An Improved Differential Evolution Algorithm and Its Applications to Orbit Design

Wei Yao[*]
Northwestern Polytechnical University, Xi'an 710072, China
University of Strathclyde, Glasgow, Scotland G1 1XJ, United Kingdom

Jianjun Luo[†]
Northwestern Polytechnical University, Xi'an 710072, China

Malcolm Macdonald[‡]
University of Strathclyde, Glasgow, Scotland G1 1XJ, United Kingdom

Mingming Wang[§]
Northwestern Polytechnical University, Xi'an 710072,China

Weihua Ma[¶]
Northwestern Polytechnical University, Xi'an 710072,China

Nomenclature

a	=	semi-major axis of orbit, km
CR	=	crossover ratio factor for differential evolution
D	=	days to repeat during multi-target visiting problem
$d_{(t,i)}$	=	Earth central angle between target P_i and sub-satellite points (SSP) at time t, rad
e	=	orbital eccentricity
F	=	evolution scaling factor for differential evolution
f	=	true anomaly of orbit, rad
h	=	angular momentum of orbit, km^2/s
i	=	inclination of orbit, rad
J	=	objective function
J_2	=	Earth's oblateness perturbation
J_t	=	objective function to describe visiting time of targets
J_b	=	objective function to describe bonus term of visited targets
K	=	number of generations
k_{rand}	=	a random integer from 0 to n

[*]Ph.D. Candidate, National Key Laboratory of Aerospace Flight Dynamics, yaowei@mail.nwpu.edu.cn.
[†]Professor, National Key Laboratory of Aerospace Flight Dynamics, jjluo@nwpu.edu.cn.
[‡]Reader, Strathclyde Space Institute, malcolm.macdonald.102@strath.ac.uk.
[§]Dr.-Ing, National Key Laboratory of Aerospace Flight Dynamics, mwang@nwpu.edu.cn.
[¶]Associate Professor, National Key Laboratory of Aerospace Flight Dynamics, whma_npu@nwpu.edu.cn.

N	=	number of optimization variables
n	=	initial population size
P_i	=	i_{th} target of multi-target visiting problem
p_i	=	priority of targets
p^*	=	high bonus term for the visited target
Q	=	fitness function
R	=	revolutions to repeat during multi-target visiting problem
R_E	=	radius of Earth, km
rand_j	=	a random number from 0 to 1 for each individual j
t_0	=	start time of Lambert transfer, s
t_f	=	stop time of Lambert transfer, s
$U_{i,j}(g)$	=	alternative vector components of the generation g
$V_{i,j}(g)$	=	trial vector components of the generation g
$X_{i,j}(g)$	=	individual of the generation g
γ	=	supplementary angle of $\lambda + \eta$, rad
Δv_1	=	magnitude of the velocity impulse at departure point of Lambert transfer, m/s
Δv_2	=	magnitude of the velocity impulse at arrival point of Lambert transfer, m/s
η	=	half-effective angle of FOV given by mission requirements, rad
θ	=	rotating angle from initial coordinate system to rotating Earth-fixed frame, rad
θ_0	=	initial Greenwich hour angle at the start time t_0, rad
λ	=	effective Earth central angle of the field of view (FOV), rad
μ	=	gravitational constant, km^3/s^2
ρ	=	distance between satellite and bounds of sensor on Earth's surface, km
ϕ	=	longitude of sub-satellite points (SSP), rad
φ	=	latitude of sub-satellite points (SSP), rad
Ω	=	Right Ascension of Ascending Node (RAAN), rad
ω	=	argument of perigee, rad
ω_E	=	rotational angular velocity of Earth, rad/s

Subscripts

i	=	sequence number of variables
j	=	sequence number of individuals
g	=	sequence number of generations

0	=	departure point of Lambert transfer
f	=	arrival point of Lambert transfer

Superscripts

U	=	upper limit of the parameters
L	=	lower limit of the parameters
best	=	choose the best vector as based vector
random	=	choose a random vector as based vector
+	=	the instant immediately after the application of an impulse
−	=	the instant immediately before the application of an impulse

I. Introduction

Differential Evolution (DE) is a basic and robust evolutionary strategy that has been applied to determining the global optimum for complex optimization problems[1–5]. It was introduced in 1995 by Storn and Price [1] and has been successfully applied to optimization problems including nonlinear, non-differentiable, non-convex, and multi-model functions. DE algorithms show good convergence, high-reliability, simplicity, and a reduced number of controllable parameters [2]. Olds and Kluever [3] applied DE to an interplanetary trajectory optimization problem and demonstrated the effectiveness of DE to produce rapid solutions. Madavan [4] discussed various modifications to the DE algorithm, improved its computational efficiency, and applied it to aerodynamic shape optimization problems. DE algorithms are easy to use, as they require only a few robust control variables, which can be drawn from a well-defined numerical interval. However, the existing various DE algorithms also have limitations, being susceptible to instability and getting trapped into local optima[2]. Notable effort has been spent addressing this by coupling DE algorithms with other optimization algorithms (for example, Self Organizing Maps (SOM) [6], Dynamic Hill Climbing (DHC) [7], Neural Networks (NN) [7], Particle Swarm Optimization (PSO) [8]) . In these cases, the additional algorithm is used as an additional loop within the optimization process, creating a hybrid system with an inner and outer loop. Such hybrid algorithms are inherently more complex and so the computation cost is increased. Attempting to address this, a self-adaptive DE was designed and applied to the orbit design problem for prioritized multiple targets by Chen[5]. However, the self-adaptive feature is somewhat limited as it relates only to the number of generations within the optimization. A Self-adaptive DE which can automatically adapt its learning strategies and the associated parameters during the evolving procedure was proposed by Qin and Suganthan[9] and 25 test functions were used to verify the algorithm.

This Note presents an improved DE algorithm that have an opportunity to escape the local optima whilst also improving the performance of the algorithm without relying on an inner and outer optimization loop. Two improvements have been added to the algorithm (in Section II) : a double self-adaptive scaling factors related to the value of the fitness function of the last generation and the number of generations, and a mutant factor in each generation that will change randomly to avoid locally optimum solutions. To validate the new method, test functions studies (in Section III) and two case studies (in Section IV) on transfer orbit design and target observation are investigated to demonstrate the accuracy, convergence and the results' standard deviation (SD) of the proposed algorithm.

