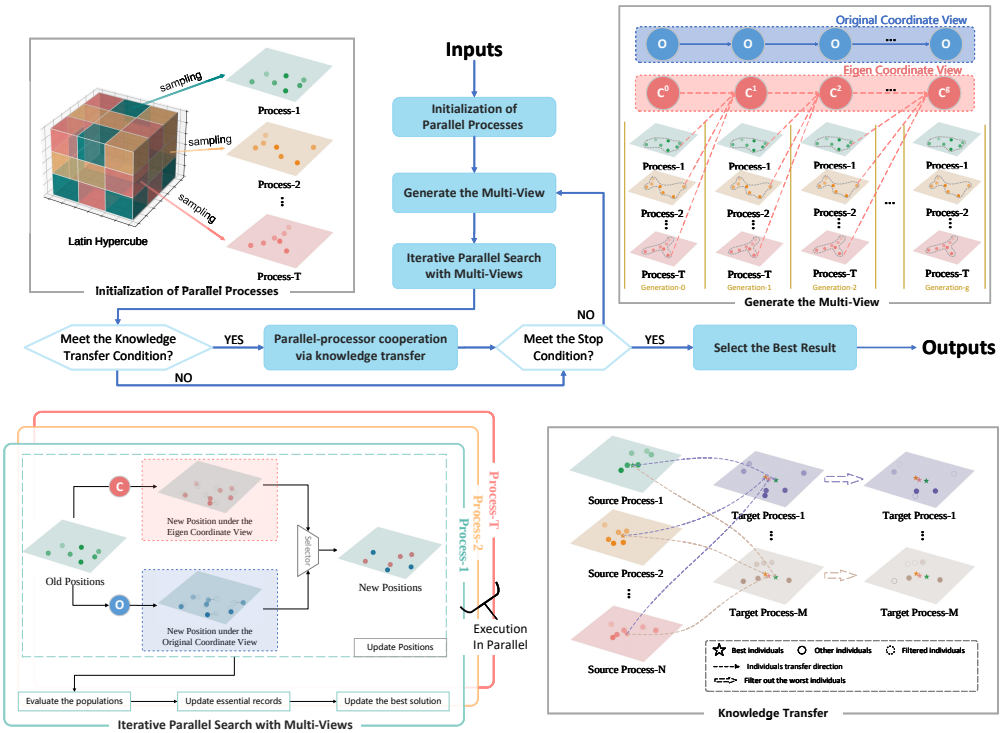


Figure 2: The framework of MPCO, in which various processes parallelly search the solution space with different optimizers and the information of the solution space is extracted from both the original coordinate view **O** and the Eigen coordinate view (decomposed from the covariance matrix **C**).

(LHS) (Stein, 1987), ensuring uniform distribution across dimensions and improving search space coverage. Algorithm 2 illustrates the orthogonal population initialization procedure. In the context, $NP, NS, D,$ **lD**, **uD** correspond to the definitions in Algorithm 1. λ denotes the total population pool size, **S** represents a Latin Hypercube matrix with NS rows and D columns, generated using an external library. Each element $s_{row,j}$ represents the j-th element of the row-th vector in **S**.

Generating the multi-coordinate views. During iterations, the original coordinate view remains static, whereas the Eigen coordinate view dynamically adjusts based on the covariance matrix $\mathbf{C}^g$. Typically, $\mathbf{C}^g$ is updated utilizing population information. The update strategy for $\mathbf{C}^g$ follows the rank-μ update policy, an effective technique for covariance matrix adjustment (Hansen, 2016). Initially, an estimator of the current population distribution $\mathbf{C}_\mu^{g+1}$ is derived as follows:

$$\mathbf{C}_\mu^{g+1} = \sum_{i=1}^{\mu} \omega_i (\overrightarrow{a_{i:\hat{\mathbb{P}}}} - \overrightarrow{m}^g)(\overrightarrow{a_{i:\hat{\mathbb{P}}}} - \overrightarrow{m}^g)^\top, \tag{5}$$

where $\omega_i = \frac{\ln(\mu+0.5)-\ln i}{\mu \ln(\mu+0.5)-\sum_{j=1}^{\mu} \ln j}$ and $\overrightarrow{m}^{g+1} = \sum_{i=1}^{\mu} \omega_i \overrightarrow{a_{i:\hat{\mathbb{P}}}}$. g denotes the current iteration, $\hat{\mathbb{P}}$ represents a population pool, and μ denotes selecting the top μ individuals from the entire population pool for the matrix update. ω_i stands for the weight coefficient, which decreases as the ranking of the individual decreases. $\overrightarrow{a_{i:\lambda}}$ represents the i-th-ranked individual in $\hat{\mathbb{P}}$, while $\overrightarrow{m}^g$ denotes the mean vector of the top μ individuals at g-th generation. Then, update the covariance matrix $\mathbf{C}^{g+1}$:

$$\mathbf{C}^{g+1} = (1 - c_\mu)\mathbf{C}^g + \frac{c_\mu}{(\sigma^g)^2}\mathbf{C}_\mu^{g+1}, \tag{6}$$

where $c_\mu \approx (1/3)\mu_{eff}/D^2$ denotes the learning rate, $\mu_{eff} = (\sum_{i=1}^{\mu} \omega_i{}^2)^{-1}$ denotes the variance effective selection mass, and σ^g denotes the step size. Specifically, the parameter μ is set to equal $(\lambda/3)$, and σ^g is set to be a constant 1. $\mathbf{C}^0$ is initialized as the identity matrix **I**, and $\overrightarrow{m}^0$ is initialized

as a random vector within the search space. After each round of sub-processes, the main process aggregates the current populations of all sub-processes to compose $\hat{\mathbb{P}}$.

Upon obtaining $\mathbf{C}^{g+1}$, conduct Eigen decomposition on the covariance matrix $\mathbf{C}$ to obtain its eigenvectors $\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^\top$. Here, $\mathbf{B}$ is an orthogonal matrix containing characteristic vectors of $\mathbf{C}$ as its columns, $\mathbf{B}^\top$ denotes the transpose of $\mathbf{B}$, and $\mathbf{D}$ is a diagonal matrix where each diagonal element represents the square root of an eigenvalue of $\mathbf{C}$. The columns of the orthogonal matrix $\mathbf{B}$, which are the eigenvectors of $\mathbf{C}$, constitute the basis of the Eigen coordinate view. Since eigenvectors maintain their direction under linear transformations, they offer insights into the correlation between different dimensions and the primary direction of variation, thereby providing reliable search directions.

