IEEE Transactions on Aerospace and Electronic Systems, Volume 57, Issue 3, June 2021, pp. 1685-1701
DOI: 10.1109/TAES.2021.3050645

Solving Constrained Trajectory Planning Problems Using Biased Particle Swarm Optimization

Runqi Chai, *Member, IEEE,* Antonios Tsourdos, Al Savvaris, Senchun Chai, *Senior Member, IEEE,* and Yuanqing Xia, *Senior Member, IEEE,*

***Abstract*—Constrained trajectory optimization has been a critical component in the development of advanced guidance and control systems. An improperly planned reference trajectory can be a main cause of poor online control performance. Due to the existence of various mission-related constraints, the feasible solution space of a trajectory optimization model may be restricted to a relatively narrow corridor, thereby easily resulting in local minimum or infeasible solution detection. In this work, we are interested in making an attempt to handle the constrained trajectory design problem using a biased particle swarm optimization approach. The proposed approach reformulates the original problem to an unconstrained multi-criterion version by introducing an additional normalized objective reflecting the total amount of constraint violation. Besides, to enhance the progress during the evolutionary process, the algorithm is equipped with a local exploration operation, a novel ε-bias selection method, and an evolution restart strategy. Numerical simulation experiments, obtained from a constrained atmospheric entry trajectory optimization example, are provided to verify the effectiveness of the proposed optimization strategy. Main advantages associated with the proposed method are also highlighted by executing a number of comparative case studies.**

***Index Terms*—Trajectory optimization, particle swarm optimization, local exploration, bias selection, restart strategy.**

I. Introduction

CONSTRAINED trajectory planning problems widely exist in the aerospace industry and considerable attention has been given to research advanced trajectory optimization algorithms during the last decade. Although this step is often performed offline in practical applications, an improperly planned reference trajectory may significantly damage the online control performance or even result in a failure of the mission. Therefore, to gain enhanced guidance and control performance, a proper design of the optimal maneuver trajectory is highly demanded. It should be noted that approaches to address this kind of problem mainly fall into two categories: indirect methods and direct methods [1], [2]. An indirect approach uses an "optimize then discretize" strategy, where first-order optimality conditions for the differential algebraic equations (DAE)-constrained systems are directly derived and solved. Many important contributions have been made on applying this type of method [3]–[5]. For example, Yang and Hexi [3] applied an indirect method in order to produce the energy-optimal trajectory for an irregular asteroid landing mission. Pontani and Conway developed an indirect heuristic approach in [4] and successfully implemented this method to address a low-thrust orbital transfer problem. Furthermore, in their later work [5], this indirect approach was extended to solve more complex trajectory design problems with enhanced numerical accuracy.

While the results from an indirect method can be treated as the theoretically optimal solutions, the application of indirect methods is usually limited to relatively low-dimensional problems and it tends to be ineffective with respect to problems containing complex path constraints. On the other hand, a direct approach applies a "discretize then optimize" strategy, where state and/or control variables are firstly discretized such that the original problem is reformulated to a nonlinear programming problem (NLP) or more precisely, a static parameter optimization problem containing a finite number of decision variables [6], [7]. A primary advantage of applying this kind of method is that it can be easily combined with well-developed parameter optimization solvers in order to address the resulting NLP. In addition, different types of system constraints are represented by a relatively straightforward manner. As a result, we pay more attention to the implementation of "discretize then optimize" approaches.

In recent years, extensive applications of direct methods can be found in the context of flight vehicle trajectory optimization and robotic motion planning problems in difficult environments [8]–[10]. For instance, a direct collocation method was adopted in [8] in order to calculate the optimal flight trajectory for an entry vehicle during the Mars entry phase. Path constrains were imposed such that the vehicle can fly along a restricted corridor. The authors in [9] implemented a direct shooting method to plan the motion of a manipulator. In their work, both the higher order dynamics and the contact/friction force constraints were taken into account, thereby making the optimization model much more complex. In addition, a direct trajectory optimization framework for general manipulation platforms was established in [10], wherein multiple mission-related constraints such as the environmental constraints, collision avoidance constraints, and some geometric constraints were involved in the problem formulation and considered during the optimization.

Apart from direct methods, the design and test of heuristic methods such as particle swarm optimization (PSO), genetic algorithm (GA) and differential evolution (DE) have also received significant attention for solving the constrained tra-

R. Chai, A. Tsourdos and A. Savvaris are with the School of Aerospace, Transport and Manufacturing, Cranfield University, UK, e-mail: (r.chai@cranfield.ac.uk), (a.tsourdos@cranfield.ac.uk), and (a.savvaris@cranfield.ac.uk).

S. Chai and Y. Xia are with the school of Automation, Beijing Institute of Technology, Beijing, China, e-mail: (chaisc97@163.com), (xia_yuanqing@bit.edu.cn).

jectory planning problems. The main reason of using these particular optimization algorithms is due to the fact that the numerical gradient-dependent optimization algorithms such as the interior-point method (IPM) [11], sequential quadratic programming (SQP) [12], [13], and other modified versions [14], [15] only guarantee convergence toward local optima, whereas heuristic methods are more likely to locate the globally optimal result. This advantage has been highlighted by a number of relevant works and because of this, various heuristic approaches have been proposed to address constrained space vehicle trajectory optimization problems [16]–[19]. For example, an improved PSO algorithm was reported in [17] to solve a reusable launch vehicle reentry trajectory design problem. In this approach, a modified mutation mechanism was designed to facilitate the evolutionary process. A parallel optimization framework incorporating PSO, GA, and DE was suggested by the authors of [18], wherein an Earth-to-Mars interplanetary problem, together with a multiple-impulse rendezvous mission, was addressed. Although the two problems were successfully addressed by this hybrid method, simulation results also revealed that the performance of the proposed method tends to be problem dependent. Furthermore, in [19], by defining the number of switches as the optimization parameters, a segmented PSO method was advocated to explore the optimal control sequence with a bang-bang structure for a time-optimal slew maneuver task. However, if a maneuver planning problem contains singular arcs or the optimal control sequence does not hold a bang-bang structure, this PSO-based trajectory planner may not be able to generate promising results or even fail to find feasible solutions.

Commonly, each particle among the swarm represents a candidate solution to an optimization problem. Based on the reported experimental results, most of the researchers or engineers concluded that with a proper selection of algorithm parameters, the PSO has the capability of getting rid of local minima for different trajectory design problems [16], [20]. This can be attributed by the fact that in the swarm update formula, both the experience of the group of particles (e.g., the so-called social component) and the experience of each individual (e.g., the so-called cognitive component) are taken into account. Moreover, compared with other evolutionary optimization algorithms such as the GA and DE, the PSO can benefit from a reduced number of function evaluations, thus making it more efficient. This key finding was validated by the investigations presented in [21] and [22]. Benefiting from the key features discussed above, the implementation of PSO and its enhanced versions to various engineering optimization problems can be appreciated and encouraged.

When applying bio-inspired optimization methods to handle trajectory planning problems, the constraint handling strategy usually plays a key role and it can significantly affect the performance of the optimization process. It is worth noting that in most PSO-based trajectory optimization solvers, a penalty function (PF) approach is commonly applied to deal with various parameter constraints adhered to the optimization model [17], [23]–[25]. That is, an additional term (e.g., the so-called penalty term) reflecting the constraint violation is augmented in the fitness function, and the particle with smaller fitness value will be considered as a better individual compared to the others among the current swarm. This method is fairly straightforward to understand and easy to implement. However, difficulties may occur in balancing the emphasis between the mission objective and penalty terms.

In order to avoid the assignment of penalty functions and additional penalty factors, we apply a locally enhanced multi-objective PSO (MOPSO) method to tackle the constrained trajectory planning problems. This method is performed by firstly defining the total amount of constraint violation as an additional objective, thereby reformulating the constrained optimization problem as an unconstrained multi-criterion version. Then classical non-dominant sorting process is used to rank all the candidate solutions. Note that the use of a PSO algorithm in constrained problems using a multi-objective approach have been reported in some important works [26], [27] and applications of this strategy to constrained engineering optimization problems have attracted significant attention (e.g., a detailed review can be found in [28]). However, most reported works addressed the problem by purely relying on the pareto dominance. Based on our previous experiments [23], [24], it was found that a direct implementation of MOPSO and pareto dominance rules to the transformed multi-objective trajectory optimization problem may lack search bias in terms of the mission constraints. Therefore, a biased search toward the feasible region should be introduced, otherwise the algorithm performance might be degraded significantly.

The main contribution of this paper lies in the following four aspects:

1) We present an attempt to address the insufficient bias issue for standard MOPSO algorithm by introducing a constraint violation (CV)-based bias selection strategy. Then, this strategy is extended to a more general form, named the ε-bias selection strategy, such that it can become more flexible to optimize the objective function and reduce the solution infeasibility at the same time.
2) An evolution restart strategy is designed and embedded in the biased MOPSO algorithm such that it can acquire an enhanced capability to avoid getting stuck in local infeasible regions.
3) The proposed approach is applied to solve a constrained atmospheric entry trajectory design problem which is similar to the one investigated in [23] except that more constraints are modeled and included in the optimization model. Case studies, along with detailed analysis, are provided to emphasize the importance of the proposed bias selection process as well as the evolution restart strategy.
4) The proposed method is compared to other evolutionary algorithms and an off-the-shelf numerical optimal control solver (named CASADI). Comparative results not only characterize the key feature but also highlight the main advantage of applying the designed approach.

To the best of the authors' knowledge, the extended bias selection method and the evolution restart strategy are firstly combined in the locally enhanced-MOPSO in this paper for addressing the constrained reentry trajectory optimization

problem.

The rest of this article is constructed as follows. In Section II, the locally enhanced-MOPSO, along with the designed ε-bias selection method and the evolution restart strategy, is introduced. Section III presents the optimal control formulation of the constrained atmospheric entry trajectory design problem in detail. Numerical simulation experiments as well as a number of comparative studies are demonstrated in Section IV. Finally, this article is concluded in Section V.