II. Optimization Algorithm

A. Traditional DE Algorithm

The implementation of DE can be described following [5] and summarized herein. The initial population of the individuals is generated as

$$X_{i,j}(0) = X_i^{\mathrm{L}} + \mathrm{rand}_j(0,1)(X_i^{\mathrm{U}} - X_i^{\mathrm{L}}) \qquad i = 1,\ldots,N; j = 1,\ldots,n. \tag{1}$$

Based on the individuals of the previous generation, a trial vector is generated by a mutation process. This operation contains two popular selections of the base vector, best and random. The best current vector is chosen as the base vector by

$$V_{i,j}^{\mathrm{best}}(g) = X_i^{\mathrm{best}}(g) + F(X_{i,j1}(g) - X_{i,j2}(g)) \qquad j1 \neq j2; j1 = 1,\ldots,n; j2 = 1,\ldots,n; g = 1,\ldots,K; \tag{2}$$

whilst the base vector is chosen randomly by

$$V_{i,j}^{\mathrm{rand}}(g) = X_i^{\mathrm{rand}}(g) + F(X_{i,j1}(g) - X_{i,j2}(g)) \qquad j1 \neq j2; j1 = 1,\ldots,n; j2 = 1,\ldots,n; g = 1,\ldots,K; \tag{3}$$

where, the scaling factor is a random, uniformly distributed value between [0,1]

$$F_{i,j} = \mathrm{rand}(0,1). \tag{4}$$

The alternative vector is produced by crossover process using Eq. (5) and the different algorithms use different base vector.

$$U_{i,j}(g) = \begin{cases} V_{i,j}^{\mathrm{best/rand}}(g) \qquad if(\mathrm{rand}_j \leq CR_i(g) \qquad or \qquad j = k_{\mathrm{rand}}) \\ X_{i,j}(g) \qquad \text{otherwise} \end{cases} \tag{5}$$

where, the predefined crossover probability CR controls the fraction of trial vectors that are used and k_{rand} ensure that at least one parameter is changed.

$$CR_i(g) \in (0,1). \tag{6}$$

Finally, it is determined whether to select the alternative vector as a member of the new generation, or not, by the value of fitness function Q. The alternative vector is selected as a member of the new generation if the fitness function is smaller than the target vector.

$$X_{i,j}(g+1) = \begin{cases} U_{i,j}(g) & if[Q_{U_{i,j}(g)} < Q_{X_{i,j}(g)}] \\ X_{i,j}(g) & \text{otherwise} \end{cases} \tag{7}$$

This process is iterated from step (2) to step (4) until the stopping criteria are satisfied; usually the maximum number of generations or the minimum value of fitness function.

B. Double Self-adaptive Scaling Factor

In the previously described, or traditional DE algorithm, the evolution scaling factor F has a significant influence on the extent of population variations as it describes the search range of the optimization algorithm. The algorithm will reduce to a random searching algorithm when $F = 1$, whereas the algorithm will lose its evolution capability when $F = 0$ and the self-adaptive factor which only related to the number of generations is studied in [5].

A double self-adaptive evolution scaling factor is introduced to enhance the optimal performance by gradually decreasing the scale factor F with each new generation. Consequently the algorithm will then have a different search range in different stages of the optimization. The searching process will proceed over a wide area at the beginning, similar to finding some tolerable results by random searching, and gradually decrease the search area as the near-optimal result is found in a local region of the current best individual, similar to local search by Newton downhill method. However, the searching process continues to require some element of randomness to ensure the exploration of the searching space.

The scaling factor is established by three parts. The first part $F_{i,j}(Q_{(g-1)})$ is related to the value of the fitness function of the last generation. The second part $F_{i,j}(g)$ is related to the number of generations, and the third part is a random factor similar to the traditional DE algorithm. The details of the functions can be designed as required and the equations used in this Note are only one example selected to illustrate the method. Consequently, the specific expression is

$$F_{i,j} = F_{i,j}(Q_{(g-1)}) \times F_{i,j}(g) \times \text{rand}(0,1). \tag{8}$$

The part of the scaling factor related to the fitness function of the last generation can be designed to ensure the factor $F_{i,j}(Q_{(g-1)})$ tends from 1 to 0 as the fitness function of the prior generation decreases. This is given as

$$F_{i,j}(Q_{(g-1)}) = \frac{1}{\log_{10}\left(\frac{\mathrm{C}\times \mathrm{Q}^{\mathrm{best}}_{(1)}}{\left|\mathrm{Q}^{\mathrm{best}}_{(\mathrm{g}-1)} - \mathrm{Q}^{\mathrm{best2}}_{(\mathrm{g}-1)}\right|}\right)} \qquad \rightarrow \qquad F_{i,j}(Q^{\mathrm{best}}_{(g-1)}) \in (0,1) \tag{9}$$

where, $Q^{\text{best}}_{(g-1)}$, $Q^{\text{best2}}_{(g-1)}$ are the two best obtained values of the fitness function in last generation, and $Q^{\text{best}}_{(1)}$ is the best value of fitness function among the initial population. The order of magnitude of $(Q^{\text{best}}_{(g-1)} - Q^{\text{best2}}_{(g-1)})$ will, most likely, change significantly during the optimization process, as such the common logarithm of $(\frac{C \times Q^{\text{best}}_{(1)}}{Q^{\text{best}}_{(g-1)} - Q^{\text{best2}}_{(g-1)}})$ is applied to ensure the factors $F_{i,j}(Q_{(g-1)})$ remains smooth. The constant C here is equal to 10 to make sure the factors $F_{i,j}(Q_{(g-1)})$ will not exceed 1.

The part of the scaling factor related to the generation sequence number is similarly designed to ensure the factors $F_{i,j}(g)$ changes from 1 to 0 as the generation sequence number increases,

$$F_{i,j}(g) = (\cos(\frac{g}{K} \times \pi) + 1) \times \frac{1}{2} \qquad \longrightarrow \qquad F_{i,j}(g) \in (0, 1). \tag{10}$$

C. Random Mutation

Applying the double self-adaptive scaling factor alone would most likely result in the algorithm becoming trapped in a local optimum, with no means of escape. To resolve this , a random mutation factor is added into the process. This mutant is produced randomly, like the initial population, rather than evolving with the whole group. The setting of the random mutation can be regarded as a re-initialization and is defined as

$$U_{i,n}(g) = X_i^{\text{L}} + \text{rand}_j(0, 1)(X_i^{\text{U}} - X_i^{\text{L}}) \qquad g = 1, \ldots, K. \tag{11}$$

Consequently, if the mutant individual has a better performance the optimization algorithm will exit the local optimal, and restart the evolution process around this individual.