Iterative parallel search with multi-views An iterative search process of an optimizer typically involves updating positions, evaluating individuals, maintaining records, and updating the best solution. In MCPO, subprocesses perform an iterative search in parallel, which closely mirrors the iterative search process of the basic optimizer, differing primarily in the position updates. The distinction lies in the utilization of multi-coordinate views, where individuals conduct the position update operators in both the original and Eigen coordinate views. This process generates two results, $\overrightarrow{x_o}$ and $\overrightarrow{x_e}$, from which individuals need to select one as their updated position.

We first introduce a generic paradigm illustrating how an optimizer utilizes the original coordinate view to execute the position update operator. In the original coordinate view, operators in most DOAs can be described as

$$\overrightarrow{r_O} = \sum_{i=1}^{m} \alpha_i \overrightarrow{y_i} + \sum_{i=1}^{n} \mathbf{W}_i \overrightarrow{z_i}, \tag{7}$$

where $\overrightarrow{r_O}$ represents the resultant vector, m, n are non-negative integers, α_i denotes a coefficient, and $\mathbf{W}_i = \text{diag}(w_1, w_2, \ldots, w_D)$ is a diagonal matrix where w_1, w_2, …, w_D are real numbers and D represents the dimensions of the problem. Additionally, $\overrightarrow{y_i}$ and $\overrightarrow{z_i}$ denote specific vectors. We then present a generic paradigm for executing operators in the Eigen coordinate view. In the Eigen coordinate view, operators can be described as follows:

$$\overrightarrow{r_E} = \mathbf{B}(\sum_{i=1}^{m} \alpha_i(\mathbf{B}^\top \overrightarrow{y_i}) + \sum_{i=1}^{n} \mathbf{W}_i(\mathbf{B}^\top \overrightarrow{z_i})) = \sum_{i=1}^{m} \alpha_i \overrightarrow{y_i} + \sum_{i=1}^{n} \mathbf{B}\mathbf{W}_i\mathbf{B}^\top \overrightarrow{z_i}, \tag{8}$$

where $\mathbf{B}$ is an orthogonal matrix derived from the eigendecomposition of $\mathbf{C}$, and $\mathbf{B}^\top$ is its transpose. Initially, vectors are transformed into the Eigen coordinate view using $\mathbf{B}^\top$. Operators are then applied to these transformed vectors as in the original manner. Finally, the vectors are converted back to the original views through $\mathbf{B}$ to conduct evaluation. It's evident that by substituting $\mathbf{W}_i$ with $\mathbf{B}\mathbf{W}_i\mathbf{B}^\top$ in the second part of Eq. (7), the operator can transition from the original coordinate view to the Eigen coordinate view.

As for selecting between the new positions of two operator computations, we propose a directional difference-based individual position strategy, as illustrated in Fig. 3. This strategy categorizes position selection into two cases based on θ, which signifies the angle between the offsets $\Delta\overrightarrow{x_o}$ and $\Delta\overrightarrow{x_e}$, where $\Delta\overrightarrow{x_o} = \overrightarrow{x_o} - \overrightarrow{x}$ and $\Delta\overrightarrow{x_e} = \overrightarrow{x_e} - \overrightarrow{x}$: if $\theta \leq \hat{\theta}$, select $\overrightarrow{x_o}$ as $\overrightarrow{x}_i^{g+1}$; if $\theta > \hat{\theta}$, select $\overrightarrow{x_e}$ as $\overrightarrow{x}_i^{g+1}$. Here, $\hat{\theta}$ represents the critical similarity threshold, set to 60°. This strategy is based on the following considerations: When the difference between the directions of two offsets is significant, priority is given to the new search direction provided by the Eigen coordinate view; conversely, the position from the original coordinate view is chosen to ensure a stable search iteration.

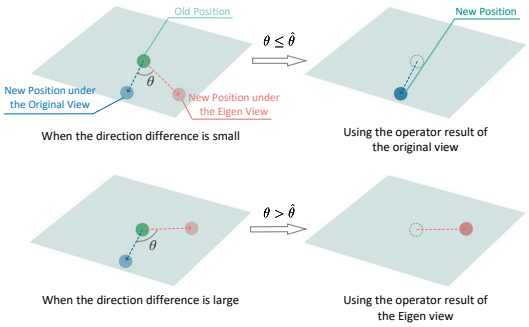


Figure 3: The directional difference-based individual position update strategy.

Knowledge transfer among the sub-processes. In general, optimizers employ various strategies to escape local optima, and MPCO is no exception. Unlike conventional optimizers, MPCO adopts

a knowledge transfer strategy external to the algorithm, leveraging superior sub-processes to assist inferior ones in escaping local optima, thereby helping the entire algorithm to converge to a better solution faster. The knowledge transfer is described in Algorithm 3. $\mathbb{S}$ represents the source domain set, $\mathbb{T}$ represents the target domain set, and *sub-process*$_t$ represents the t-th sub-process. Since sub-processes typically demonstrate varying performance levels during the search iterations, knowledge transfer involves migrating individuals from sub-processes with better performance to those with poorer performance, thereby enhancing the latter. The key aspects of knowledge transfer strategies include determining the timing of the transfer as well as identifying the target domains.

As for transfer timing, during the initial iterations, the search directions of each sub-process are uncertain, and the optimal solution does not offer adequate guidance. Therefore, MPCO sets only two transfer timings, occurring at 50% and 75% of the maximum iterations, respectively. As for the source domain and target domain, to mitigate the potential negative effects of transfer, the source and target domains are determined based on the status of feasible solutions in each process. In MPCO, the set of sub-processes that have obtained feasible solutions is identified as the source domain, as these processes demonstrate better convergence and performance.

4 Experimental Study

4.1 Experimental Setup

DLVAC Tasks and Test Functions. To validate the effectiveness of MPCO, we instantiated seven DLVAC tasks with different constraint configurations. As summarized in Table 1, these tasks cover different constraint types and exhibit diverse dynamic characteristics. For each task, the number of dynamic stages is set to $T = 10$, and the initial numbers of transfer functions are 25, 40, 25, 20, 36, 50, and 36, respectively. Notably, we report the feasible rate $\rho = |F|/|S|$ to quantify the sparsity, where $|F|$ is the number of feasible solutions and $|S| = 100{,}000$ is the total number of random samples. A value of $\rho = 0.000\%$ indicates that the feasible region occupies less than 0.001% of the sampled search space. Additionally, we selected test functions from CEC2006 (Liang et al., 2006) and CEC2010 (Mallipeddi & Suganthan, 2010) as summarized in Table 2.