II. Biased Particle Swarm Optimization Approach

A. Constrained Optimal Control Problem

In a constrained optimal control problem, a dynamical system is commonly adhered, which has the form of

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

where $x(t) \in \mathbb{R}^{n_x}$ and $y(t) \in \mathbb{R}^{n_u}$ represent, respectively, the system state and control variables defined on the time interval $t \in [t_0, t_f]$. n_x and n_u are the dimensions of the state and control. Variable path constraints and boundary conditions are frequently considered for a number of practical missions. They can be expressed by:

$$\begin{aligned} &g(x(t), u(t)) \leq 0 \\ &x(t_0) = x_0 \\ &x(t_f) = x_f \end{aligned} \tag{2}$$

The objective function is used to evaluate the system performance for a specific mission profile. A general form of it can be written as:

$$\min_{u(t)} J_1 = \int_{t_0}^{t_f} L(x(t), u(t))dt + \Phi(x(t_f), t_f) \tag{3}$$

where $L(x(t), u(t))$ and $\Phi(x(t_f), t_f)$ are, respectively, the process and terminal performance indicators. As a result, the overall constrained optimal control problem can be modeled as:

$$\begin{aligned} \min_{u(t)} \quad & J_1 = \int_{t_0}^{t_f} L(x(t), u(t))dt + \Phi(x(t_f), t_f) \\ \text{s.t.} \quad & \forall t \in [t_0, t_f] \\ & \dot{x}(t) = f(x(t), u(t)) \\ & g(x(t), u(t)) \leq 0 \\ & x(t_0) = x_0 \\ & x(t_f) = x_f \end{aligned} \tag{4}$$

B. Unconstrained Multi-Objective Optimal Control Problem

A direct transcription method is adopted to solve the constrained optimal control problem given by Eq.(4). That is, the control variable is parameterized over a finite set of temporal nodes $\{t_i\}_{i=0}^{N_k-1}$, in which N_k stands for the size of the temporal set. The discretized control sequence is then denoted as $u = (u_0, ..., u_{N_k-1})$. After the control discretization and numerical integration for ordinary differential equations, the static version of problem (5) can be written as:

$$\begin{aligned} \min_{u(t_i)} \quad & J_1 = \sum_{i=0}^{N_k-1} L(x(t_i), u(t_i))dt + \Phi(x(t_{N_k}), u(t_{N_k})) \\ \text{s.t.} \quad & \forall t_i, i \in \{0, 1, ..., N_k\} \\ & x(t_{i+1}) = x(t_i) + \Delta h \sum_{j=1}^{s} b_j f(x_j, u_{ij}) \\ & x_j = x(t_i) + \Delta h \sum_{m=1}^{s} a_{jm} f(x_m, u_{im}) \\ & g(x(t_i), u(t_i)) \leq 0 \\ & x(t_{N_k}) = x_f \end{aligned} \tag{5}$$

where x_m and u_{im} stand for the intermediate state and control values defined on $[t_i, t_{i+1}]$. Δh is the step length, while b_j and a_{jm} are discretization coefficients determined by the applied numerical integration method. Take fourth order Runge-Kutta method as an example, (b_1, b_2, b_3, b_4) can be set to $(\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6})$, whereas the non-zero elements of a_{jm} can be assigned as (a_{21}, a_{32}, a_{43})=$(\frac{1}{2}, \frac{1}{2}, 1)$, respectively.

Instead of directly addressing the constrained optimization problem (5), a slight modification of the problem formulation may also be effective. For example, the standard interior-point method (as well as its enhanced versions) applies the barrier function and solves the scalarized version of the problem. In this subsection, we introduce an additional normalized objective function so as to transform problem (5) to an unconstrained multi-objective version, which will then be optimized by the MOPSO algorithm introduced in the following subsections.

If the optimization problem contains m inequality constraints and n terminal constraints, the constraint violation value of a candidate solution (x, u) for the ith inequality constraint and jth terminal constraint can be written as:

$$\mu_g^i = \begin{cases} 0, & g^i(x,u) \leq 0; \\ \frac{g^i(x,u)}{\bar{g}^i}, & 0 \leq g^i(x,u) \leq \bar{g}^i; \\ 1, & g^i(x,u) \geq \bar{g}^i. \end{cases} \tag{6}$$

$$\mu_{x_f}^j = \begin{cases} 1, & x^j(t_{N_k}) \geq \bar{x}_f^j; \\ \frac{x^j(t_{N_k}) - x_f^j}{\bar{x}_f^j - x^j(t_{N_k})}, & x_f^j \leq x^j(t_{N_k}) \leq \bar{x}_f^j; \\ 0, & x^j(t_{N_k}; u) = x_f^j; \\ \frac{x_f^j - x^j(t_{N_k})}{x_f^j - \underline{x}_f^j}, & \underline{x}_f^j \leq x^j(t_{N_k}) \leq x_f^j; \\ 1, & \underline{x}_f^j \geq x^j(t_{N_k}). \end{cases} \tag{7}$$

In Eq.(6) and Eq.(7), $\bar{g}^i$=$\max(g^i(x,u))$ is the maximum violation value of the ith inequality constraint in the current searching space. The terms $\bar{x}_f^j$ and $\underline{x}_f^j$ can be defined analogically. x_f^j stands for the jth targeted terminal state value. For simplicity, constraint functions defined in Eq.(6) and Eq.(7) assume scalar values of the constraints. If the constraint functions of a candidate solution become a vector, to execute the division operation, each element in the vector should be divided by $\bar{g}^i$, $\bar{x}_f^j$ or $\underline{x}_f^j$, correspondingly.

Note that $\bar{g}^i$, $\bar{x}_f^j$ and $\underline{x}_f^j$ are mainly applied to normalize each constraint violation. From the definition of Eq.(6) and

Eq.(7), it is obvious that the values of $\mu_g^i,\ \mu_{x_f}^j \in [0,1]$ are able to reflect the magnitude of constraint violation for the constraints given by (2). As a result, based on Eq.(6) and Eq.(7), the normalized constraint violation function can be added together, thereby resulting in a scalar constraint violation $J_2 \in [0,1]$ which has the form of:

$$J_2 = \frac{1}{m}\sum_{i=1}^{m}\mu_g^i + \frac{1}{n}\sum_{j=1}^{n}\mu_{x_f}^j \tag{8}$$

The scalar constraint violation is considered as a separate objective function to be minimized.

By minimizing the objective functions given by Eq.(3) and Eq.(8), the original problem formulation has been transformed to an unconstrained bi-objective version. A compact form of this unconstrained bi-objection optimal control model is written as:

$$\begin{aligned} \min_{u(t)} \quad & J_1 = \int_{t_0}^{t_f} L(x(t),u(t))dt + \Phi(x(t_f),t_f) \\ \min_{u(t)} \quad & J_2 = \tfrac{1}{m}\sum_{i=1}^{m}\mu_g^i + \frac{1}{n}\sum_{j=1}^{n}\mu_{x_f}^j \end{aligned} \tag{9}$$

C. MOPSO Algorithm

To find a control sequence such that the two mission objectives considered in Eq.(9) are optimized, certain parameter optimization algorithms should be adopted. In this paper, we focus on the design and test of a modified MOPSO algorithm. MOPSO is a typical bio-inspired multi-objective optimization algorithm [29]. For the considered problem, each particle among the swarm represents a potential control sequence consisting of a position vector $\mathbf{z}$ and a velocity vector $\mathbf{v}$:

$$\begin{aligned} \mathbf{z}(s) &= [\mathbf{u}_1(s), \mathbf{u}_2(s), ..., \mathbf{u}_{N_j}(s)] \\ \mathbf{v}(s) &= [\mathbf{v}_1(s), \mathbf{v}_2(s), ..., \mathbf{v}_{N_j}(s)] \end{aligned} \tag{10}$$

In Eq.(10), N_j and $s = 1, 2, ..., N_s$ are, respectively, the size of the swarm and the index of the current iteration. For convenience reasons, we denote $\mathbf{z}_j,\ j = 1, 2, ..., N_j$ as the j-th component of $\mathbf{z}(s)$ in the rest of the paper. During the optimization iteration, the particle explores the searching area by introducing a recurrence relation:

$$\mathbf{z}(s+1) = \mathbf{z}(s) + \mathbf{v}(s+1) \tag{11}$$

where $\mathbf{v}(s+1)$ is given by:

$$\begin{aligned} \mathbf{v}(s+1) \quad &= w \cdot \mathbf{v}(s) \\ &+ c_1 r_1 \cdot (\mathbf{p}(s) - \mathbf{z}(s)) \\ &+ c_2 r_2 \cdot (\mathbf{g}(s) - \mathbf{z}(s)) \end{aligned} \tag{12}$$

Variables appeared in Eq.(12) are defined below:

- w: The inertia weight factor;
- $\mathbf{p}(s)$: The personal best position in the sth iteration;
- $\mathbf{g}(s)$: The global best position in the sth iteration;
- c_1, c_2: Factors reflecting strength of attraction;
- r_1, r_2: Two random constants on $(0, 1]$.

In terms of the personal best position of the jth particle $\mathbf{p}_j(s)$, this vector should be updated via

$$\mathbf{p}_j(s) = \begin{cases} \text{rand}\{\mathbf{p}_j(s-1), \mathbf{z}_j(s)\} & \text{if} \quad \mathbf{z}_j(s) \not\prec\not\succ \mathbf{p}_j(s-1) \\ \mathbf{p}_j(s-1) & \text{if} \quad \mathbf{z}_j(s) \prec \mathbf{p}_j(s-1) \\ \mathbf{z}_j(s) & \text{if} \quad \mathbf{z}_j(s) \succ \mathbf{p}_j(s-1) \end{cases} \tag{13}$$

where $\mathbf{p}_j(s-1)$ is the personal best position of the j-th particle at the $(s-1)$-th iteration. Here, the notation $\prec$ is the dominant relation determined by the concept of Pareto optimal, and $\mathbf{z}_1 \prec \mathbf{z}_2$ means $\mathbf{z}_1$ is dominated by $\mathbf{z}_2$. In Eq.(13), the character $\not\prec\not\succ$ means the mutually dominant relation. In this case, the algorithm randomly selects one of these two vectors.

The nondominated solutions are then collected to form an external archive $\mathbf{A}(s) = [\mathbf{z}_1(s), \mathbf{z}_2(s), ..., \mathbf{z}_{N_a}(s)]$, where $|\mathbf{A}(s)| = N_a$ stands for the number of nondominated solutions in the current archive and this number will be changed during the evolutionary process. Note that $N_a \leq N_A$, in which N_A represents the maximum size of the archive specified by the designer. To update $\mathbf{A}(s)$, the following algorithm (e.g., Algorithm 1) is performed. Note that in Algorithm 1, $|\cdot|$ stands for the size of a set. After performing $\mathbf{A}(s) = \mathbf{A}(s) \cup \mathbf{A}(s-1)$ in Algorithm 1, if the size of $\mathbf{A}(s)$ is greater than N_A, then we delete the most infeasible one or the less optimal one according to the value of J_1 or J_2 until the size reaches N_A.