As shown in Table 1, six different algorithms can be derived with different properties based on the choice of the best or random (rand) base vector, the introduction of the random mutant (RM), and the self-adaptive (SA) factors. DE (best/rand) algorithms are the traditional differential algorithm, with two kinds of base vector. SA-DE (best/rand) represent the algorithms that have double self-adaptive scaling factors, and SA-DE-RM (best/rand) represent the algorithms with both double self-adaptive scaling factors and the random mutant. Each of these algorithms will be considered in comparison in later sections.

Table 1 The property list for six algorithms

No.	Algorithms	Random Mutant	Self Adaptive Factors	Base Vector
1	DE (best)	*N*	*N*	the best vector of last generation
2	SA-DE (best)	*N*	*Y*	the best vector of last generation
3	SA-DE-RM (best)	*Y*	*Y*	the best vector of last generation
4	DE (rand)	*N*	*N*	the random vector of last generation
5	SA-DE (rand)	*N*	*Y*	the random vector of last generation
6	SA-DE-RM (rand)	*Y*	*Y*	the random vector of last generation

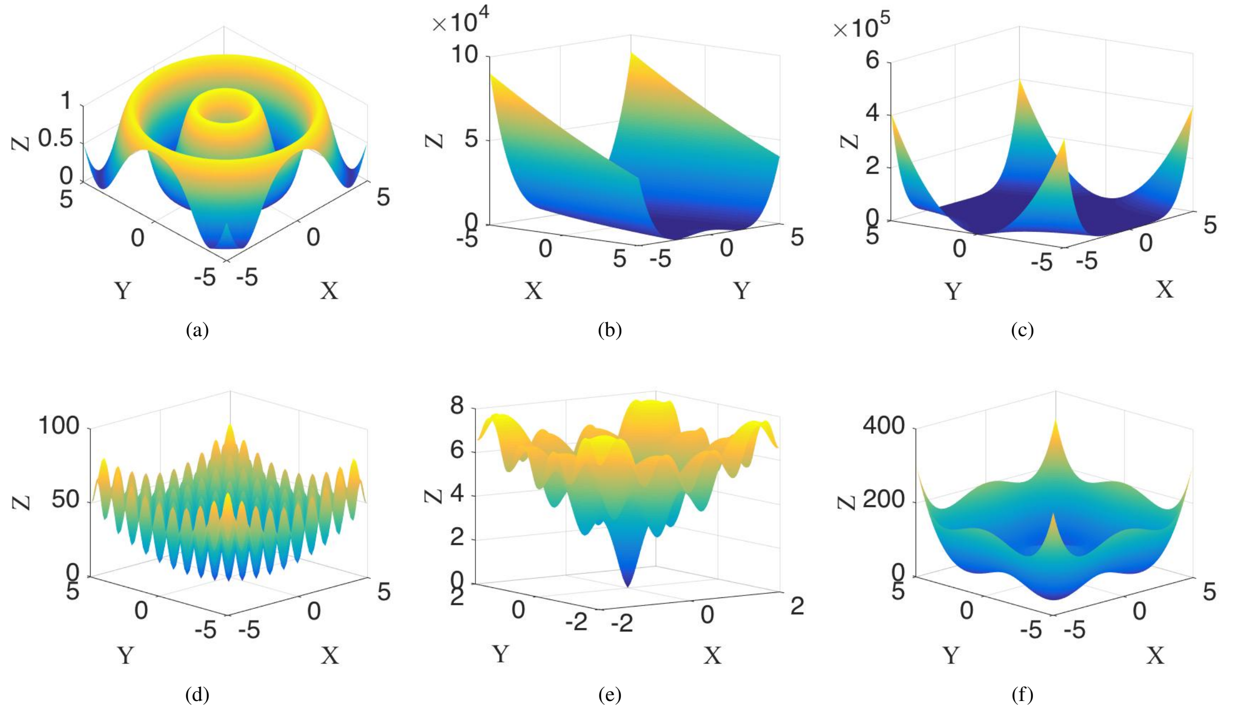


Fig. 1 3D-plot of test functions: a) Schaffer, b) Rosenbrock, c) Beale's, d) Rastrigin, e) Ackley's, f) Styblinski-Tang

III. Test Function Simulation

To investigate the feasibility and effectiveness of the introduced algorithm, eight classic functions are used to test the different algorithms. Figure 1 shows the 3D-plot of the six two dimension (2D) test functions, it can be seen that they all have a certain complexity, and applying the algorithms in Table 1 provides a comprehensive analysis of the algorithms.

A. Test Function Introduction

Table 2 shows the search space and global minimum of the test functions. These functions are some typical benchmarks chosen from [1, 2, 6, 10] and they have different properties include multi-modal or unimodal, non-separable or separable, notated, shifted, scalable. A simple introduction about the test functions is given as following and more details can be found in [1, 2, 6, 10]. The global minimum of Schaffer function [10] is surrounded by two circular valleys and the optimization process risks becoming trapped in these local minimums. Rosenbrock valley [10] is a non-convex function and the global optimum lays inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult. Beale's function [6] is multi-modal, with sharp peaks at the corners of the input domain. Rastrigin function [10] is based on the function of De Jong with the addition of cosine modulation in order to produce frequent local minimum. Thus, the function is a typical example of nonlinear highly multi-modal function however, the locations of the minimum are regularly distributed. The Ackley's function [10] is characterized by a nearly flat outer region, and a large hole at the center. The function poses a risk for optimization algorithms to become trapped in one of its local minimum. In the Styblinski-Tang function, a large difference between the magnitudes of middle or corner brings the similar difficulty to Beale's function. The Hyper-Ellipsoid and Griewank [10] are both 10 dimension (10D) test functions which can verify the performance of algorithms for high dimension problem.