Constraints	Task1	Task2	Task3	Task4	Task5	Task6	Task7
Cut-Off Freq $\hat{\omega}_0$	[0.1,10]	[0.1,6]	[0.4,10]	[0.01,5]	[0.01,15]	[2,9]	[0.5,10]
Low-Freq Amp $\hat{A}_d$	4.0	2.0	2.6	4.0	7.4	4.5	8.0
High-Freq Amp $\hat{A}_g$	-8.0	-10.0	-8.0	-5.5	-11.0	-4.0	-7.5
Oscillation Amp $\hat{A}_h$	—	-7.0	-10.0	-5.0	-6.3	-9.5	-5.0
Elastic Amp $\hat{A}_e$	—	—	—	—	—	-10.0	-7.8
Low-Freq Phase $\hat{\varphi}_d$	30.0	20.0	17.0	16.0	22.0	18.6	14.4
Oscillation Phase $\hat{\varphi}_h$	—	—	—	26.0	30.0	—	—
Elastic Phase $\hat{\varphi}_e$	—	—	20.0	—	16.0	—	24.3
Dimension $D_{\mathbf{x}}$	16	17	21	21	32	21	32
Feasible Rate ρ	0.002%	0.002%	0.001%	0.003%	0.000%	0.000%	0.000%

Table 1: Details of the DLVAC Tasks.

Experimental Settings. ***(i) Baseline Method***: The baseline methods comprise a selection of classical optimization algorithms, including PSO (Kennedy & Eberhart, 1995), FA (Yang & He, 2013), CS (Yang & Deb, 2014), GWO (Mirjalili et al., 2014), WOA (Mirjalili & Lewis, 2016), PSO-sono (Meng et al., 2022), HHO (Heidari et al., 2019), SCHO (Bai et al., 2023), and RIME (Su et al., 2023). Among them, PSO, FA, CS, GWO, and WOA serve as the base algorithms for MPCO. PSO-sono enhances PSO's global search capability by refining the iteration paradigm. Conversely, the remaining algorithms introduce novel operators tailored for multidimensional optimization problems, exhibiting commendable performance as well; ***(ii) Evaluation Metrics***: Avg success rate: Average success rate across all runs, Avg iterations: Average number of iterations to reach a feasible solution, Avg time: Average time to achieve a feasible solution, Avg fitness: Average fitness of

Name	D	Type of function	ρ	E	I	a
g01	13	quadratic	0.00111%	0	9	6
g03	10	polynomial	0.0000%	1	0	1
g06	2	cubic	0.0066%	0	2	2
g07	10	quadratic	0.0003%	0	8	6
g08	2	nonlinear	0.86%	0	2	0
g10	8	linear	0.0010%	0	6	6
c08	30	Non Separable	37.5278%	0	1	–
c13	30	Separable	0.0000%	0	3	–
c15	30	Non Separable	0.6023%	0	3	–
c17	30	Non Separable	0.0000%	1	2	–

Table 2: The details of the test functions.

	Method	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
Avg success rate ↑	PSO-sono	33.33%	0.00%	76.67%	0.00%	36.67%	6.67%	13.33%
	HHO	23.33%	3.33%	6.67%	3.33%	13.33%	13.33%	3.33%
	SCHO	46.67%	16.67%	63.33%	6.67%	83.33%	43.33%	43.33%
	RIME	33.33%	33.33%	66.67%	33.33%	66.67%	33.33%	66.67%
	MPCO	96.67%	40.00%	100.00%	80.00%	100.00%	93.33%	100.00%
Avg iterations ↓	PSO-sono	76.00	NaN	72.39	NaN	72.45	82.00	86.25
	HHO	28.29	29.00	34.50	7.00	23.50	29.75	40.00
	SCHO	34.57	44.60	29.32	30.00	41.44	16.69	39.15
	RIME	52.00	63.67	39.55	44.50	28.92	33.40	38.75
	MPCO	26.48	37.50	22.50	33.35	13.50	27.80	30.80
Avg time(s) ↓	PSO-sono	136.48	NaN	157.14	NaN	193.06	265.39	257.27
	HHO	76.36	91.02	84.92	14.16	68.76	96.48	134.39
	SCHO	90.61	138.44	79.27	61.98	128.98	45.50	119.01
	RIME	138.00	166.78	141.69	101.21	94.88	104.96	113.26
	MPCO	84.33	129.50	74.40	85.10	65.34	88.54	100.07
Avg fitness ↓	PSO-sono	-3.97	NaN	-3.79	NaN	-4.23	-1.28	-4.32
	HHO	-6.84	-7.00	-3.57	-5.51	-3.96	-2.55	-3.46 7
	SCHO	-5.44	-4.66	-4.70	-4.53	-6.45	-3.74	-4.75
	RIME	-3.95	-5.10	-5.14	-1.90	-6.16	-3.63	-4.77
	MPCO	-5.66	-6.32	-5.29	-4.10	-7.92	-4.55	-5.26

Table 3: The performances of MPCO and other algorithms on DLVAC tasks.

the best solution found; ***(iii) Parameters***: Population size (*NP*) = 50, maximum iterations = 200. The number of sub-processes (*NS*) = 5 for MPCO. Results are averaged over 20 independent runs. Hyperparameters are set according to the original works proposing these algorithms. ***(iv) Environment***: The simulation environment is MATLAB 2019b, running on a 13th Gen Intel(R) Core(TM) i7-13700 CPU with 16 cores and 24 virtual processors.

4.2 Experimental Results

Results on DLVAC Tasks. The relevant results obtained with all algorithms on DLVAC Tasks are presented in Table 3 and Fig. 4. The results of test functions are presented in Table4. It is evident that MPCO outperforms most algorithms in various aspects. Particularly in terms of success rate, MPCO consistently achieves the highest performance across all test functions, surpassing its base optimizers and other algorithms, showcasing its robust solving capability. Moreover, MPCO demonstrates efficiency in solving speed by requiring fewer iterations compared to other algorithms, leading to reduced solving time. However, it is noteworthy that in certain test functions, although MPCO exhibits fewer solving generations, the solving time may not reach the minimum due to additional overhead caused by the main process communicating with multiple sub-processes to acquire the total population pool. Nonetheless, MPCO still demonstrates superior solving efficiency overall. In terms of final fitness, MPCO consistently achieves superiority. This can be attributed to the incorporation of knowledge transfer strategies, which enable the majority of sub-processes to find feasible solutions in later iterations and effectively explore their vicinity. Additionally, the main process selects the best solution among all sub-processes, ensuring an excellent final result.

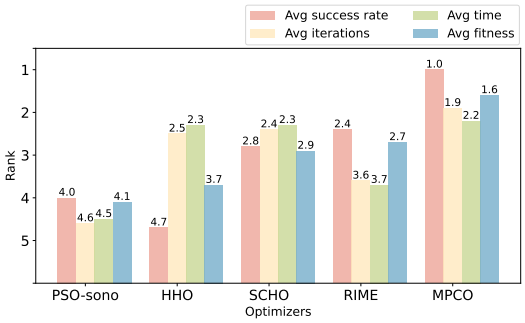


Figure 4: Average ranking of MPCO and other optimizers on DLVAC tasks.