Algorithm 1 Archive update process

Input: $\mathbf{A}(s-1)$ and $\mathbf{p}(s)$;
Output: $\mathbf{A}(s)$;
/*Main update process*/
for $j := 1, 2, ..., |\mathbf{p}(s)|$ **do**
  for $m := 1, 2, ..., |\mathbf{A}(s-1)|$ **do**
    if $\mathbf{z}_m \prec \mathbf{p}_j$ **then**
      Remove $\mathbf{z}_m$ from $\mathbf{A}(s-1)$
      Set an indicator $Ind = 1$
    else
      Break
    end if
  end for
  if $Ind \neq 1$ **then** ▷ //*No individual in $\mathbf{A}(s-1)$ dominates $\mathbf{p}_j$*//
    Add $\mathbf{p}_j$ to $\mathbf{A}(s)$
  end if
end for
Perform $\mathbf{A}(s) = \mathbf{A}(s) \cup \mathbf{A}(s-1)$
Output $\mathbf{A}(s)$
/*End archive update process*/

D. ε-Bias Selection

As investigated in the previous work [23], [24], the performance of using a heuristic algorithm to trajectory optimization problems might be significantly degraded if certain actions are not taken to put emphasis on the searching direction toward the feasible region. Consequently, we design a bias selection strategy, named ε-bias selection, to further update the external archive.

Prior to introduce the ε-bias selection strategy in detail, a reduced version of this strategy is firstly presented to illustrate the concept of bias selection. Subsequently, this reduced version will be extended to a more general one.

Let us consider two candidate particles $\mathbf{z}_1$ and $\mathbf{z}_2$ among the set $\mathbf{A}(s)$. Apparently, compared to J_1, the total degree of constraint violation J_2 has a higher priority and should be biased. If we denote the value of J_1 and J_2 for $\mathbf{z}_1$ and $\mathbf{z}_2$ as $J_1(\mathbf{z}_1)$, $J_1(\mathbf{z}_2)$, $J_2(\mathbf{z}_1)$, and $J_2(\mathbf{z}_2)$, then a constraint violation (CV)-based bias selection strategy can be designed. That is, the particle $\mathbf{z}_1$ is considered superior with respect to $\mathbf{z}_2$ if and only if the following CV-dominance conditions are triggered:

1) $(J_1(\mathbf{z}_1) < J_1(\mathbf{z}_2)) \wedge (J_2(\mathbf{z}_1) = J_2(\mathbf{z}_2) = 0)$;
2) $0 < J_2(\mathbf{z}_1) < J_2(\mathbf{z}_2)$;
3) $(J_2(\mathbf{z}_1) = 0) \wedge (J_2(\mathbf{z}_2) > 0)$.

The CV-based selection rules suggest that the comparison between particles should be made strictly according to the biased objective (e.g., J_2). In this way, the nondominated feasible candidate can always be preserved until a more optimal candidate is obtained. In the following, we extend the CV-based bias selection strategy to a more general ε-bias selection method. Specifically, in this strategy, the particle $\mathbf{z}_1$ is considered superior to another candidate $\mathbf{z}_2$ if the following ε-dominance conditions are triggered:

1) $(\mathbf{z}_2 \prec \mathbf{z}_1) \wedge (J_2(\mathbf{z}_1) \leq \varepsilon) \wedge (J_2(\mathbf{z}_2) \leq \varepsilon)$;
2) $\varepsilon < J_2(\mathbf{z}_1) < J_2(\mathbf{z}_2)$;
3) $(J_2(\mathbf{z}_1) \leq \varepsilon) \wedge (J_2(\mathbf{z}_2) > \varepsilon)$.

According to Eq.(9), it is obvious that $J_2 \in [0, 1]$. If $\varepsilon = 0$, then the ε-dominance conditions reduce to the CV-dominance conditions. On the contrary, if $\varepsilon = 1$, then the ε-dominance conditions are equivalent to the standard Pareto-dominance rules $\prec$ (no bias case) which are widely applied in the multi-objective approaches (see e.g., [26] and [27]). The value of $\varepsilon \in [0, 1]$ can be viewed as a balancing parameter able to adjust the degree between these two extreme cases. Here we present a simple adaptive formula in order to set ε:

$$\varepsilon = J_2^{max} - J_2^{min} \tag{14}$$

where J_2^{max} and J_2^{min} represent, respectively, the maximum and minimum J_2 values in the archive. At the beginning of the evolution, when all the particles are relatively far from the feasible border, ε tends to be small according to Eq.(14) and more emphasis/attention can be paid to the constraint violation. On the other hand, when some of the particles are close to the feasible border or they are already in the feasible region, ε tends to be larger such that J_1 and J_2 can be considered at the same time. Compared to the CV-based bias selection strategy, the extended ε-bias selection strategy offers more flexibility to simultaneously optimize the objective function and reduce the solution infeasibility. Note that the ε-bias selection strategy will be applied to update the external archive $\mathbf{A}(s)$ at the end of each iteration.

E. Local Exploration

Early works on developing MOPSO suggested that this algorithm has a strong global exploration ability [26], [28]. To also emphasize the local exploration of the searching process, a gradient-assisted operation can be introduced. It is worth noting that the combination of an evolutionary algorithm with a gradient-based method to improve the local search can be found in a number of previous works [30], [31]. Based on the reported results, it was verified that such a local update strategy has the capability of improving the quality of the final solution. Therefore, we introduce this approach to update the elements in the archive, thus making more progresses during the iteration.

Let us denote the directional derivative of the two objectives along $\mathbf{e}_m$ as

$$\begin{aligned} \nabla_{\mathbf{e}_m} J_1(\mathbf{z}_m) &= \lim_{\Delta \to 0} \left\{ \frac{J_1(\mathbf{z}_m + \Delta \cdot \mathbf{e}_m) - J_1(\mathbf{z}_m)}{\Delta} \right\} \\ \nabla_{\mathbf{e}_m} J_2(\mathbf{z}_m) &= \lim_{\Delta \to 0} \left\{ \frac{J_2(\mathbf{z}_m + \Delta \cdot \mathbf{e}_m) - J_2(\mathbf{z}_m)}{\Delta} \right\} \end{aligned} \tag{15}$$

in which $m = 1, ..., N_a$, and $\mathbf{z}_m \in \mathbf{A}(s)$. Δ stands for the step length. It was shown in [31] that the above two directional derivatives can be further written as:

$$\begin{aligned} \nabla_{\mathbf{e}_m} J_1(\mathbf{z}_m) &= (\nabla J_1(\mathbf{z}_m))^T \cdot \mathbf{e}_m \\ \nabla_{\mathbf{e}_m} J_2(\mathbf{z}_m) &= (\nabla J_2(\mathbf{z}_m))^T \cdot \mathbf{e}_m \end{aligned} \tag{16}$$

In Eq.(16), $\nabla J(\mathbf{z}_m)$ stands for the gradient of J with respect to $\mathbf{z}_m$. A direction vector $\mathbf{e}_m$ which descends both J_1 and J_2 can be obtained via

$$\mathbf{e}_m = -\left(\omega_1 \frac{\nabla J_1(\mathbf{z}_m)}{\|\nabla J_1(\mathbf{z}_m)\|} + \omega_2 \frac{\nabla J_2(\mathbf{z}_m)}{\|\nabla J_2(\mathbf{z}_m)\|} \right) \tag{17}$$

where the two weight coefficients hold $\omega_1 + \omega_2 = 1$, and $\omega_1 < \omega_2$. Subsequently, the elements among the current archive are updated via

$$\hat{\mathbf{z}}_m = \mathbf{z}_m + \Delta \cdot \mathbf{e}_m \tag{18}$$

It is important to remark that by viewing the definition of μ_g and μ_{x_f}, it is obvious that the resulting objective function J_2 may not be differentiable at some points. Hence, we replace their equations by a piecewise smooth form in practical applications. More precisely, the repair of μ_g is achieved by Eq.(19), where λ and κ are positive constants. Note that the repair of μ_{x_f} (e.g., $\varrho(\mu_{x_f}, \lambda, \kappa)$) can be obtained analogically. In this study, the gradient update process is performed in every E generation.

F. Evolution Restart Strategy

For some practical constrained optimization problems, complex constraints might be involved in the problem formulation. Due to the strong nonconvexity or nonlinearity of these constraints, the feasible searching space can be significantly restricted and the PSO algorithm is likely to stagnate in one of the local infeasible regions. In order to tackle this problem, we propose an evolution restart strategy.

The key component of this restart strategy is to determine whether the current archive has already got stuck in an infeasible region. Actually, this can be reflected by analyzing J_2 value for all the particles. If the difference of J_2 value between particles is small, then the current swarm is highly likely to converge to an infeasible region. More precisely, the following two conditions can be applied as an indicator of getting stuck in infeasible regions:

1) $\forall \mathbf{z}_m \in \mathbf{A}(s)$, $J_2(\mathbf{z}_m) \neq 0$,
2) The variance of $J_2(\mathbf{z}_m)$ is less than a restart threshold μ.

If these conditions are triggered, the evolution restart strategy will be executed. That is, all the particles among the swarm

$$\varrho(\mu_g,\lambda,n)=\begin{cases}1, & \mu_g>1+n;\\ (-\mu_g^2+2(1+\kappa)\mu_g-(\kappa-1)^2)/4\kappa, & 1-\kappa\le\mu_g\le 1+\kappa;\\ \mu_g, & \lambda<\mu_g<1-\kappa;\\ (\mu_g+\lambda)^2/4\lambda, & -\lambda\le\mu_g\le\lambda;\\ 0, & \mu_g<-\lambda.\end{cases} \tag{19}$$

are randomly re-generated on their searching space. Although evolution histories may contain valuable information and can potentially provide feedback so as to guide the optimization, determining whether these historical data are promising is still a challenging issue. In addition, a large amount of space should be pre-allocated to store these historical data. Hence, we decide to simply discard these data and restart the evolution by randomly re-initializing all the particles in the swarm.

G. Overall Algorithm Framework

For the proposed algorithm, the global best particle $\mathbf{g}(s)$ is selected from the updated archive $\mathbf{A}(s)$. Note that in the transformed problem formulation, J_1 is the primary mission objective to be optimized, while J_2 reflects the constraint violation of the solution. Therefore, $\mathbf{g}(s)$ can be selected by following the procedures specified in Algorithm 2.