Table 2 Test function characteristic list

Function	Formula	Search Space
Schaffer	$f_1(x, y) = 0.5 + \frac{\sin^2\sqrt{(x^2+y^2)}-0.5}{[1+0.001(x^2+y^2)]^2}$	$(-100 < x_1, y_1 < 100)$
Rosenbrock	$f_2(x, y) = 100(y - x^2)^2 + (1 - x)^2$	$(-2.048 < x_2, y_2 < 2.048)$
Beale's	$f_3(x, y) = \sum_{i=1}^{3}[\frac{3\times(2^i-1)}{2^i} - x + xy^i]^2$	$(-4.5 < x_3, y_3 < 4.5)$
Rastrigin	$f_4(x, y) = 20 + x^2 + y^2 - 10\cos(2\pi x) - 10\cos(2\pi y)$	$(-5.12 < x_4, y_4 < 5.12)$
Ackley's	$f_5(x, y) = 20 \times (1 - e^{-\frac{1}{5}\sqrt{\frac{1}{2}(x^2+y^2)}}) + (1{-}^{\frac{1}{2}[\cos(2\pi x)+\cos(2\pi y)]}) \times e$	$(-5 < x_5, y_5 < 5)$
Styblinski-Tang	$f_6(x, y) = \frac{x^4-16x^2+5x+y^4-16y^2+5y}{2} + 78.33234$	$(-5 < x_6, y_6 < 5)$
Hyper-Ellipsoid	$f_7(x_i) = \sum_{i=1}^{10}(i^2 {x_i}^2)$	$(-1 < x_i < 1)$
Griewank	$f_8(x_i) = \sum_{i=1}^{10}(\frac{{x_i}^2}{4000}) - \prod_{i=1}^{10}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$(-100 < x_i < 100)$

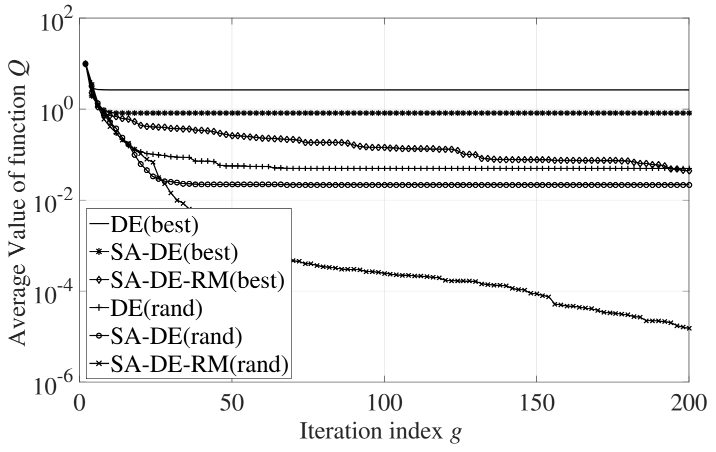
B. Numerical Simulations

The algorithms are applied to the eight classical test functions to determine their performance. The simulation results are shown in Table 3. The maximum number of generations is set as 200, with each generation having 20 individuals, and each optimization is run 50 times. From Table 3, all of the best result for the test functions are within 10^{-5} of the global minimums. The SA-DE-RM (rand) algorithm gives the best solution both on average and with minimum standard deviation (SD) .

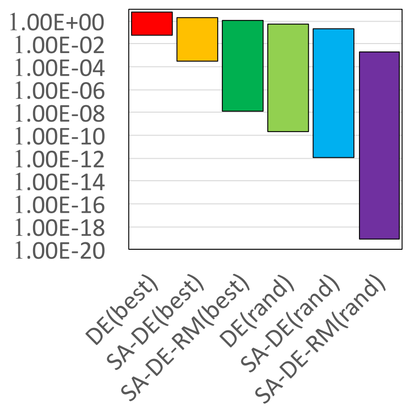
Figure 2(b) gives the distribution range of the 50 solutions for Beale's function. It shows the upper and lower bound of the results for 50 simulations to find the different spread range. Fig. 2(a)gives the corresponding average value. The result shows when introduced the self-adaptive scaling factor give a better performance than DE algorithms, and likewise when the random mutant is added a further improvement in performance is gained. It should be noticed from Fig. 2(a) that the two SA-DE algorithms do get trapped in local optima, and make little or no progress during the last 100 generations. However, the algorithms with the random mutant can escape the local minimum to find a better solution. It was found that the test results for the other functions gave a similar conclusion, these results are summarized in Table 3.

Table 3 Simulation results for six algorithms and 50 Monte Carlo samples

Function	Metrics	DE (best)	SA-DE (best)	SA-DE-RM (best)	DE (rand)	SA-DE (rand)	SA-DE-RM (rand)
Schaffer Founction (2D)	optimum	$f_1(0,0) = 0$					
	best	$f_1(-4.303567 \times 10^{-9}, 6.247260 \times 10^{-10}) = 2.8732 \times 10^{-8}$					
	average	2.292×10^{-2}	1.624×10^{-2}	9.847×10^{-3}	9.444×10^{-3}	8.940×10^{-3}	8.520×10^{-3}
	SD	4.029×10^{-2}	2.603×10^{-2}	1.169×10^{-2}	1.056×10^{-2}	9.319×10^{-3}	9.052×10^{-3}
Rosenbrock Founction (2D)	optimum	$f_2(1,1) = 0$					
	best	$f_2(1.000365, 1.000754) = 1.8966 \times 10^{-7}$					
	average	2.994×10^{-1}	2.430×10^{-1}	1.009×10^{-1}	1.695×10^{-1}	6.585×10^{-2}	1.801×10^{-2}
	SD	5.348×10^{-1}	5.030×10^{-1}	1.879×10^{-1}	6.111×10^{-1}	1.325×10^{-1}	3.173×10^{-2}
Beale's Founction (2D)	optimum	$f_3(3, 0.5) = 0$					
	best	$f_3(3.000000, 0.500000) = 8.0019 \times 10^{-20}$					
	average	2.6394	8.178×10^{-1}	4.439×10^{-2}	4.939×10^{-2}	2.176×10^{-2}	1.499×10^{-5}
	SD	3.4009	1.6236	1.539×10^{-1}	1.126×10^{-1}	5.331×10^{-2}	3.593×10^{-5}
Rastrigin Founction (2D)	optimum	$f_4(0,0) = 0$					
	best	$f_4(1.713657 \times 10^{-9}, -3.947406 \times 10^{-10}) = 3.552714 \times 10^{-15}$					
	average	2.9907	1.0548	3.269×10^{-1}	4.452×10^{-1}	1.419×10^{-1}	5.971×10^{-2}
	SD	4.0222	1.3774	6.360×10^{-1}	8.457×10^{-1}	3.724×10^{-1}	2.437×10^{-1}
Ackley's Founction (2D)	optimum	$f_5(0,0) = 0$					
	best	$f_5(-3.585047 \times 10^{-17}, -7.617551 \times 10^{-17}) = 8.881784 \times 10^{-16}$					
	average	7.053×10^{-1}	1.191×10^{-1}	4.184×10^{-2}	6.596×10^{-3}	8.116×10^{-5}	2.177×10^{-5}
	SD	1.1822	2.296×10^{-1}	2.327×10^{-1}	3.003×10^{-2}	2.610×10^{-4}	4.357×10^{-5}
Styblinski -Tang Founction (2D)	optimum	$0 < f_6(-2.903534, 2.903534) < 2 \times 10^{-5}$					
	best	$f_6(-2.903534, 2.903534) = 8.592457 \times 10^{-6}$					
	average	3.765×10^{-1}	2.500×10^{-1}	3.430×10^{-2}	1.392×10^{-4}	2.204×10^{-5}	8.655×10^{-6}
	SD	4.100×10^{-1}	3.115×10^{-1}	8.474×10^{-2}	7.738×10^{-4}	9.419×10^{-5}	8.657×10^{-6}
Hyper -Ellipsoid Founction (10D)	optimum	$f_7(0,0,0,0,0,0,0,0,0,0) = 0$					
	best	$f_7 = 1.167836 \times 10^{-13}$					
	average	2.696×10^{-1}	2.134×10^{-1}	2.864×10^{-2}	3.088×10^{-3}	1.033×10^{-4}	4.875×10^{-7}
	SD	2.951×10^{-1}	2.403×10^{-1}	5.653×10^{-2}	4.621×10^{-3}	2.815×10^{-4}	3.447×10^{-6}
Griewank Founction (10D)	optimum	$f_8(0,0,0,0,0,0,0,0,0,0) = 0$					
	best	$f_8 = 1.332268 \times 10^{-15}$					
	average	1.037×10^{-3}	8.585×10^{-4}	8.672×10^{-5}	1.104×10^{-5}	4.811×10^{-7}	1.492×10^{-10}
	SD	1.113×10^{-3}	9.456×10^{-4}	1.851×10^{-4}	2.460×10^{-5}	1.224×10^{-6}	1.055×10^{-9}