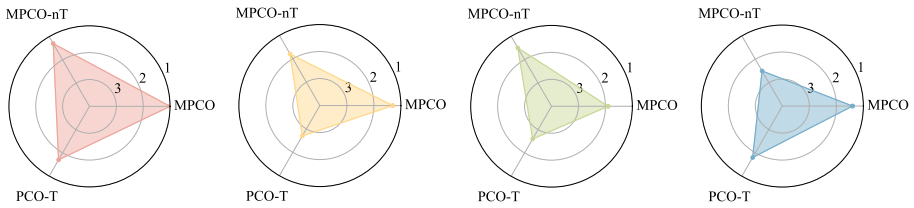
	Method	g01	g03	g06	g07	g08	g10	c13	c15	c16	c17
Avg success rate ↑	PSO	100%	100%	50%	100%	100%	100%	100%	95%	80%	65%
	FA	100%	100%	100%	100%	100%	75%	100%	100%	85%	80%
	CS	100%	100%	100%	100%	100%	100%	100%	95%	30%	20%
	GWO	100%	100%	70%	100%	100%	100%	100%	100%	20%	40%
	WOA	100%	100%	65%	95%	100%	35%	100%	100%	75%	100%
	PSO-sono	45%	85%	100%	100%	100%	100%	100%	95%	40%	5%
	HHO	100%	100%	100%	100%	100%	40%	100%	100%	65%	100%
	SCHO	100%	100%	100%	50%	100%	95%	100%	90%	5%	10%
	RIME	100%	85%	60%	0%	100%	65%	95%	0%	0%	0%
	MPCO	100%	100%	100%	100%	100%	100%	100%	100%	95%	100%
Avg iterations ↓	PSO	3.73	32.61	25.81	11.75	1.25	23.46	13.33	55.84	174.00	181.15
	FA	8.95	41.8	17.1	25.65	1.85	64.2	9.5	22.7	127.12	132.875
	CS	7.50	23.95	24.05	38.10	2.10	48.60	117.20	29.21	145.00	66.00
	GWO	2.30	55.30	9.29	6.55	1.65	26.75	11.00	40.60	175.25	176.50
	WOA	9.95	20.45	18.77	22.74	2.15	22.29	2.05	28.25	70.33	79.40
	PSO-sono	29.44	29.06	11.75	20.45	2.15	26.10	4.90	42.32	156.75	144.53
	HHO	2.1	33.05	21.65	12.8	2.3	111.25	4.05	23.45	105.46	108
	SCHO	3.65	29.54	22.31	13.36	2.10	18.26	12.55	65.11	24.00	141.90
	RIME	7.20	44.94	39.58	NaN	5.15	16.50	31.00	NaN	NaN	NaN
	MPCO	1	15.9	6.9	5.85	1.05	12.1	7.35	10.65	93.74	76
Avg time(ms) ↓	PSO	5.20	8.10	7.60	4.35	0.67	7.87	7.70	29.68	100.12	79.08
	FA	48.45	177.20	81.15	131.30	9.10	319.13	51.65	125.05	714.12	711.25
	CS	9.75	17.55	18.00	34.70	1.75	41.70	151.25	42.58	210.00	78.00
	GWO	3.60	20.20	3.14	3.60	0.90	11.70	10.30	42.70	179.50	127.50
	WOA	46.50	4.15	4.46	7.05	1.00	7.86	2.05	132.00	53.60	364.85
	PSO-sono	18.78	11.41	4.06	9.70	0.75	11.35	4.05	33.11	112.12	89.00
	HHO	3.55	15.70	11.30	7.75	2.25	66.38	4.60	21.70	109.15	89.70
	SCHO	15.65	6.35	5.75	5.20	0.63	5.26	6.60	36.61	12.00	71.20
	RIME	10.65	13.12	13.42	NaN	2.55	8.34	18.58	NaN	NaN	NaN
	MPCO	3.45	5.70	4.33	3.90	0.95	3.94	7.64	18.48	70.84	88.05
Avg fitness ↓	PSO	-7.10	-0.03	-6.94E+03	41.21	-9.58E-02	8.76E+03	-53.42	2.47E+14	1.10	1.29E+03
	FA	-3.49	-0.11	-6.27E+03	647.63	-9.58E-02	1.50E+04	-46.26	3.40E+12	0.96	1.47E+02
	CS	-13.76	-0.22	-6.94E+03	43.13	-9.58E-02	9.61E+03	-32.43	6.52E+14	1.22	2.82E+03
	GWO	-10.64	-0.00	-6.96E+03	154.70	-9.58E-02	8.70E+03	-46.85	1.20E+13	1.05	1.68E+02
	WOA	-12.59	-0.02	-6.27E+03	264.76	-9.58E-02	1.31E+04	-7.08	7.07E+14	1.13	1.64E+03
	PSO-sono	-2.18	-0.11	-5.32E+03	34.60	-9.58E-02	8.54E+03	-54.07	1.35E+14	1.05	3.04E+02
	HHO	-13.73	-0.01	-6.87E+03	40.45	-9.58E-02	1.19E+04	-53.75	1.86E+12	0.93	1.36E+03
	SCHO	-6.68	-0.02	-6.62E+03	459.90	-9.58E-02	1.06E+04	-41.80	4.02E+13	1.09	2.26E+02
	RIME	-13.45	-0.04	-5.99E+03	NaN	-0.08	9.82E+03	-54.67	NaN	NaN	NaN
	MPCO	-14.72	-0.16	-6.96E+03	27.29	-9.58E-02	7.73E+03	-56.46	3.54E+12	1.00	1.24E+02

Table 4: The performances of MPCO and other algorithms on selected constrained test functions.

Across the seven DLVAC Tasks examined, MPCO consistently achieves the best **Avg success rate** and maintains top rankings in **Avg iterations** in most instances, demonstrating excellent solving capability. Regarding the **Avg time** metric, MPCO slightly trails behind HHO and SCHO in some cases. Two main reasons account for this: firstly, the communication overhead between subprocesses, and secondly, in certain cases, HHO or SCHO exhibit very low **Avg success rate** but significantly lower computational costs. This phenomenon can be attributed to randomness, as the initial population may have been generated near the feasible solution region. Concerning the **Avg fitness** metric, MPCO also achieves the best average ranking. Our MPCO based on several classical optimizers, requires no additional settings or modifications to internal algorithm parameters. By introducing adaptive function landscape feature coordinates, it improves exploratory capability while enhancing generality.