Algorithm 2 $\mathbf{g}(s)$ selection process

Input: $\mathbf{A}(s)$;
Output: $\mathbf{g}(s)$;
/*Main process*/
Initialize $\mathbf{F}(s)=\{\}$ and $\mathbf{IF}(s)=\{\}$
for $m:=1,2,...,|\mathbf{A}(s)|$ **do**
  if $J_2(\mathbf{z}_m)>0$ **then**
    $\mathbf{IF}(s)=\mathbf{IF}(s)\cup\mathbf{z}_m$
  else
    $\mathbf{F}(s)=\mathbf{F}(s)\cup\mathbf{z}_m$
  end if
end for
if $\mathbf{F}(s)=\varnothing$ **then**
  $\mathbf{g}(s)=\arg\min_{\mathbf{z}_m\in\mathbf{IF}(s)}J_2(\mathbf{z}_m)$
else
  $\mathbf{g}(s)=\arg\min_{\mathbf{z}_m\in\mathbf{F}(s)}J_1(\mathbf{z}_m)$
end if
Output $\mathbf{g}(s)$
/*End the process*/

In summary, the conceptual block diagram of the proposed MOPSO-based trajectory optimization algorithm is visualized in Fig. 1. In order to clearly present how the optimization process is executed, the general steps are summarised in the pseudocode (see Algorithm 3). Note that in Step 9 of Algorithm 3, the evolutionary process is terminated when either of the following two rules can be triggered:

1) $s\ge N_s$;
2) $\forall\mathbf{z}_m\in\mathbf{A}(s)$, $J_2(\mathbf{z}_m)=0$ and the difference of $\mathbb{E}(J_2)$ between two consecutive iterations (e.g., the s-th and the $(s-1)$-th iteration) is less than a tolerance value ϵ.

In the second rule, $\mathbb{E}(\cdot)$ outputs the expectation value and this rule indicates no further improvement can be made among feasible solutions.

Algorithm 3 General steps for the optimization process

Input: The algorithm parameters w, r_1, r_2, c_1, c_2, ω_1, ω_2, Δ, N_j, $s=1$, and N_s;
Output: The final archive $\mathbf{A}(s)$;
/*Main optimization iteration*/
Step 1: Randomly initialize the position and velocity vectors of the particles;
Step 2: Obtain the state trajectory using numerical integration;
Step 3: Calculate the two objective values for all particles among the current swarm;
Step 4: Apply the nondominant sorting and **Algorithm 1** to construct and update the archive $\mathbf{A}(s)$;
Step 5: Update $\mathbf{A}(s)$ via the gradient-assisted local exploration;
Step 6: Perform the ε-bias selection process to update $\mathbf{A}(s)$;
Step 7: Search the global best particle $\mathbf{g}(s)$ from $\mathbf{A}(s)$ via **Algorithm 2**;
Step 8: Update the velocity and position vectors of the particles;
Step 9: Check whether the termination condition is triggered?
  if not, set $s=s+1$ and return back to Step 2.
Step 10: Terminate the optimization and output the final archive $\mathbf{A}(s)$;
/*End optimization iteration*/

III. Constrained Atmospheric Entry Problem

In this section, an optimal control formulation of the constrained atmospheric entry trajectory design problem is detailed. Specifically, the system dynamics used to describe the motion of the spacecraft are formulated in Section III.A. Following that, a number of entry boundary and path constraints are constructed in Section III.B. Finally, the mission objectives selected to assess the performance of the entry maneuver are introduced in Section III.C.

A. System Model

The following set of first order differential equations can be used for describing the motion of the entry vehicle:

$$\dot{x}=\begin{bmatrix}V\sin\gamma\\ \frac{V\sin\psi\cos\gamma}{r\cos\phi}\\ \frac{V\cos\psi\cos\gamma}{r}\\ -\frac{D}{m}-g\sin\gamma\\ \frac{L\cos\sigma}{mV}+(\frac{V^2-gr}{rV})\cos\gamma\\ \frac{L\sin\sigma}{mV\cos\gamma}+\frac{V}{r}\sin\psi\cos\gamma\tan\phi\\ K_\sigma\end{bmatrix}x+\begin{bmatrix}0\\0\\0\\0\\0\\0\\-K_\sigma\end{bmatrix}u \tag{20}$$

where the system state variables are defined as $x=[x_p,x_a,\sigma]^T\in\mathbb{R}^7$. Here, $x_p=[h,\theta,\phi]^T\in\mathbb{R}^3$ determines the

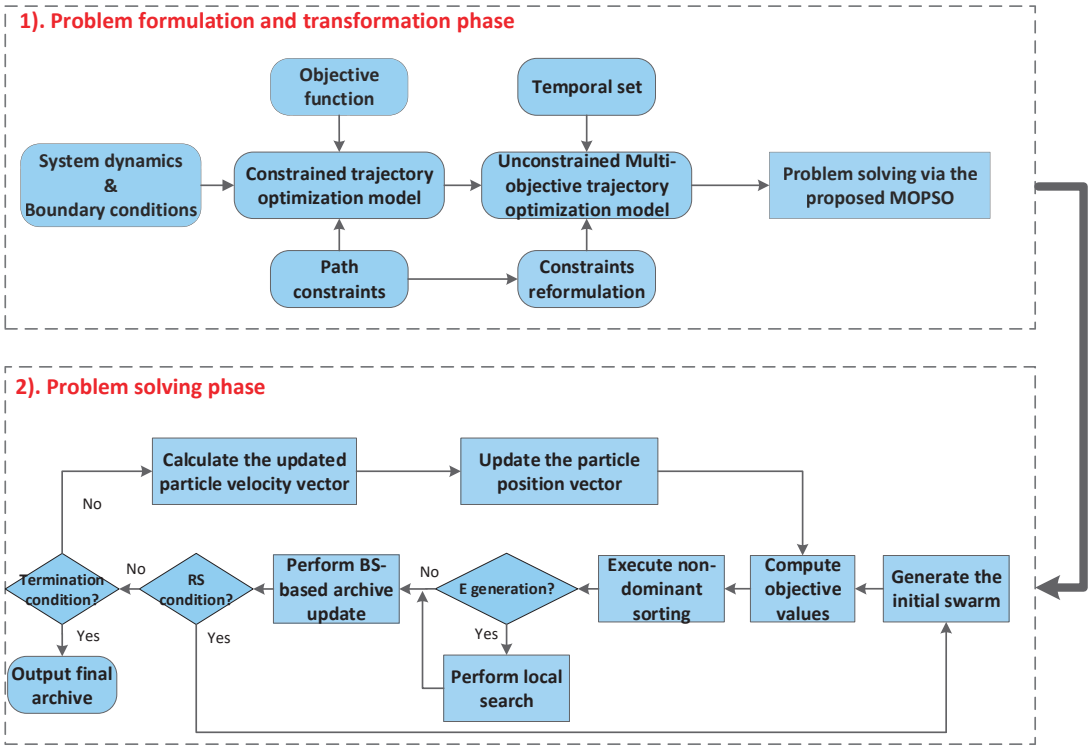

Fig. 1: Conceptual block diagram of the proposed algorithm

3-D position of the entry vehicle, consisting of the altitude h, longitude θ and latitude ϕ, respectively. The radius distance r can be obtained via $r = h + R_e$, where R_e denotes the radius of the Earth. The components of $x_a = [V, \gamma, \psi]^T \in \mathbb{R}^3$ stand for the velocity, flight path angle (FPA), and the heading angle of the entry vehicle, respectively. $[\sigma, \sigma_c]$ denotes the actual and demanded bank angle profiles, and the control variable is assigned as $u = \sigma_c$. The physical meaning of other variables/parameters appeared in Eq.(20), together with their values or calculation equations, can be found in Table I.

TABLE I: Variable Definitions

Variables	Calculation/values
g: gravity	$g = \frac{\mu}{r^2}$
r: radius distance	$r = h + R_e$
ρ: atmospheric density	$\rho = \rho_0 exp(-h/H)$
D: drag force	$D = \frac{1}{2}\rho C_D V^2$
L: lift force	$L = \frac{1}{2}\rho C_L V^2$
C_D: drag coefficient:	$C_D = C_{D_0} + C_{D_1}\alpha + C_{D_2}\alpha^2$
C_L: lift coefficient:	C_L=$C_{L_0} + C_{L_1}\alpha$
S: reference area	S=250m^2
R_e: radius of the Earth	R_e=6371.2km
ρ_0: sea-level air density	ρ_0=1.2256kg/m^3
H: density scale height	H=7.25km
m: mass	m=92073kg
μ: gravitational parameter	μ=398603.2km^3/s^2

B. Entry Phase Constraints

During the planetary entry flight, the following four types of constraints are required to be satisfied:

1) Safety corridor constraints;
2) Variable terminal boundary constraints;
3) State and control path constraints;
4) Angular rate constraints.

1) Safety corridor constraints: To protect the structure of the entry vehicle, the aerodynamic heat transfer rate Q, the dynamic pressure P, and the load factor N_L must be restricted to certain safety corridors during the entire flight. This can be expressed by:

$$0 \le Q(r, V, \alpha) \le \bar{Q} \tag{21}$$

$$0 \le P(r, V) \le \bar{P} \tag{22}$$

$$0 \le N_L(r, V) \le \bar{N} \tag{23}$$

In Eqs.(21)-(23), the permissable peak values of $(\bar{Q}, \bar{P}, \bar{N})$ are set to $(125, 280, 2.5)$. The heat transfer rate Q is a function of radial distance r, velocity V and angle of attack (AOA) α, whereas the dynamic pressure P and load factor N_L are mainly determined by the radial distance r and the velocity V. The value of α (in degree) can be computed via the following equation [32]:

$$\alpha = \begin{cases} 40 - w_1(V - \hat{V})^2/340^2, & \text{if } V < \hat{V}; \\ 40, & \text{if } V \ge \hat{V}. \end{cases} \tag{24}$$

in which $\hat{V} = 4570$m/s, and the value of w_1 is equal to 0.20705.