(a) Test average for Beale's function simulation



(b) Distribution range for Beale's function simulation

Fig. 2 The simulation result of Beale's function for 50 simulations

It need to be noticed that the computation time cost of the six algorithm are almost same because it is only related with the number of populations and the max number of generations.

IV. Case Study Simulation

A. Lambert Transfer Problem Simulation

1. Problem Definition

The problem seeks to minimize the two-impulse optimal transfer between two circulars, non-coplanar orbits. Therefore, the fitness function is

$$\min(Q) = \Delta v_1 + \Delta v_2 \tag{12}$$

where the two velocity impulses Δv_1 and Δv_2 can be given as

$$\Delta v_1 = |\boldsymbol{v(t_0^+)} - \boldsymbol{v(t_0^-)}|, \quad \Delta v_2 = |\boldsymbol{v(t_f^+)} - \boldsymbol{v(t_f^-)}|. \tag{13}$$

The angular momentum can be written as

$$h = \sqrt{\mu a(1-e^2)} \tag{14}$$

The three transform matrices about Ω, i and ω are given as

$$R_{(\Omega)} = \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_{(i)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix}, R_{(\omega)} = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{15}$$

Thus, the state vectors at the departure point can be derived as

$$\boldsymbol{r_1} = R'_{(\Omega_0)} R'_{(i_0)} R'_{(\omega_0)} \frac{h_0^2/\mu}{1 + ecos(f(t_0^-))} \begin{bmatrix} \cos(f(t_0^-)) \\ \sin(f(t_0^-)) \\ 0 \end{bmatrix}, \tag{16}$$

$$\boldsymbol{v(t_0^-)} = R'_{(\Omega_0)} R'_{(i_0)} R'_{(\omega_0)} \frac{\mu}{h_1} \begin{bmatrix} -\sin(f(t_0^-)) \\ \cos(f(t_0^-)) + e \\ 0 \end{bmatrix}, \tag{17}$$

where Ω_0, i_0 and ω_0 are the orbit elements of initial orbit; $R'_{(\Omega_0)}$,$R'_{(i_0)}$,$R'_{(\omega_0)}$ are the transposed matrix of each transform matrix and $f(t_0^-)$ is the true anomaly of the departure point on the initial orbit. Similarly,$\boldsymbol{r_2}$,$\boldsymbol{v(t_f^+)}$ can be calculated using the orbit elements of final orbit, where $f(t_f^+)$ is the true anomaly of the arrival point on the final orbit. $f(t_0^-)$ and $f(t_f^+)$ are the first two design variables.

After determining $\boldsymbol{v(t_0^-)}$ and $\boldsymbol{v(t_f^+)}$, as a classic Lambert problem, $\boldsymbol{v(t_0^+)}$,$\boldsymbol{v(t_f^-)}$ can be calculated from $\boldsymbol{r_1}$,$\boldsymbol{r_2}$,Δt

by algorithm 5.2 (the details of the algorithm is discussed in [11]) and then the total required velocity change of the maneuver can be given from Eq. (12) and (13) (where Δt is the third design variable) . As the orbit elements of the initial and final orbit are known, except the true anomaly, there will be three unknown parameters of the Lambert transfer problem to be optimized: the true anomaly on the initial orbit at the departure point (where the first impulse occurs) $f(t_0^-)$,the true anomaly on the final orbit at the arrival point (where the second impulse occurs) $f(t_f^-)$, and transfer time Δt.

2. Numerical Simulation

For the Lambert transfer problem each different DE algorithm uses 10 particles (n = 10) and 2000 generations (K = 2000), with each solution generated 50 times via a Monte Carlo approach where the initial population is randomly distributed each time. Each individual includes three design variables: $[f(t_0^-),\ f(t_f^+),\ \Delta t]$. The design variables are constrained to the following ranges, which define the search space:

$$f(t_0^-) \in [0, 2\pi] \qquad f(t_f^+) \in [0, 2\pi] \qquad f\Delta t \in [0, 20\text{TU}] \tag{18}$$

where the normalized set of units are given as: the Earth radius represents the distance unit (DU), whereas the time unit (TU) is TU=$\sqrt{\text{DU}^3/\mu}$. The search space in Eq. (18) means the true anomaly of the departure point and arrival point can be search in a complete orbit period and the transfer time is less than twenty time unit.