Ablation Study The MPCO framework primarily consists of two components: individual position updates through multiple coordinate views and a knowledge transfer strategy. This section is divided into the analysis and experimentation of these components. For the ablation study, we define two sub-schemes: (i) MPCO-nT: Utilizes multiple coordinate views without the knowledge transfer strategy. Base optimizers remain the same as MPCO. (ii) PCO-T: Retains the knowledge transfer strategy but does not utilize multiple coordinate views. Base optimizers remain the same as MPCO.

The relevant performance results are presented in Fig. 5, with each indicator representing the average result of 20 runs. Upon removing the knowledge transfer strategy (i.e., MPCO-nT), a varying degree



(a) **Avg success rate** (b) **Avg iterations** (c) **Avg time** (d) **Avg fitness**

Figure 5: Average ranking of MPCO-nT, PCO-T and MPCO on DLVAC Tasks.

of reduction in the final fitness of the algorithm is observed across some test functions, according to Fig. 5(d). Conversely, as depicted in Fig. 5(a) and Fig. 5(b), the removal of the multi-coordinate view (i.e., PCO-T) leads to a decrease in the success rate and an increase in the number of solving iterations. In particular, Fig. 5(c) shows that MPCO ranks lower than MPCO-nT, as the addition of the knowledge transfer component incurs some additional computational overhead.

It can be inferred that the performance of MPCO benefits from position updates through multiple coordinate views and knowledge transfer, contributing to its robustness across diverse problem domains. The Eigen coordinate view offers valuable guidance, enhancing exploration and exploitation capabilities. Moreover, the knowledge transfer strategy helps sub-processes overcome local optima and find feasible solutions more effectively. By integrating these strategies, MPCO ensures improved solution quality while maintaining computational efficiency.

Sensitivity Study MPCO introduces no additional parameters aside from the number of sub-processes (*NS*), which we examine in this section. Fig. 6 displays the performance of MPCO across different *NS* values, with each metric representing the average of 20 runs. As *NS* increases, success rates gradually approach 100%, while iterations and time costs decrease. However, setting *NS* to 25 leads to similar or higher time costs compared to *NS=5*. This is due to *NS* exceeding the available processor count, causing queued sub-processes to await processor availability, thus prolonging execution times. In practical applications, it's vital to align the number of sub-processes with the available processors to maximize MPCO's performance. Overestimating *NS* beyond available processor capacity may lead to suboptimal execution times.

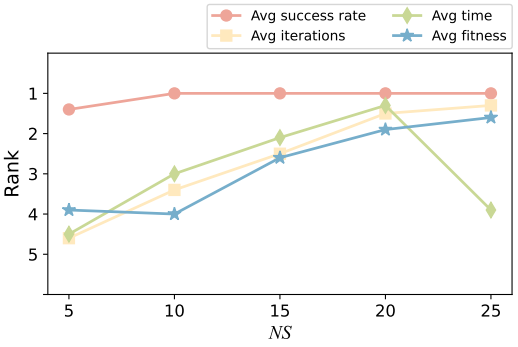


Figure 6: Average ranking of MPCO with different *NS* on DLVAC tasks.

5 Conclusion

In this paper, we develop MPCO, a multi-view parallel cooperative optimization framework tailored to address strongly constrained optimization problems, with a particular focus on launch vehicle attitude control parameter design. MPCO leverages parallel sub-process exploration with different base optimizers, along with multiple coordinate views and knowledge transfer, to enhance its exploratory capability. Additionally, we instantiate a version of MPCO using PSO, FA, CS, GWO, and WOA as base optimizers. Through comparative testing against several classical optimization algorithms on both test functions and real-world launch vehicle attitude control parameter design tasks, we demonstrate the effectiveness of the proposed framework and instantiation through comprehensive simulation result analysis. Notably, MPCO's adaptability, due to its absence of extra parameters, positions it to potentially enhance existing optimizers. Moving forward, we aim to expand the range of base optimizers in MPCO and potentially tailor specific optimizers to better address various real-world optimization challenges, thus enhancing the framework's flexibility and applicability.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, improving clarity and readability, and refining the presentation of the manuscript. We did not use generative AI tools to generate experimental data or to independently conduct experiments or draw scientific conclusions, and other required disclosure tasks are not applicable to this work. All AI-assisted content was carefully reviewed, revised, and verified by the authors to ensure its technical accuracy and consistency with the intended scientific claims. We take full responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

REFERENCES

Jianfu Bai, Yifei Li, Mingpo Zheng, Samir Khatir, Brahim Benaissa, Laith Abualigah, and Magd Abdel Wahab. A sinh cosh optimizer. *Knowledge-Based Syst.*, 282:111081, 2023.

Boris Benedikter, Alessandro Zavoli, Guido Colasurdo, Simone Pizzurro, and Enrico Cavallini. Convex optimization of launch vehicle ascent trajectory with heat-flux and splash-down constraints. *Journal of Spacecraft and Rockets*, 59(3):900–915, 2022.

Qi Bian, Brett Nener, and Xinmin Wang. Control parameter tuning for aircraft crosswind landing via multi-solution particle swarm optimization. *Eng. Optimization*, 50(11):1914–1925, 2018.

Qi Bian, Brett Nener, Ting Li, and Xinmin Wang. Multimodal control parameter optimization for aircraft longitudinal automatic landing via the hybrid particle swarm-bfgs algorithm. *Proc. Institution Mechanical Engineers, G, J. Aerosp. Eng.*, 233(12):4482–4491, 2019.

Il-Yop Chung, Wenxin Liu, David A Cartes, and Karl Schoder. Control parameter optimization for a microgrid system using particle swarm optimization. In *2008 IEEE Int. Conf. Sustainable Energy Technol.*, pp. 837–842. IEEE, 2008.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Hao Hao, Xiaoqun Zhang, and Aimin Zhou. Enhancing saeas with unevaluated solutions: a case study of relation model for expensive optimization. *Sci. China Inf. Sci.*, 67(2):1–18, 2024.

Ali Asghar Heidari, Seyedali Mirjalili, Hossam Faris, Ibrahim Aljarah, Majdi Mafarja, and Huiling Chen. Harris hawks optimization: Algorithm and applications. *Future Generation Comput. Syst.*, 97:849–872, 2019.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proc. ICNN'95-Int. Conf. Neural Networks*, volume 4, pp. 1942–1948. IEEE, 1995.

Jiasheng Li and Wenguo Li. On-line pid parameters optimization control for wind power generation system based on genetic algorithm. *IEEE Access*, 8:137094–137100, 2020.

Jing J Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *J. Appl. Mechanics*, 41(8):8–31, 2006.

Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 24:910, 2010.

Zhenyu Meng, Yuxin Zhong, Guojun Mao, and Yan Liang. PSO-sono: A novel pso variant for single-objective numerical optimization. *Inf. Sci.*, 586:176–191, 2022.

Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Adv. Eng. Softw.*, 95: 51–67, 2016.

Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Adv. Eng. Softw.*, 69:46–61, 2014.

AA Mousa, Mohammed A El-Shorbagy, and Waiel F Abd-El-Wahed. Local search based hybrid particle swarm optimization algorithm for multiobjective optimization. *Swarm Evol. Comput.*, 3: 1–14, 2012.

Shengzhao Pang, Heng Liu, Yingxue Chen, Bo Cheng, Zhaoyong Mao, and Yigeng Huangfu. Distributed electric propulsion aircraft attitude control based on assisted training multi-agent reinforcement learning. *IEEE Transactions on Transportation Electrification*, 2025.

Chao Qian, Dan-Xuan Liu, and Zhi-Hua Zhou. Result diversification by multi-objective evolutionary algorithms with theoretical guarantees. *Artif. Intell.*, 309:103737, 2022.

Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *IEEE Int. Conf. Evol. Comput. Proc. IEEE World Congress Comput. Intell.*, pp. 69–73. IEEE, 1998.

Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

Hang Su, Dong Zhao, Ali Asghar Heidari, Lei Liu, Xiaoqin Zhang, Majdi Mafarja, and Huiling Chen. RIME: A physics-based optimization. *Neurocomputing*, 532:183–214, 2023.

Xuewen Xia, Ling Gui, Guoliang He, Chengwang Xie, Bo Wei, Ying Xing, Ruifeng Wu, and Yichao Tang. A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm. *J. Comput. Sci.*, 26:488–500, 2018.

Xin-She Yang and Suash Deb. Cuckoo search: recent advances and applications. *Neural Comput. Appl.*, 24:169–174, 2014.

Xin-She Yang and Xingshi He. Firefly algorithm: recent advances and applications. *Int. J. Swarm Intell.*, 1(1):36–50, 2013.

Zhilong Yu, Yinghui Li, Maolong Lv, Binbin Pei, Wenfeng Xu, and Xiangwei Bu. Fixed-time adaptive fuzzy fault-tolerant attitude control for tailless aircraft without angular velocity measurements. *IEEE Transactions on Fuzzy Systems*, 32(4):2458–2470, 2024.

Hongyu Zhou, Xiaogang Wang, Xun Li, and Naigang Cui. Trajectory planning for reusable launch vehicle with multisolution-based maximum entropy reinforcement learning. *IEEE Transactions on Aerospace and Electronic Systems*, 61(5):11613–11627, 2025.

A Flight Attitude Indicators

A.1 Bode plots

For compactness, we introduce four auxiliary functions to represent the contributions of the second-order factors. Specifically, $f_{b,k}^{A,t}(\omega)$ and $f_{d,k}^{A,t}(\omega)$ characterize the magnitude contributions of the second-order factors in the numerator and denominator of $G_k^t(s)$, respectively, while $f_{b,k}^{\varphi,t}(\omega)$ and $f_{d,k}^{\varphi,t}(\omega)$ characterize their corresponding phase contributions. Their explicit forms are given by:

$$\begin{cases} f_{b,k}^{A,t}(\omega) = \left(1 - \left(\dfrac{\omega}{\omega_{b,k}^t}\right)^2\right)^2 + \left(2\xi_{b,k}^t \dfrac{\omega}{\omega_{b,k}^t}\right)^2, \quad f_{b,k}^{\varphi,t}(\omega) = \dfrac{2\xi_{b,k}^t \frac{\omega}{\omega_{b,k}^t}}{1 - \left(\frac{\omega}{\omega_{b,k}^t}\right)^2}, \\ f_{d,k}^{A,t}(\omega) = \left(1 - \left(\dfrac{\omega}{\omega_{d,k}^t}\right)^2\right)^2 + \left(2\xi_{d,k}^t \dfrac{\omega}{\omega_{d,k}^t}\right)^2, \quad f_{d,k}^{\varphi,t}(\omega) = \dfrac{2\xi_{d,k}^t \frac{\omega}{\omega_{d,k}^t}}{1 - \left(\frac{\omega}{\omega_{d,k}^t}\right)^2}. \end{cases} \tag{9}$$

The flight attitude indicators used to evaluate control parameters are extracted from the Bode plots generated by the corresponding transfer functions, following domain-specific rules. As shown in Figure 7, these indicators serve as the optimization objectives of the DLVAC tasks.

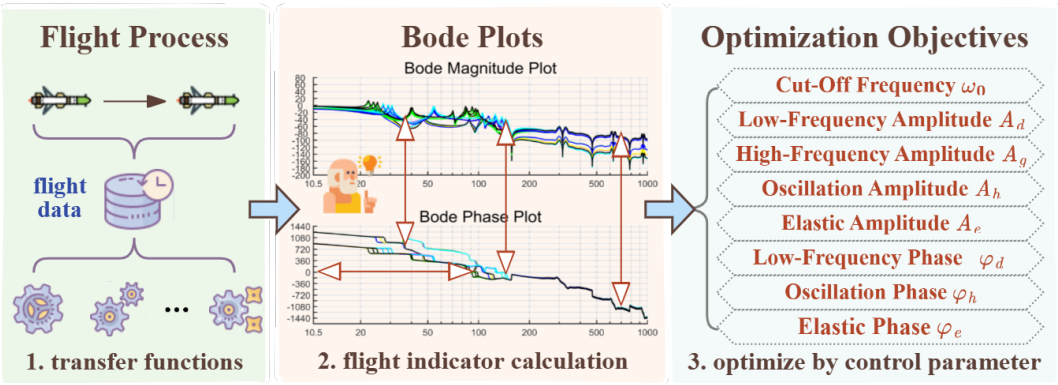


Figure 7: Illustration of the generation of launch vehicle flight attitude indicators.

A.2 Domain-specific Rules

Based on the Bode plots, the flight control indicators of the correction networks are only determined using the following rules:

- **Cut-Off Frequency** (ω_0): The frequency point where the magnitude characteristic curve first crosses 0 dB downward within the entire frequency range.
- **Low-Frequency Amplitude** (A_d): The frequency point where the phase characteristic curve first crosses 180° upwards corresponds to the magnitude point on the magnitude-frequency characteristic curve.
- **High-Frequency Amplitude** (A_g): The frequency point where the phase characteristic curve first crosses 180° downwards corresponds to the magnitude point on the magnitude-frequency characteristic curve.
- **Oscillation Amplitude** (A_h): The magnitude point corresponding to the peak value on the magnitude characteristic curve.
- **Elastic Amplitude** (A_e): The magnitude point corresponding to the peak value on the magnitude characteristic curve.
- **Low-frequency Phase** (φ_d): The frequency point where the magnitude-frequency characteristic curve first crosses 0 dB corresponds to the phase point on the phase-frequency characteristic curve, with the difference from 180°.
- **Oscillation Phase** (φ_h): The frequency point where the magnitude-frequency characteristic curve crosses 0 dB corresponds to the phase point on the phase-frequency characteristic curve, with the absolute difference from 180°.
- **Elastic Phase** (φ_e): The frequency point where the magnitude-frequency characteristic curve crosses 0 dB corresponds to the phase point on the phase-frequency characteristic curve, with the absolute difference from 180°.