It is worth noting that in Eq.(21), the heat transfer rate Q consists of two major components:

$$Q(r, V, \alpha) = Q_r(\alpha) \cdot Q_d(r, V) \tag{25}$$

in which $Q_r(\alpha)$ stands for the aerodynamic heat flux and is calculated via Eq.(26), whereas $Q_d(r, V)$ represents the radiation heat transfer and is given by Eq.(27).

$$Q_r(\alpha) = q_0 + q_1\alpha + q_2\alpha^2 + q_3\alpha^3 \tag{26}$$

$$Q_d(r,V) = k_Q \rho^{0.5} V^{3.07} \tag{27}$$

In Eq.(26), (q_0, q_1, q_2, q_3)=$(1.067, -1.101, 0.6988, -0.1903)$. Furthermore, the dynamic pressure and load factor can be calculated via Eq.(28) and Eq.(29), respectively.

$$P(r,V) = \tfrac{1}{2}\rho V^2 \tag{28}$$

$$N_L(r,V) = \tfrac{\sqrt{L^2+D^2}}{mg} \tag{29}$$

2) Boundary constraints: The terminal boundary constraints are imposed such that the flight states can reach specific values at t_f in order to start the terminal area energy management phase [11], [16]. Specifically, the altitude and flight path angle are required to satisfy

$$\begin{array}{l} |h(t_f) - h_f| \le \varepsilon_{h_f} \\ |\gamma(t_f) - \gamma_f| \le \varepsilon_{\gamma_f} \end{array} \tag{30}$$

where $h_f = 30km$ and $\gamma_f = -5^\circ$ are the targeted terminal altitude and flight path angle values, respectively. $\varepsilon_{h_f} = 500$m and $\varepsilon_{\gamma_f} = 0.1^\circ$ stand for the permissable errors. Moreover, the terminal velocity is required to satisfy

$$V_f^{min} \le V(t_f) \le V_f^{max}$$

where V_f^{min} and V_f^{max} are set to 900m/s and 1100m/s, resulting in $V(t_f) \in [900, 1100]m/s$.

3) State and control path constraints: The state and control path constraints are imposed such that system state and control variables can be constrained within tolerant regions during the entire entry flight (e.g., $\forall t \in [0, t_f]$). These constraints can be written as:

$$\begin{array}{ll} \underline{h} \le h(t) \le \bar{h} & \underline{\theta} \le \theta(t) \le \bar{\theta} \\ \underline{\phi} \le \phi(t) \le \bar{\phi} & \underline{V} \le V(t) \le \bar{V} \\ \underline{\gamma} \le \gamma(t) \le \bar{\gamma} & \underline{\psi} \le \psi(t) \le \bar{\psi} \\ \underline{\sigma} \le \sigma(t) \le \bar{\sigma} & \underline{\sigma_c} \le \sigma_c(t) \le \bar{\sigma}_c \end{array} \tag{31}$$

where $\underline{x} = [\underline{h}, \underline{\theta}, \underline{\phi}, \underline{V}, \underline{\gamma}, \underline{\psi}, \underline{\sigma}]$ and $\underline{u} = \underline{\sigma_c}$ denote the lower bounds of x and u, while $\bar{x} = [\bar{h}, \bar{\theta}, \bar{\phi}, \bar{V}, \bar{\gamma}, \bar{\psi}, \bar{\sigma}]$ and $\bar{u} = \bar{\sigma}_c$ represent the upper bounds with respect to x and u, respectively.

4) Angular rate constraints: Early studies suggested that compared to the position and velocity profiles, more oscillations can be found in the angular variable trajectories, which is usually not desirable [11], [16]. Therefore, different from some existing research works [11], [16], [17], [25], the angular rate constraints are also considered in this work such that the evolution of the corresponding angular variables can become smoother. Specifically, these constraints can be modeled as:

$$\begin{array}{l} \underline{\dot{\gamma}} \le \dot{\gamma}(t) \le \bar{\dot{\gamma}} \\ \underline{\dot{\psi}} \le \dot{\psi}(t) \le \bar{\dot{\psi}} \\ \underline{\dot{\sigma}} \le \dot{\sigma}(t) \le \bar{\dot{\sigma}} \end{array} \tag{32}$$

in which the values of $[\underline{\dot{\gamma}}, \underline{\dot{\psi}}, \underline{\dot{\sigma}}]$ and $[\bar{\dot{\gamma}}, \bar{\dot{\psi}}, \bar{\dot{\sigma}}]$ are assigned as $[-0.5^\circ, -0.5^\circ, -0.5^\circ]$/s and $[0.5^\circ, 0.5^\circ, 0.5^\circ]$/s, respectively. Imposing these constraints might be helpful for some particular uses of the entry vehicle such as the reginal reconnaissance [14] and payload delivery [33]. Since the angular trajectories are less likely to have instantaneous variations, the information gathering of inaccessible areas or high-precision payload delivery tends to be much easier.

C. Objectives

The atmospheric entry mission is established as an optimization problem. Different performance indices reflecting the quality of the entry flight are formulated in objective functions. For example:

- An efficiency-related measure can be designed by minimizing the flight time duration (e.g., the terminal time instant t_f. That is,

$$\text{Obj}_1 = \min t_f \tag{33}$$

- A safety-related measure can be selected by minimizing the total amount of aerodynamic heat. That is,

$$\text{Obj}_2 = \min \int_{t_0}^{t_f} Q dt \tag{34}$$

- An entry capability-related measure can be designed by maximizing the cross range (e.g. the terminal $\phi(t_f)$). That is,

$$\text{Obj}_3 = \max \phi(t_f) \tag{35}$$

- An energy-related measure can be designed by minimizing the terminal kinetic energy, which is written as

$$\text{Obj}_4 = \min V(t_f) \tag{36}$$

In the later simulation result section, all the above objectives will be considered separately.

IV. Test Results and Analysis

A. Test Case Specification

To carried out the simulations, algorithm-related parameters are firstly assigned. Specifically, r_1 and r_2 are randomly generated on the interval $[0, 1]$. $[c_1, c_2]$ is set to $[1.49445, 1.49445]$, while w is calculated via $w = (1 + r_1)/2$. ω_1 and ω_2 are set to 0.3 and 0.7, respectively. $[N_j, N_s, N_k]$ is assigned as $[40, 2000, 100]$. $\epsilon = 10^{-6}$. The demanded bank angle is randomly initialized within the region $\sigma_c \in [-90, 1]^\circ$. The proposed algorithm is performed on a PC with Intel Quar-Core i7-4790 CPU (8GB RAM).

For the considered atmospheric entry problem, four test cases are investigated. Case i stands for minimizing $J_1 = \text{Obj}_i$, $i \in \{1, 2, 3, 4\}$ while simultaneously satisfying all types of constraints. By carrying out the reformulation process shown in Fig.1, the total amount of constraint violation value is considered as an additional objective function J_2 for each mission case, thereby resulting in four unconstrained bi-objective formulations.

To highlight the advantage of using the proposed design, comparative studies were performed between the biased MOPSO approach and other well-developed trajectory optimization algorithms. For example, a PSO-based trajectory optimization method suggested in [16], [17], together with an artificial bee colony-based trajectory planning algorithm reported in [25], is chosen for the comparative study. It is worth noting that for these two evolutionary methods, the penalty function strategy is applied to deal with constraints

existing in the optimization model. In addition, the evolution restart strategy introduced in Section II.F is also applied in these methods. We abbreviate these two methods as PFPSO and PFABC, respectively.

B. Performance of Different Methods

Optimal results calculated by applying different heuristic trajectory optimization algorithms for a single trial are firstly presented and analyzed in this subsection. The optimized state/control evolutions, along with the corresponding path constraint history, are visualized in Figs.2-5 for the four entry mission cases.

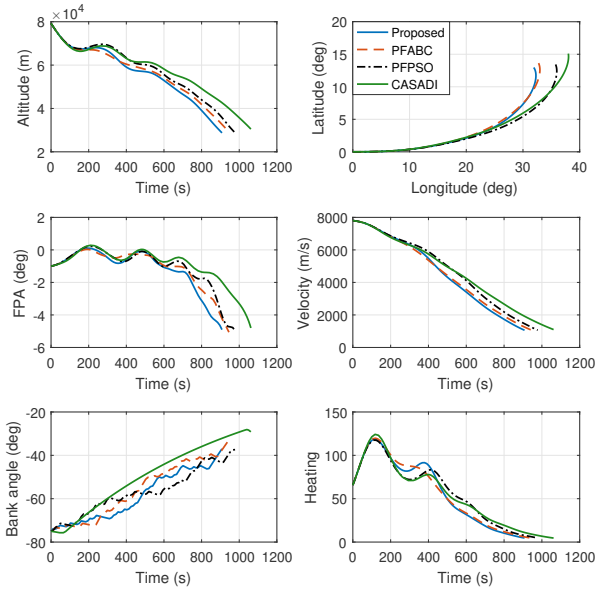

Fig. 2: State/control/constraint evolutions: Case 1

As can be observed from Figs.2-5, the pre-specified entry terminal boundary conditions and safety-related path constraints can be satisfied for all the mission cases, thereby confirming the validity of the investigated heuristic methods. That is, both the penalty function-based and the multi-objective transformation-based constraint handling strategies are able to guide the searching direction toward the feasible region. In terms of the flight trajectories, same trend can be observed from the solutions generated by different heuristic algorithms for all the considered mission cases. Moreover, for the third mission case, the three evolutionary methods can produce almost identical solutions. Moreover, by viewing the system state and control profiles, it is clear that the obtained trajectories are relatively smooth. This can be attributed to the differential equation imposed on the actual bank angle variable σ. This equation can also be understood as a first-order filter and it indirectly restricts the rate of the actual bank angle.

A comparison is also made between the proposed method and another numerical optimal control solver, named CASADI [34], for solving the constrained atmospheric entry problem (the interior point solver IPOPT [35] is applied in CASADI).