Table 4 Orbital elements of the initial and target orbits and the optimal transfer

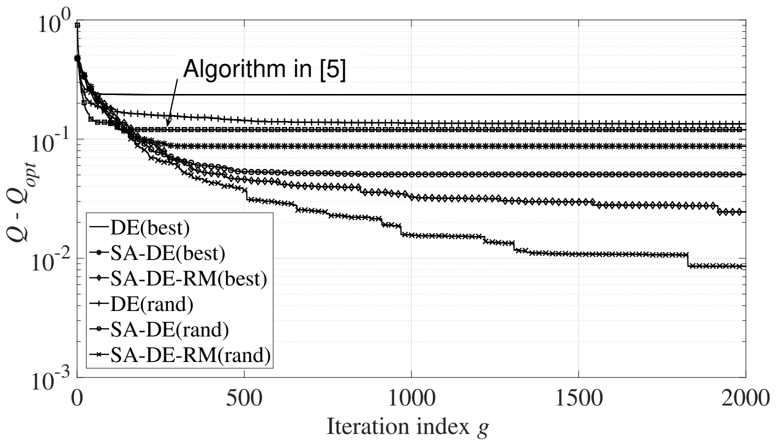
Orbit	a, km	e	i, deg	Ω, deg	ω, deg
Initial	9645.83	0.2	5	0	270
Target	11575	0.2	0	0	30
Optimal transfer	12441.7	0.104	5.25	7.56	21.24

Table 5 True anomalies and transfer time of the global optimal solution

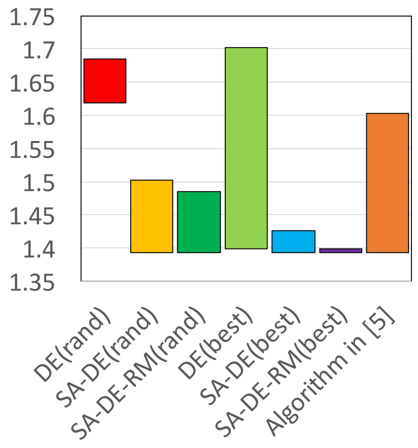
Optimal parameters	Value
f_{1-}	163.8 deg
f_{2+}	157.5 deg
f_{1+}	45.0 deg
f_{2-}	158.7 deg
t	4490.5 s

The initial orbit and final orbit are given through their orbit elements in Table 4. Tables 4 and 5 present the results of the optimization process by SA-DE-RM algorithm, including the true anomaly of the departure point on the initial orbit and transfer orbit, the true anomaly of the arrival point on the final orbit and transfer orbit, and the orbit elements of transfer orbit. The required velocity change, as shown Table 6, is 1.392970 km/s .

As shown in Table 6, Fig. 3(a) and 3(b), the six different algorithms behave differently in various evaluation parameters and the random base vector method typically performs better than the best base vector. Focusing on the two



(a) The average value of the fitness function for 50 simulations



(b) The distribution range

Fig. 3 The simulation result of Lambert problem for 50 simulations

base vector types, the performance of the evolution algorithm is significantly improved with each step: introducing the self-adaptive factors and the random mutant. It should be noticed that the four algorithms with self-adaptive factors show a better convergence and furthermore the two algorithms with random mutants are better at escaping local optimum when trapped and, as a result, are able to find a better solution. From Fig. 3, the algorithm in [5] performs only better than the two traditional DE algorithm.

From Table 6, the SA-DE-RM (rand) algorithm has the most stable performance, with the lowest standard deviation (SD) of 0.0295. Fig. 3(a) illustrates the objective evolution as a function of the iteration index and y-axis represents the error with the optimal value in logarithmic scale. Fig. 3(b) gives the distribution range of the result of 50 Monte Carlo simulations, with the SA-DE-RM (rand) algorithm again seen to have the smallest range, and to be nearest to the best found solution. In this case, the best found transfer trajectory gives a minimum value of Q equal to 1.392970km/s. Whilst the result given by the algorithm in [5] is 1.393264km/s, and hence very similar it is of note that the SD is significantly greater.

Table 6 Simulation results, average and SD for six algorithms by 50 Monte Carlo samples

No.	Algorithms	Best Result, km/s	Average Value, km/s	SD
1	DE (best)	1.619151	1.628888	0.2410682
2	SA-DE (best)	1.393047	1.480205	0.1477757
3	SA-DE-RM (best)	1.392989	1.417463	0.0662108
4	DE (rand)	1.398566	1.527296	0.1687570
5	SA-DE (rand)	1.392994	1.443502	0.1220198
6	SA-DE-RM (rand)	1.392970	1.401533	0.0295157
7	Algorithm in [5]	1.393264	1.513093	0.1672050

B. Multi-target Visiting Problem Simulation

1. Problem Definition

The objective in this case study is to find a trajectory that will naturally overfly multiple fixed targets as often as possible, and as close to nadir as possible, during a given period. The effective Earth central angle λ of the field of view (FOV) for an observation satellite can be computed, following[12], as

$$\sin\lambda = \frac{\rho \sin\eta}{R_E}, \quad \rho = R_E \cos\gamma + a\cos\eta, \quad \sin\gamma = \frac{a\sin\eta}{R_E} \tag{19}$$

.

Then, using Eq. (16) to compute the position of the satellite $\boldsymbol{r} = [X, Y, Z]$ in an inertial coordinate system, the satellite's position about rotating Earth-fixed frame $\boldsymbol{r'} = [X', Y', Z']$ is given as follows:

$$\boldsymbol{r'} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{r} \tag{20}$$

where, at time $t = t_0 + \Delta t$, the rotating angle is $\theta = \theta_0 + \omega_E \Delta t$.

Thus, the latitude φ and longitude ϕ of the sub-satellite points (SSP) can be given as

$$\varphi = \sin^{-1}(Z'/a) \tag{21}$$

$$\phi = \begin{cases} \cos^{-1}(\frac{X'}{a\cos\varphi}) & if(Y' > 0) \\ 360 - \cos^{-1}(\frac{X'}{a\cos\varphi}) & if(Y' \leq 0) \end{cases} . \tag{22}$$

Considering an oblate Earth to the order of J_2, $\dot{\Omega}$ and $\dot{\omega}$ can be given, following[11], as,

$$\dot{\Omega} = -[\frac{3}{2}\frac{\sqrt{\mu}J_2R_E^2}{(1-e^2)^2 a^{\frac{7}{2}}}]\cos i, \quad \dot{\omega} = -[\frac{3}{2}\frac{\sqrt{\mu}J_2R_E^2}{(1-e^2)^2 a^{\frac{7}{2}}}](\frac{5}{2}\sin^2 i - 2) \tag{23}$$

Thus during the computation of SSP, Ω and ω at time $t = t_0 + \Delta t$ need to be updated as

$$\Omega = \Omega_0 + \dot{\Omega}\Delta t, \quad \omega = \omega_0 + \dot{\omega}\Delta t \tag{24}$$

The corresponding semi-major axis can be compute by iterative algorithm, following[11], as

$$a_0 = \frac{\mu^{\frac{1}{3}}}{(n\omega_E)^{\frac{2}{3}}}, n = \frac{R}{D} \tag{25}$$

$$a_{k+1} = \frac{\mu^{\frac{1}{3}}}{(n\omega_E)^{\frac{2}{3}}}[1 - \frac{3}{2}(-J_2)(\frac{R_E}{a_k})^2(1 - \frac{3}{2}\sin^2 i)]^{\frac{3}{2}} 1 + (-J_2)(\frac{R_E}{a_k})^2[\frac{3}{2}(n\cos i - \frac{3}{4}(5\cos^2 i - 1)]^{\frac{3}{2}}, \tag{26}$$

where R and D are both integer variables and the stopping criterion is

$$|a_{k+1} - a_k| < \text{tolerance} \tag{27}$$

with tolerance set to 10^{-10} km.