It is important to note that when multiple results are available, the maximum or minimum value is selected. While the rule for elastic gain is identical to that of oscillation gain, the frequency range and target values differ. Similarly, although the rule for the elastic phase margin is the same as for the oscillation phase, the specified frequency range and target values also vary.

B MPCO Algorithms

We provide the pseudo-code of our MPCO in Algorithm 1, Algorithm 2, and Algorithm 3.

Algorithm 1: Framework of **MPCO**

Input: $NP, NS, D, \mathbf{lD} = (ld_1, \ldots, ld_D), \mathbf{uD} = (ud_1, \ldots, ud_D), \hat{\Gamma} = (\Gamma_1, ..., \Gamma_{NP})$;
Output: The Best Solution;

$\hat{\mathbb{P}} = \{\mathbb{P}_1, \ldots, \mathbb{P}_{NS}\} \leftarrow$ pop.Init($NP, NS, D, \mathbf{lD}, \mathbf{uD}$);
for $i = 1$ to *NS* **do**
  // Initialize the optimizer's hyperparameters.
  $\Gamma_i \leftarrow$ opt.Init($\mathbb{P}_i, \Gamma_i$);
end
// Conduct iterative searches.
while *the termination criterion is not met* **do**
  // Update the covariance matrix based on population information.
  $\mathbf{C} \leftarrow$ updateCovarianceMatrix($\mathbf{C}, \hat{\mathbb{P}}$);
  // Update the Eigen coordinate view.
  $\mathcal{R}_E \leftarrow$ updateEigenView($\mathbf{C}$);
  // Execute an iterative search in parallel for each sub-process.
  for $i = 1$ to *NS* **do**
    for $j = 1$ to *NP* **do**
      // Generate a new position under the original coordinate view.
      $\overrightarrow{x_o} \leftarrow$ genPos($\overrightarrow{x_{i,j}}, \mathcal{R}_O, \Gamma_i$);
      // Generate a new position under the Eigen coordinate view.
      $\overrightarrow{x_e} \leftarrow$ genPos($\overrightarrow{x_{i,j}}, \mathcal{R}_E, \Gamma_i$);
      // Determine the final new position.
      $\overrightarrow{x_{i,j}} \leftarrow$ updatePos($\overrightarrow{x_{i,j}}, \overrightarrow{x_o}, \overrightarrow{x_e}$);
      Evaluate the individual;
      Update the essential records;
    end
    Update the best solution;
  end
  // Conduct the knowledge transfer.
  if *met the round to knowledge transfer* **then**
    // Select the source domain.
    $\mathbb{S} \leftarrow$ getSourceDomain($\hat{\mathbb{P}}$);
    // Select the target domain.
    $\mathbb{T} \leftarrow$ getTargetDomain($\hat{\mathbb{P}}$);
    for $t \in \mathbb{T}$ **do**
      // Transfer the best solutions.
      $\hat{\mathbb{P}}_t \leftarrow$ transfer($\mathbb{S}, \hat{\mathbb{P}}_t$);
      Update the essential records;
    end
  end
end
// Select the best result among the optimal solutions of all sub-processes.
Result $\leftarrow$ chooseBest($\hat{\mathbb{P}}$);
Return Result.;

C Additional Experiments

Test Functions. To simulate real-world constrained optimization problems, we selected benchmark test functions from CEC2006 (Liang et al., 2006) and CEC2010 (Mallipeddi & Suganthan,

Algorithm 2: Population Initialization

Input: $NP, NS, D, \mathbf{lD} = (ld_1, ld_2, \ldots, ld_D), \mathbf{uD} = (ud_1, ud_2, \ldots, ud_D)$;
Output: $\hat{\mathbb{P}} = \{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_{NS}\}$;

$\lambda = NP * NS$;
$\mathbf{S} = \{\overrightarrow{s_1}, \overrightarrow{s_2}, \ldots, \overrightarrow{s_\lambda}\} \leftarrow$ LHS$(D, \mathbf{lD}, \mathbf{uD}, \lambda)$;
for $n = 1$ to *NS* **do**
  $\mathbb{P}_n \leftarrow \emptyset$;
  for $i = 1$ to *NP* **do**
    Initialize $\overrightarrow{x} = (x_1, x_2, \ldots, x_D) = (0, 0, \ldots, 0)$;
    for $j = 1$ to *D* **do**
      $row = (n-1) * NP + i$;
      $x_j = ld_j + s_{row,j} * (ud_j - ld_j)$;
    end
    $\mathbb{P}_n = \mathbb{P}_n \cup \overrightarrow{x}$;
  end
end
Return $\hat{\mathbb{P}} = \{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_{NS}\}$;

Algorithm 3: Knowledge Transfer

$\mathbb{S} \leftarrow \emptyset$;
$\mathbb{T} \leftarrow \emptyset$;
for $i = 1$ to *NS* **do**
  if *sub-process*$_i$ *has found a feasible solution* **then**
    $\mathbb{S} = \mathbb{S} \cup$ *sub-process*$_i$;
  else
    $\mathbb{T} = \mathbb{T} \cup$ *sub-process*$_i$;
  end
end
if $\mathbb{S}$ *is not empty* **then**
  Collect current optimal solutions in $\mathbb{S}$ as $ePOP$;
  for $t \in \mathbb{T}$ **do**
    Replace the worst individuals in *sub-process*$_t$ with the $ePOP$;
    Initialize the necessary records of the replaced individuals;
    Evaluate the replaced individuals;
    Update the essential records;
  end
end

2010). Table 2 lists the selected functions, where D denotes problem dimensionality. The ρ metric quantifies the feasible solution space area, calculated as $\rho = |F|/|S|$, with $|F|$ representing feasible solutions and $|S|$ the total randomly generated. In our study, $|S|$ comprised 1,000,000 solutions. It's worth noting that, a ρ value of 0.0000% implies an extremely small feasible region, constituting less than 0.0001% of the total. **E** and **I** denote equality and inequality constraints, respectively. Parameter **a** indicates constraints effectively surrounding the optimal solution, with "-" indicating undetermined values. Higher **a** values denote greater challenges for achieving optimality.

It is worth noting that, to demonstrate strong constraint problems, we selectively chose individual instances with very small ρ values from CEC2006 and CEC2010. For the sake of generality, we additionally included test function g08 with a relatively large ρ value to showcase MPCO's performance in solving general constrained optimization problems.