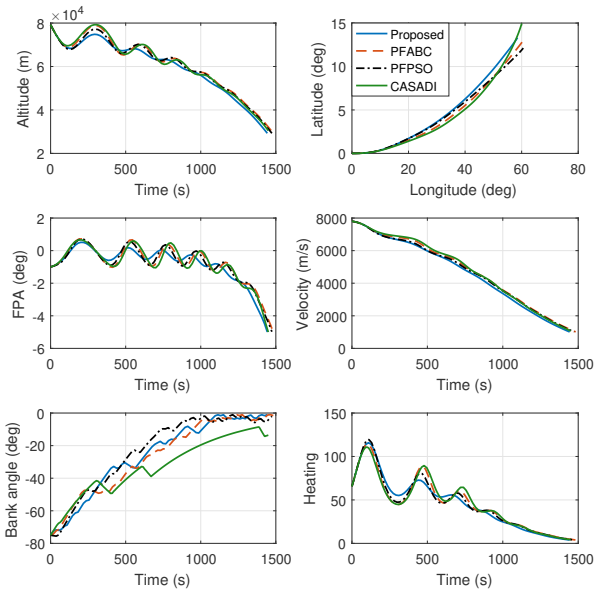

Fig. 3: State/control/constraint evolutions: Case 2

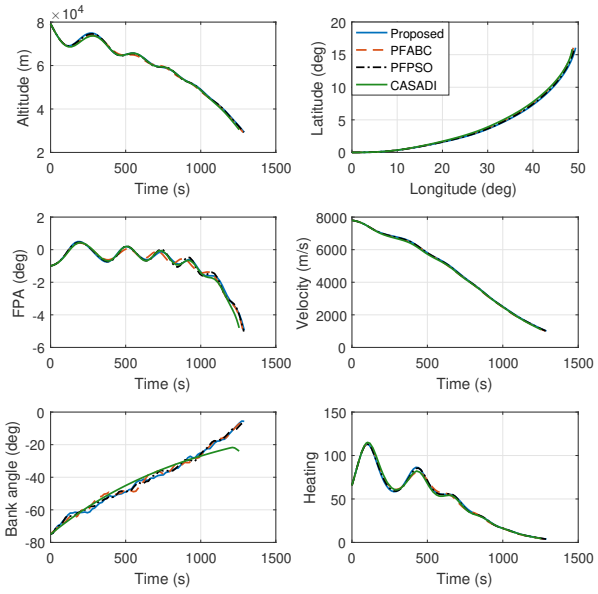

Fig. 4: State/control/constraint evolutions: Case 3

This solver has become increasingly popular and it has been applied in the literature to address a number of motion planning or trajectory optimization problems [36], [37]. The four mission cases are solved using CASADI with $\epsilon = 10^{-6}$ as the optimization tolerance. The optimized trajectories are visualized in Figs.2-5.

As can be viewed from Figs.2-5, the trajectories produced by the proposed method and CASADI are comparable and generally follow a same trend. However, CASADI has its unique features. For example, compared to the developed approach, CASADI is able to produce much smoother bank

TABLE II: Optimal results obtained via different methods

Case No.	PFPSO		PFABC		Proposed		CASADI	
	J_1	J_2	J_1	J_2	J_1	J_2	J_1	J_2
Case.1	977.85	0	945.32	0	**908.23**	**0**	1061.15	0
Case.2	40212	0	40613	0	**39591**	**0**	40018	0
Case.3	16.066	0	16.060	0	**16.068**	**0**	15.839	0
Case.4	947.50	0	964.88	0	**936.16**	**0**	1086.81	0

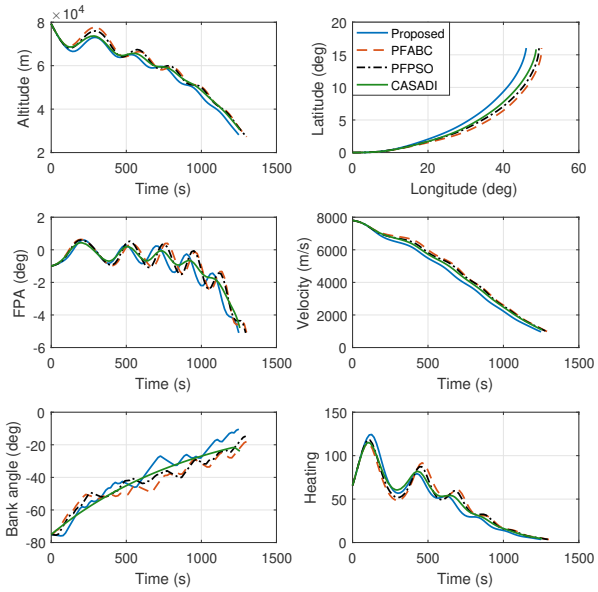

Fig. 5: State/control/constraint evolutions: Case 4

angle profiles for all the mission cases. This is apparent from Figs.2-5, where more oscillations can be identified on the bank angle trajectories generated using the proposed method and other heuristic methods. This is mainly due to the randomness of the evolutionary process.

To provide a clear demonstration of the performance achieved via different optimization methods, quantitative results for cases 1-4 are tabulated in Table II. It should be noted that to only compare J_1 is not that relevant in cases where solutions can be infeasible. It is trivial that the objective function value can be improved if infeasible solutions can occur. Hence, in the comparison shown in Table II, J_1 and J_2 are pairwise compared between different methods (e.g., a J_1 value is associated with a J_2 value, and it is relevant to compare this pair with other pairs).

From the solution pairs displayed in Table II, although the relative differences are in general not really significant, it can be seen that using the proposed approach is able to achieve better solutions with more optimal objective values for all the considered mission cases. Note that for mission case 3, the aim is to maximize the final latitude value. Hence, a larger objective value is desired for this mission case. According to the reported solution pairs and trajectory profiles, no constraint defined in Section III.B is violated, thus guaranteing the effectiveness of both the CASADI and the proposed methods. More importantly, based on these results, one can rule out that the better J_1 values are not the result of infeasible solutions.

In addition, multiple trials were executed to compare the convergence ability and robustness of the proposed approach and the CASADI. Specifically, 100 independent trials were performed using the proposed method with randomly initialized swarms. Similarly, 100 trials were performed using CASADI by specifying different initial guess values. The resulting solution pairs for these two methods are collected to generate the histograms (as displayed in Fig. 6 and Fig. 7) such that the relative difference between these two methods in terms of J_1 and J_2 can be clearly shown.

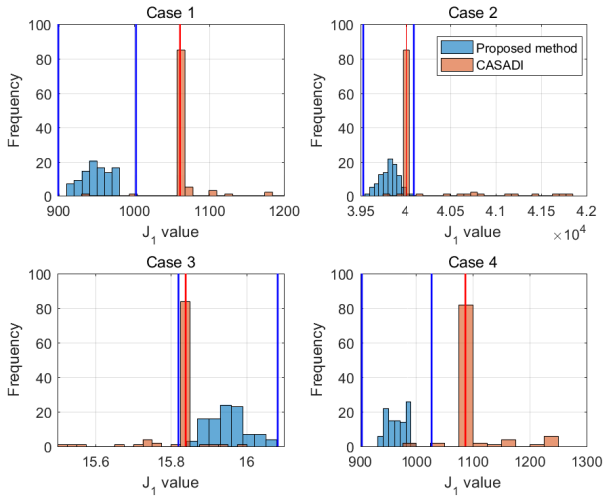

Fig. 6: Histograms of J_1 for the two methods

In Fig. 6, the lower and upper outlier boundaries for the proposed approach and CASADI are indicated by the blue and red vertical lines, respectively. From the results presented in Fig. 7, it is obvious that the proposed approach is able to drive the candidate solution to the feasible region for all the trials. As for CASADI, on the other hand, outliers can be found in the obtained J_2 histograms, indicating that CASADI converges to the local infeasible solution for multiple times. In addition, by viewing the corresponding J_1 histograms, a number of outliers can also be detected. This can be attributed to the result of infeasible solutions. Hence, for the considered problem, CASADI tends to be sensitive with respect to the initial guess values and has a greater possibility of converging to local infeasible solutions. From this point of view, benefiting from the evolution restart strategy developed in Section II.F, the proposed approach is more robust than its counterpart.

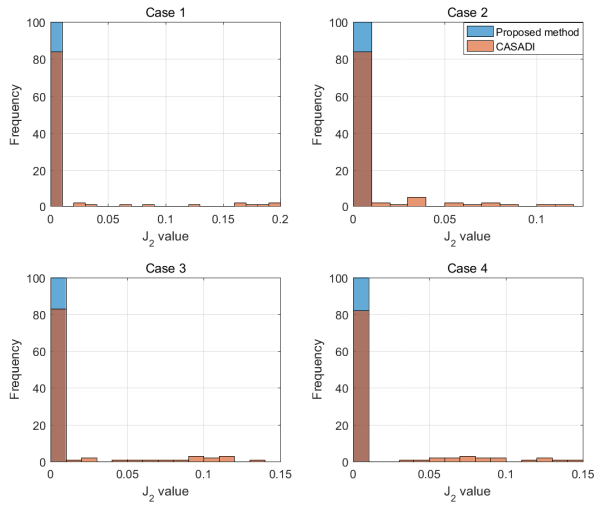

Fig. 7: Histograms of J_2 for the two methods

C. Convergence Analysis for Evolutionary Methods

In this subsection, we focus on the analysis of convergence performance of different evolutionary trajectory optimization methods investigated in this paper. Specifically, attention is given to the evolution histories of J_1 as well as J_2 for the considered four mission cases. Firstly, the average value of J_2 for each optimization iteration is presented in Fig. 8.

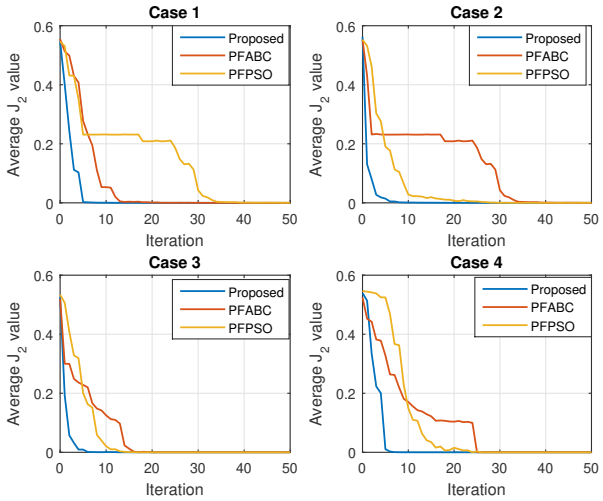

Fig. 8: J_2 evolutions: Cases 1-4

As can be seen from Fig.8, the proposed multi-objective approach tends to result in faster J_2 convergence histories for the four mission cases in comparison to the PFABC and the PFPSO algorithms. Specifically, by applying the proposed method, the number of optimization iterations required to steer the average value of J_2 to zero is less than or equal to 10 for all mission cases. While for other methods, this number becomes almost double. The J_2 evolution trajectories highlight the fact that the proposed approach is able to quickly locate the feasible solution and drive the current swarm/population moving toward the feasible region.

Next, the average value of J_1 for each optimization iteration is presented in Fig. 9.