Assuming the projection of FOV is a circle, the target position, P_i, is $[\phi_i, \varphi_i]$ and the position of SSP at time t is $[\phi_t, \varphi_t]$, the Earth central angle between the target and SSP is given as follows by use of spherical triangle

$$d_{(t,i)} = \cos^{-1}(\cos(\frac{\pi}{2} - \varphi_t)\cos(\frac{\pi}{2} - \varphi_i) + \sin(\frac{\pi}{2} - \varphi_t)\sin(\frac{\pi}{2} - \varphi_i)\cos(\phi_t - \phi_i)). \tag{28}$$

The target P_i will be visited at time t when the following condition is true,

$$d_{(t,i)} < \lambda \tag{29}$$

where λ is given by Eq. (19) .

Considering the priority of targets, p_i, a priority-weighted objective function is given as follows:

$$\max(J) = J_t + J_b \tag{30}$$

$$J_t = \sum_{i=1}^{N_1} \frac{p_i(\sum_{j=1}^{N_2} t_{i,j})}{10}, \quad J_b = \sum_{i=1}^{N_1} p^* \cdot o_i \tag{31}$$

where, $o_i = 1$ means the target P_i is visited at least once and $o_i = 0$ represents the target is missed; $p^* = 100$ means a high bonus term for the target not be lost. $t_{i,j}$ represents the time period of the jth times visiting for the target P_i and N_1, N_2 represent the number of targets and the times of visit for P_i. Here a conversion $Q = 1/J$ is necessary and Q will be used during the optimal process as the fitness function to make sure the smaller fitness function is better.

2. Numerical Simulation

The maximum generation is set to 500, with 20 individuals in each generation. An orbit is sought to flyover the 10 targets shown in Table 7 within a repeat period of 2 days. Considering the imaging resolution constraint and the effect of atmospheric drag, the bounds of altitude are set as [300 km, 900 km], and the corresponding bounds of R is [29,31]. As shown in Table 7, the highest latitude of the targets is 55.5 deg, thus the bounds of inclination i should be set to [50,90]. The sensor's half-effective FOV is given by mission requirements but assumed here to be $\eta = 20$ deg, and this study assumes the start time is 00:00:00 at 01 Jan 2017, giving an initial Greenwich hour angle $\theta_0 = 100.84$ deg. Table 8 and

Table 7 Distribution and Priority of Observation Targets

ID	City	Longitude, deg	Latitude, deg	Priority
P_1	Moscow	37.4	55.5	0.72
P_2	London	0.1	51.3	0.85
P_3	Peking	116.2	39.6	1.00
P_4	Washington	-77.0	38.5	1.00
P_5	Los Angeles	-118.2	34	0.85
P_6	Miami	-80.1	25.5	0.73
P_7	HongKong	115.1	21.2	0.68
P_8	Rio	-43.2	-22.5	0.62
P_9	Sydney	151.1	-33.5	0.90
P_{10}	Buenos Aires	-58.3	-34.4	0.65

Table 8 Simulation results of multi-visiting problem

No.	Algorithms	i, deg	Ω, deg	f, deg	R	J_t	J_b
1	DE (best)	63.21	16.82	23.64	29	73.35	1000
2	SA-DE (best)	64.35	106.75	178.46	29	76.81	1000
3	SA-DE-RM (best)	55.18	171.59	172.80	29	79.16	1000
4	DE (rand)	55.64	96.07	62.53	29	76.73	1000
5	SA-DE (rand)	55.52	30.20	60.69	29	79.43	1000
6	SA-DE-RM (rand)	55.51	125.69	295.75	29	80.11	1000
7	Algorithm in [5]	54.97	125.01	176.51	29	75.70	1000

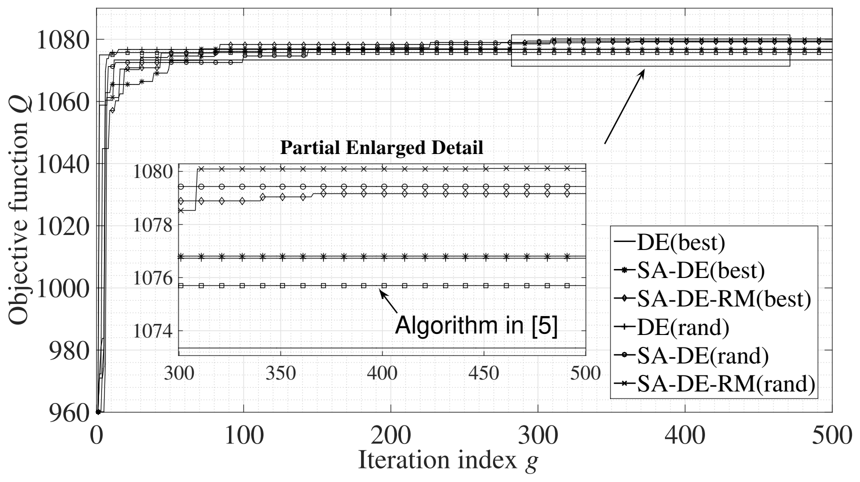


Fig. 4 The convergence of six algorithms for multi-target visiting problem

Fig. 4 shows the convergence result for multi-target visiting problem. As all the targets have been visited at least once, the second part of the objective function, J_b, is 1000, and J_t represents the priority-weighted function to describe the visiting time of targets. As the results shows, although all the six algorithm can give a result with 100% coverage, the SA-DE-RM (rand) algorithm obtains the best solution $J = 1080.11$ which is better than the result of algorithm in [5].

Fig. 5 gives the ground track of the optimal orbit, which visits the targets 17 times over the 29 revolutions in two days, and Table 9 shows the mission schedule of the best found result.