Baseline Methods. The baseline methods comprise a selection of classical single-objective optimization algorithms, including PSO (Kennedy & Eberhart, 1995), FA (Yang & He, 2013), CS (Yang & Deb, 2014), GWO (Mirjalili et al., 2014), WOA (Mirjalili & Lewis, 2016), PSO-sono (Meng

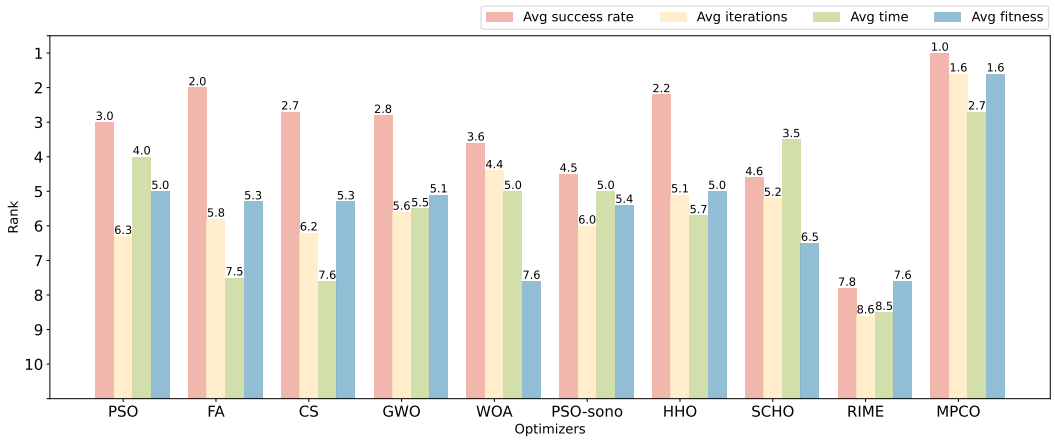


Figure 8: Average ranking of MPCO and other optimizers on Test Functions.

et al., 2022), HHO (Heidari et al., 2019), SCHO (Bai et al., 2023), and RIME (Su et al., 2023). Among them, PSO, FA, CS, GWO, and WOA are renowned for their exceptional performance as swarm intelligence algorithms, demonstrating robust global exploratory capability and adaptability. Furthermore, these five optimizers serve as the base algorithms for MPCO. PSO-sono enhances PSO's global search capability by refining the iteration paradigm. Conversely, the remaining algorithms introduce novel operators tailored for multidimensional optimization problems, exhibiting commendable performance as well.

Performance Evaluation. The results obtained on test functions are presented in Table 4 and Figure 8. The ablation results on test functions are presented in Table 5. The sensitivity analysis on test functions is presented in Table 6.

Table 5: The performance of MPCO-nT and PCO-T compared to MPCO on selected constrained test functions. The best performances are underlined.

	Method	g01	g03	g06	g07	g08	g10	c13	c15	c16	c17
Avg success rate ↑	MPCO-nT	100%	100%	100%	95%	100%	100%	100%	100%	90%	95%
	PCO-T	100%	100%	90%	95%	100%	90%	100%	100%	90%	95%
	MPCO	100%	100%	100%	100%	100%	100%	100%	100%	95%	100%
Avg iterations ↓	MPCO-nT	1.00	14.05	7.55	6.50	1.15	14.90	7.90	10.45	99.26	80.05
	PCO-T	1.05	15.35	8.80	7.90	1.15	19.50	7.65	10.80	124.89	103.55
	MPCO	1.00	15.90	6.90	5.85	1.05	12.10	7.35	10.65	93.74	76.00
Avg time(ms) ↓	MPCO-nT	3.10	5.66	4.55	4.58	0.81	3.63	7.38	18.06	76.46	97.66
	PCO-T	3.43	5.36	4.61	4.79	0.89	5.30	7.92	18.30	95.37	119.81
	MPCO	3.45	5.70	4.33	3.90	0.95	3.94	7.64	18.48	70.84	88.05
Avg fitness ↓	MPCO-nT	-13.80	-0.13	-6.26E+03	36.94	-9.58E-02	8.09E+03	-49.82	3.75E+12	0.97	141.83
	PCO-T	-13.72	-0.14	-6.58E+03	30.93	-9.58E-02	8.09E+03	-54.82	3.50E+12	0.95	109.55
	MPCO	-14.72	-0.16	-6.96E+03	27.29	-9.58E-02	7.73E+03	-56.46	3.54E+12	1.00	123.51

Table 6: The performances of MPCO with different *NS* on selected constrained test functions. The best performances are underlined.

	NS	g01	g03	g06	g07	g08	g10	c13	c15	c16	c17
Avg success rate ↑	5	100%	100%	100%	100%	100%	100%	100%	100%	95%	100%
	10	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	15	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	20	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	25	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Avg iterations ↓	5	1.00	15.90	6.90	5.85	1.05	12.10	7.35	10.65	93.74	76.00
	10	1.00	10.90	4.85	5.50	1.00	8.45	6.90	6.10	55.05	49.70
	15	1.00	8.25	4.45	5.00	1.00	8.40	6.50	4.70	55.00	33.45
	20	1.00	6.05	3.60	4.80	1.00	7.40	6.20	4.15	44.00	37.40
	25	1.00	6.95	3.40	4.30	1.00	7.70	6.20	4.05	51.55	30.95
Avg time(ms) ↓	5	3.45	5.70	4.33	3.90	0.95	3.94	7.64	18.48	70.84	88.05
	10	3.44	3.91	3.02	3.65	0.90	2.75	7.17	10.59	41.44	58.42
	15	3.45	2.96	2.77	3.32	0.91	2.74	6.76	8.16	40.40	39.32
	20	3.43	2.17	2.24	3.18	0.91	2.41	6.44	7.20	33.12	43.97
	25	4.22	3.26	3.14	4.84	1.38	3.61	10.30	10.37	44.38	72.99
Avg fitness ↓	5	-14.72	-0.16	-6.96E+03	27.29	-9.58E-02	7.73E+03	-56.46	3.54E+12	1.00	123.51
	10	-14.07	-0.30	-6.96E+03	26.27	-9.58E-02	7.64E+03	-56.17	1.38E+12	0.95	77.60
	15	-14.51	-0.38	-6.96E+03	26.25	-9.58E-02	7.56E+03	-57.94	8.09E+11	0.85	76.54
	20	-14.98	-0.42	-6.96E+03	25.99	-9.58E-02	7.19E+03	-57.25	5.69E+11	0.83	76.65
	25	-14.99	-0.51	-6.96E+03	26.00	-9.58E-02	7.20E+03	-58.54	3.82E+11	0.85	65.47