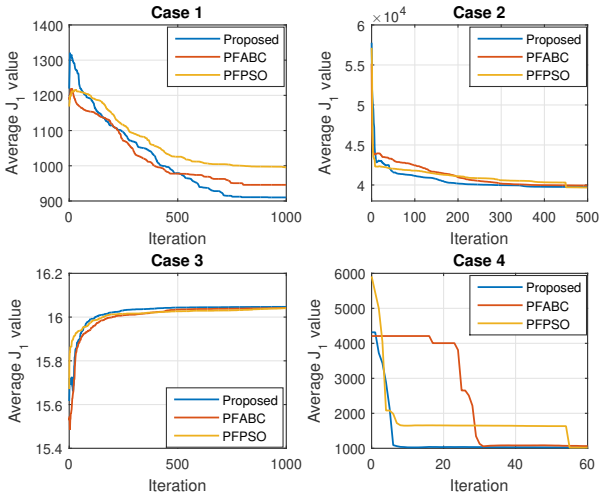

Fig. 9: J_1 evolutions: Cases 1-4

Similar to the results presented in Fig. 8, the proposed multi-objective approach has the capability of producing faster J_1 convergence histories for all the considered mission cases in comparison to its counterparts. More precisely, the average J_1 cost value achieved via the proposed algorithm converges to a more optimal steady value in less optimization iterations. Then this average J_1 value remains almost the same and does not decrease significantly in later optimization iterations. To clearly illustrate this behaviour, convergence results for mission case 4 are partly extracted and presented. For instance, by limiting the maximum number of iterations to 60, the history of the average J_1 value is shown in the last subfigure of Fig. 9. From this subfigure, it is apparent that enhanced convergence performance can be obtained by applying the proposed algorithm for solving the constrained atmospheric entry trajectory optimization problem.

D. Computational Performance of Different Methods

Apart from the results compared in terms of algorithm iterations, performance comparison between the new algorithm and other methods should also be done in terms of the computational cost. To achieve this, attention is firstly given to the computational times required by CASADI and the proposed method. Note that an important parameter which can have an impact on the resulting computational times is the index of optimization tolerance ϵ. By specifying different ϵ levels, mission case 1 to case 4 are re-performed and the average computational results of 50 successful runs are tabulated in Table III.

According to the data shown in Table III, it is obvious that the computational times required by the proposed method are generally less than that of CASADI except for mission case 1 and mission case 3 when ϵ is set to 10^{-6}. In addition,

TABLE III: Computational performance of CASADI and the proposed method (in seconds)

Case No.	Proposed			CASADI		
	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$	$\epsilon = 10^{-8}$
Case.1	102.24	108.37	113.15	87.86	144.34	208.87
Case.2	32.38	34.33	36.72	39.98	79.98	114.52
Case.3	50.14	54.56	58.25	41.62	82.63	125.56
Case.4	19.42	21.21	23.08	25.47	53.86	82.35

the computational performance of CASADI is more sensitive with respect to ϵ in comparison with the proposed approach. That is, the computation times required by CASADI tend to largely increase as ϵ becomes tighter. By contrast, only a slight increase of the computation time can be seen from the reported results for the proposed global exploration-based approach.

As for different evolutionary algorithms studied in this paper, we design the comparative experiments by introducing three indicators. These indicators can reflect the computational cost required by the evolutionary algorithms from different aspects:

- T_1: The average computation time required for different evolutionary algorithms to find the first feasible solution.
- T_2: The average computation time required for different evolutionary algorithms to drive the entire population to the feasible region.
- T_3: The average computation time required for different evolutionary algorithms to drive the average J_1 value of all feasible solutions to reach a certain level $\hat{J}_1$.

For mission cases 1-4, we assign the $\hat{J}_1$ values as $(1000, 41000, 16, 1000)$. 50 independent runs were executed for all the four mission cases and the average results are tabulated in Table IV.

In fact, due to the implementation of local exploration process, ε-bias selection method, and evolution restart strategy, the proposed method performs additional steps and tends to be more costly at each iteration. However, certain benefits can be obtained by performing these additional steps. For example, as can be observed in Table IV, the proposed approach tends to be less time-consuming than its counterparts in terms of finding the first feasible solution and driving the entire population to the feasible region for all the mission cases. Moreover, compared to other algorithms, the proposed approach can rapidly drive the candidate solution set to achieve a desirable level. As such, the effectiveness and advantages of performing these additional steps can be appreciated.

E. Impact of the Bias Selection Strategy and Local Exploitation Process

In previous subsections, it is illustrated that we can achieve better final solutions by applying the proposed method in comparison to other trajectory optimization planners. However, it is still not clear whether the implementation of the proposed bias selection strategy as well as the gradient-based local exploration method is able to make contributions to the problem solving process. Therefore, new experiments are designed to further test the impact of applying the bias selection strategy and the local exploration method. Two additional experiments are performed:

- Experiment 1: We compare the results produced by applying the proposed algorithm with and without the bias selection strategy.
- Experiment 2: We compare the results produced by applying the proposed algorithm with and without the local exploration method.

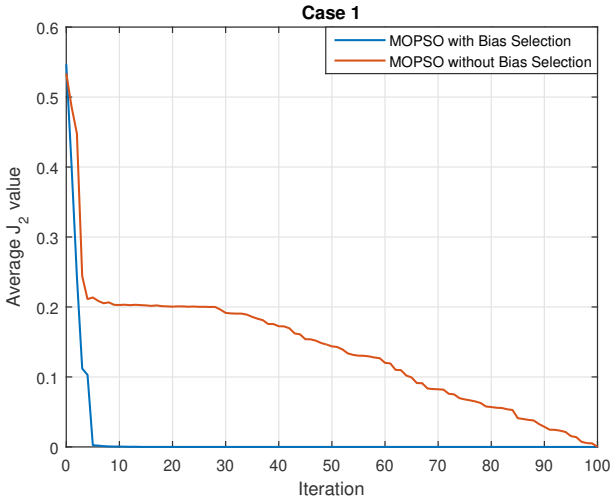

Fig. 10: Experiment 1: J_2 results for mission case 1

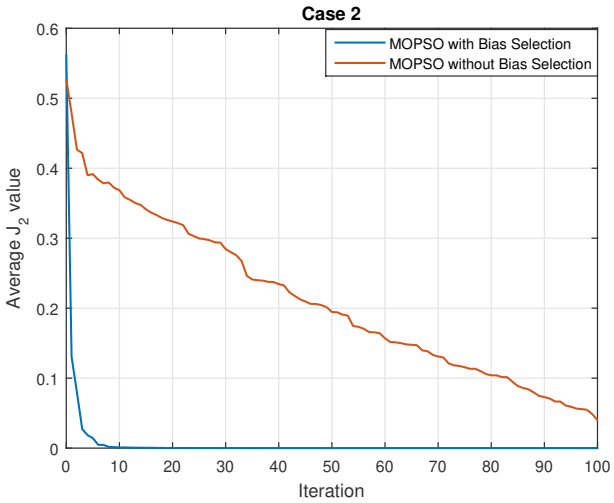

Fig. 11: Experiment 1: J_2 results for mission case 2

The average constraint violation histories (e.g., J_2 evolutions) for mission case 1 and case 2 are presented in Fig. 10 and Fig. 11, respectively. According to the presented J_2 trajectories, it is evident that without using the bias selection strategy, the proposed algorithm does not work as well as the

TABLE IV: Computational performance of different evolutionary methods

Case No.	Proposed			PFABC			PFPSO		
	T_1(s)	T_2(s)	T_3(s)	T_1(s)	T_2(s)	T_3(s)	T_1(s)	T_2(s)	T_3(s)
Case.1	0.64	5.23	63.25	1.62	8.64	64.36	2.05	43.27	125.43
Case.2	0.78	7.44	26.68	2.78	15.53	37.50	1.82	15.15	51.15
Case.3	0.73	6.86	25.74	2.36	8.89	31.26	1.44	8.58	30.08
Case.4	0.77	6.89	17.91	2.29	13.21	18.72	1.58	13.26	18.36

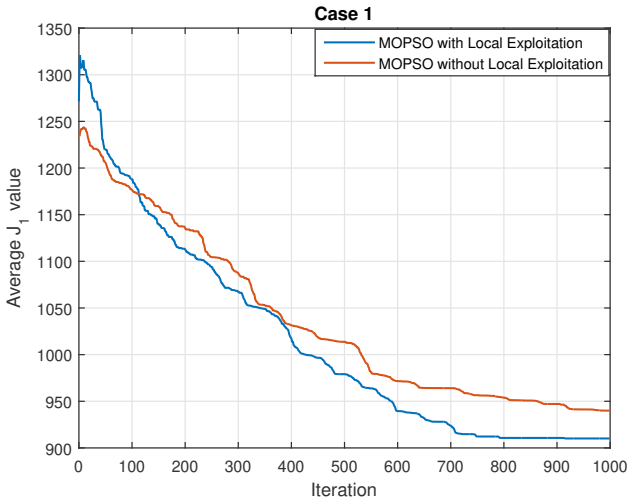

Fig. 12: Experiment 2: J_1 results for mission case 1

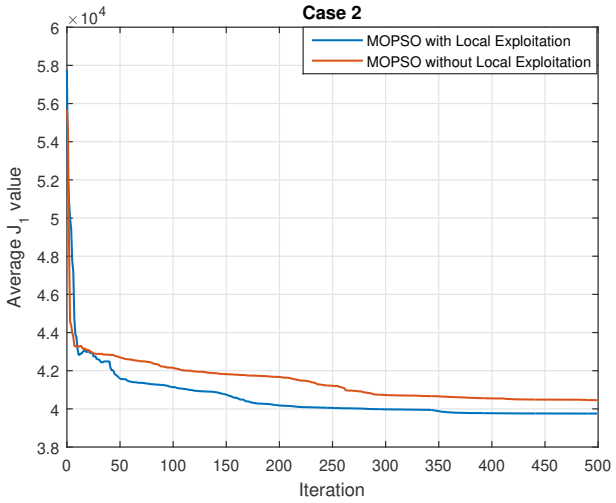

Fig. 13: Experiment 2: J_1 results for mission case 2

one equipped with the bias selection strategy. More specifically, after executing a large number of iterations, there are still infeasible solutions among the swarm. In addition, the J_2 convergence history tends to be much slower if a search bias is not introduced to the multi-objective trajectory optimization process. Hence, we can conclude that the implementation of the bias selection strategy is able to have positive influences for guiding the multi-objective optimization process to find more promising solutions.

As for the experiment 2, the corresponding J_1 evolution results for the entry mission case 1 and 2 are illustrated in Fig. 12 and Fig. 13, respectively. From the trajectories displayed in Fig. 12 and Fig. 13, it is clear that MOPSO equipped with the local exploration method can quickly steer J_1 to a more optimal steady value for the considered atmospheric entry mission cases. Consequently, we can conclude that it is beneficial to apply the gradient-based local exploration method to update the candidate set during the optimization iteration. Note that for experiment 1 and experiment 2, similar conclusions can also be made for mission cases 3-4. So we omit the the presentation of their results for space reasons.