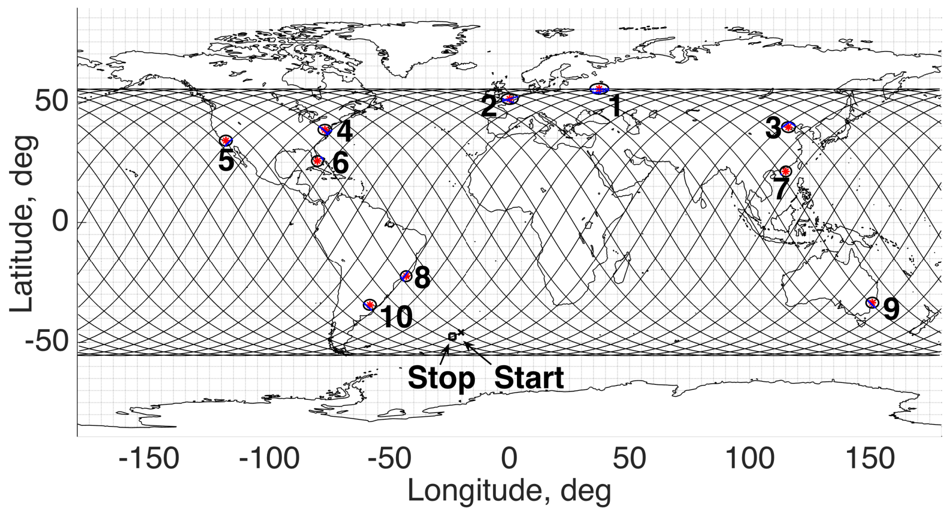


Fig. 5 Ground track of the optimal orbit

Table 9 Schedule Results

Target ID	Recurring ID: j	Start time $d-h:m:s$	Stop time $d-h:m:s$	Duration time
P_8	1	0-03:25:25	0-03:26:35	$t_{8,1}$=70
P_1	1	0-03:54:35	0-03:55:35	$t_{1,1}$=60
P_2	1	0-05:30:05	0-05:31:25	$t_{2,1}$=80
P_1	2	0-05:36:25	0-05:37:45	$t_{1,2}$=80
P_4	1	0-08:39:45	0-08:40:35	$t_{4,1}$=50
P_6	1	0-17:16:45	0-17:17:15	$t_{6,1}$=30
P_8	2	0-17:33:35	0-17:34:15	$t_{8,2}$=40
P_{10}	1	0-19:15:25	0-19:16:35	$t_{10,1}$ =70
P_3	1	0-20:06:35	0-20:07:35	$t_{3,1}$=60
P_3	2	1-02:59:05	1-02:59:45	$t_{3,2}$=40
P_1	3	1-04:25:45	1-04:27:05	$t_{1,3}$=80
P_7	1	1-04:44:55	1-04:45:05	$t_{7,1}$=10
P_9	1	1-05:02:45	1-05:03:55	$t_{9,1}$=70
P_1	4	1-06:07:55	1-06:08:35	$t_{1,4}$=40
P_2	2	1-09:26:05	1-09:27:05	$t_{2,2}$=60
P_5	1	1-10:45:15	1-10:46:25	$t_{5,1}$=70
P_4	2	1-16:03:25	1-16:04:45	$t_{4,2}$=80

V. Conclusions

This paper proposed a double self-adaptive differential evolution algorithm with a random mutant. When a random mutant and a double self-adaptive scaling factor are introduced into the traditional differential evolution algorithm, the scaling factors of the proposed algorithm can adjust with the optimization procedure and the algorithm can jump out of the local optimal. Different from the previous research, the self-adaptive scaling factors in this Note can be affected by not only the number of generations but also the fitness function of the last generation. When the algorithms are applied to several test function studies include low dimension and high dimension and compared with the other algorithms, the simulations demonstrated that the advanced algorithm can give a better performance in solution accuracy, convergence, and the results' standard deviation. The case studies presented in this Note prove the novel self-adaptive algorithm with random mutant can provide a better performance on multi-target, maneuver optimal problems than others.

VI. Acknowledgements

This work was supported in part by the National Natural Scientific Foundation of China (Grant No. 61690210, 61690211)

References

[1] Storn, R., and Price, K., "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, Vol. 11, No. 4, 1997, pp. 341–359.

[2] Price, K., Storn, R. M., and Lampinen, J. A., *Differential evolution: a practical approach to global optimization*, Springer Science & Business Media, 2006.

[3] Olds, A. D., Kluever, C. A., and Cupples, M. L., "Interplanetary mission design using differential evolution," *Journal of Spacecraft and Rockets*, Vol. 44, No. 5, 2007, pp. 1060–1070. doi:10.2514/1.27242.

[4] Madavan, N. K., "On improving efficiency of differential evolution for aerodynamic shape optimization applications," *AIAA Paper*, Vol. 4622. doi:10.2514/6.2004-4622.

[5] Chen, Y., Mahalec, V., Chen, Y., He, R., and Liu, X., "Optimal satellite orbit design for prioritized multiple targets with threshold observation time using self-adaptive differential evolution," *Journal of Aerospace Engineering*, Vol. 28, No. 2, 2013, p. 04014066. doi:10.1061/(ASCE)AS.1943-5525.0000393.

[6] Subramanian, S. V., and DeLaurentis, D. A., "A hybrid differential evolution self-organizing-map algorithm for optimization of expensive black-box functions," *AIAA Aviation-15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2014. doi:10.2514/6.2014-2730.

[7] Madavan, N. K., "Aerodynamic shape optimization using hybridized differential evolution," *AIAA Paper*, Vol. 3792. doi: 10.2514/6.2003-3792.

[8] Zhu, K.-J., Li, J.-F., and Baoyin, H.-X., “Satellite scheduling considering maximum observation coverage time and minimum orbital transfer fuel cost,” *Acta Astronautica*, Vol. 66, No. 1, 2010, pp. 220–229. doi:10.1016/j.actaastro.2009.05.029.

[9] Qin, A. K., and Suganthan, P. N., “Self-adaptive differential evolution algorithm for numerical optimization,” *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 2, IEEE, 2005, pp. 1785–1791. doi:10.1109/CEC.2005.1554904.

[10] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., and Tiwari, S., “Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization,” *KanGAL report*, Vol. 2005005, 2005, p. 2005.

[11] Curtis, H. D., *Orbital mechanics for engineering students*, Butterworth-Heinemann, 2013.

[12] Vallado, D. A., *Fundamentals of astrodynamics and applications*, Vol. 12, Springer Science & Business Media, 2001.