F. Impact of the Restart Strategy

In this subsection, we are interested in studying and testing the effectiveness of the restart strategy (RS) proposed in Section II.F. Experiments were designed by comparing the results produced via the proposed algorithm with and without this strategy. Note that empirical studies were carried out and the value of the restart threshold is set to 10^{-3} throughout the simulation.

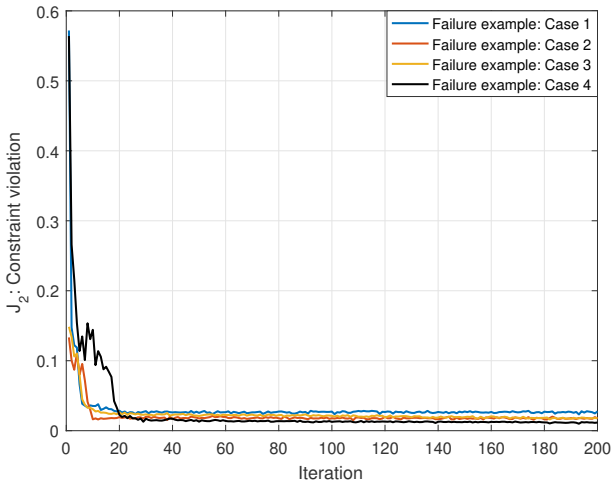

Fig. 14: Failure case examples

100 independent runs were performed for the four mission cases and statistical results including the average value of the primary objective (denoted as mean(J_1)), the average constraint violation value of failure cases (denoted as mean(J_2)), times of infeasible solution converged (denoted as T_d), and the successful rate (computed via $r_s = 1 - T_d/100$) are tabulated in Table V.

In addition, Fig. 14 illustrates examples of convergence failure (collected from Table V) for the four mission cases. As can be seen from Fig. 14, the J_2 evolution trajectories for different mission cases converge to a value which is above zero. This further confirms that the evolution process has the

TABLE V: Results obtained with and without RS

Case No.	Proposed method with RS			
	mean(J_1)	mean(J_2)	T_d	r_s
Case.1	944.26	0	0	100%
Case.2	39801	0	0	100%
Case.3	15.952	0	0	100%
Case.4	963.35	0	0	100%
Case No.	Proposed method without RS			
	mean(J_1)	mean(J_2)	T_d	r_s
Case.1	967.86	0.031	9	91%
Case.2	40168	0.028	9	91%
Case.3	15.902	0.027	6	94%
Case.4	983.77	0.025	13	87%

possibility to get stuck in local infeasible regions for different reentry mission cases.

By analyzing the results presented in Table V, one can conclude that better solution pairs (e.g., J_1 and J_2 values) are obtained if the proposed algorithm is equipped with the restart strategy. On the contrary, the proposed algorithm without applying the restart strategy is more likely to converge to local infeasible regions. This is mainly reflected by the reported successful rate and times of infeasible solution converged. In summary, based on the reported results, more optimal solutions and enhanced convergence performance can be achieved if the restart strategy is applied in the proposed algorithm. In other words, the contributions made by the restart strategy to the proposed algorithm can be appreciated.

V. Conclusion

In this paper, a biased MOPSO method is suggested to solve the constrained trajectory optimization problems. The proposed method firstly reformulates the original problem to an unconstrained multi-objective optimization model. Subsequently, a locally-enhanced evolutionary process, along with a ε-bias selection method and an evolution restart strategy, is applied to search the optimal solution of the transformed model. To verify the effectiveness of the suggested approach, numerical experiments were carried out on solving a constrained atmospheric entry maneuver planning problem. Comparative studies against other widely-applied trajectory optimization strategies were also performed and presented. From the executed simulations, we have concluded that:

- By analyzing the comparative results, one can observe that it is likely to achieve more optimal solutions and enhanced convergence performance if the evolution restart strategy is applied in the proposed algorithm.
- Benefits can be acquired from applying the bias selection-based non-dominant sorting process.
- Performing the local line search operation is able to locally explore the solution space, thereby making further progresses during the evolutionary process.

Therefore, we believe the constructed approach is of interest to the community focusing on trajectory planner design, and it can be an effective alternative to offer promising results for the considered reentry trajectory optimization problem.

References

[1] B. A. Conway, "A survey of methods available for the numerical optimization of continuous dynamic systems," *Journal of Optimization Theory and Applications*, vol. 152, no. 2, pp. 271–306, 2012.

[2] R. Padhi, S. N. Balakrishnan, and T. Randolph, "Adaptive-critic based optimal neuro control synthesis for distributed parameter systems," *Automatica*, vol. 37, no. 8, pp. 1223–1234, 2001.

[3] H. Yang and H. Baoyin, "Fuel-optimal control for soft landing on an irregular asteroid," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 1688–1697, 2015.

[4] M. Pontani and B. Conway, "Optimal low-thrust orbital maneuvers via indirect swarming method," *Journal of Optimization Theory and Applications*, vol. 162, no. 1, pp. 272–292, 2014.

[5] M. Pontani and B. A. Conway, "Minimum-fuel finite-thrust relative orbit maneuvers via indirect heuristic method," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 5, pp. 913–924, 2014.

[6] D. Gonzalez-Arribas, M. Soler, and M. Sanjurjo-Rivo, "Robust aircraft trajectory planning under wind uncertainty using optimal control," *Journal of Guidance, Control, and Dynamics*, pp. 1–16, 2017.

[7] T. Guo, J. Li, H. Baoyin, and F. Jiang, "Pseudospectral methods for trajectory optimization with interior point constraints: Verification and applications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 3, pp. 2005–2017, 2013.

[8] J. Zhao and S. Li, "Mars atmospheric entry trajectory optimization with maximum parachute deployment altitude using adaptive mesh refinement," *Acta Astronautica*, vol. 160, pp. 401–413, 2019.

[9] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2013.

[10] M. Gabiccini, A. Artoni, G. Pannocchia, and J. Gillis, *A Computational Framework for Environment-Aware Robotic Manipulation Planning*. Springer International Publishing, 2018, pp. 363–385.

[11] T. R. Jorris and R. G. Cobb, "Three-dimensional trajectory optimization satisfying waypoint and no-fly zone constraints," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 2, pp. 551–572, 2009.

[12] D. Garg, M. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington, "A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica*, vol. 46, no. 11, pp. 1843–1851, 2010.

[13] B. Tian, W. Fan, R. Su, and Q. Zong, "Real-time trajectory and attitude coordination control for reusable launch vehicle in reentry phase," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1639–1650, 2015.

[14] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, "Improved gradient-based algorithm for solving aeroassisted vehicle trajectory optimization problems," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 8, pp. 2093–2101, 2017.

[15] G.-P. Liu, J.-B. Yang, and J.-F. Whidborne, *Multiobjective Optimisation and Control*. Engineering Systems Modeling and Control Series, 2003.

[16] A. Rahimi, K. Dev Kumar, and H. Alighanbari, "Particle swarm optimization applied to spacecraft reentry trajectory," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 307–310, 2012.

[17] H. Zhou, X. Wang, and N. Cui, "A novel reentry trajectory generation method using improved particle swarm optimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3212–3223, 2019.

[18] M. Rosa Sentinella and L. Casalino, "Cooperative evolutionary algorithm for space trajectory optimization," *Celestial Mechanics and Dynamical Astronomy*, vol. 105, no. 1, p. 211, 2009.

[19] D. Spiller, C. Circi, and F. Curti, "Particle swarm with domain partition and control assignment for time-optimal maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 4, pp. 968–977, 2017.

[20] M. Pontani and B. A. Conway, "Particle swarm optimization applied to space trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 5, pp. 1429–1441, 2010.

[21] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII*. Springer Berlin Heidelberg, 1998, Conference Proceedings, pp. 611–616.

[22] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, *A Comparison of Particle Swarm Optimization and the Genetic Algorithm*, ser. Structures, Structural Dynamics, and Materials and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2005.

[23] R. Chai, A. Savvaris, and A. Tsourdos, "Violation learning differential evolution-based hp-adaptive pseudospectral method for trajectory optimization of space maneuver vehicle," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 4, pp. 2031–2044, 2017.

[24] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, "Unified multiobjective optimization scheme for aeroassisted vehicle trajectory planning," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 7, pp. 1521–1530, 2018.

[25] H. Duan and S. Li, "Artificial bee colony based direct collocation for reentry trajectory optimization of hypersonic vehicle," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 1, pp. 615–626, 2015.
[26] J. Moore and R. C. Chapman, "Application of particle swarm to multiobjective optimization," in *In: Department of Computer Science and Software Engineering, Auburn University*, 1999.
[27] G. Venter and R. T. Haftka, "Constrained particle swarm optimization using a bi-objective formulation," *Structural and Multidisciplinary Optimization*, vol. 40, no. 1, p. 65, 2010.
[28] S. Lalwani, S. Singhal, R. Kumar, and N. Gupta, "A comprehensive survey: Applications of multi-objective particle swarm optimization (mopso) algorithm," *Transactions on Combinatorics*, vol. 2, no. 1, pp. 39–101, 2013.
[29] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.
[30] H. K. Singh, T. Ray, and W. Smith, "Performance of infeasibility empowered memetic algorithm for cec 2010 constrained optimization problems," in *IEEE Congress on Evolutionary Computation*, 2010, Conference Proceedings, pp. 1–8.
[31] P. A. N. Bosman, "On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 51–69, 2012.
[32] Z. Wang and M. J. Grant, "Constrained trajectory optimization for planetary entry via sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 10, pp. 2603–2615, 2017.
[33] X. Wang, J. Guo, S. Tang, S. Qi, and Z. Wang, "Entry trajectory planning with terminal full states constraints and multiple geographic constraints," *Aerospace Science and Technology*, vol. 84, pp. 620–631, 2019.
[34] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
[35] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.
[36] T. Marcucci, M. Gabiccini, and A. Artoni, "A two-stage trajectory optimization strategy for articulated bodies with unscheduled contact sequences," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 104–111, 2017.
[37] O. Celik, H. Abdulsamad, and J. Peters, "Chance-constrained trajectory optimization for non-linear systems with unknown stochastic dynamics," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Conference Proceedings, pp. 6828–6833.

Solving constrained trajectory planning problems using biased particle swarm optimization

Chai, Runqi

2021-01-11

Chai R, Tsourdos A, Savvaris A, et al., (2021) Solving constrained trajectory planning problems using biased particle swarm optimization. IEEE Transactions on Aerospace and Electronic Systems, Volume 57, Issue 3, June 2021, pp. 1685-1701

https://doi.org/10.1109/TAES.2021.3050645