Hindawi Publishing Corporation
International Journal of Aerospace Engineering
Volume 2016, Article ID 3416046, 13 pages
http://dx.doi.org/10.1155/2016/3416046

Research Article

Multiple Gravity Assist Spacecraft Trajectories Design Based on BFS and EP_DE Algorithm

Mingcheng Zuo, Guangming Dai, Lei Peng, Maocai Wang, and Jinlian Xiong

School of Computer Science, China University of Geosciences, Wuhan 430074, China

Correspondence should be addressed to Guangming Dai; gmdai@126.com

Received 8 April 2016; Revised 5 June 2016; Accepted 14 June 2016

Academic Editor: Haibo Wang

The paper deals with the multiple gravity assist trajectories design. In order to improve the performance of the heuristic algorithms, such as differential evolution algorithm, in multiple gravity assist trajectories design optimization, a method combining BFS (breadth-first search) and EP_DE (differential evolution algorithm based on search space exploring and principal component analysis) is proposed. In this method, firstly find the possible multiple gravity assist planet sequences with pruning based BFS and use standard differential evolution algorithm to judge the possibility of all the possible trajectories. Then select the better ones from all the possible solutions. Finally, use EP_DE which will be introduced in this paper to find an optimal decision vector of spacecraft transfer time schedule (launch window and transfer duration) for each selected planet sequence. In this paper, several cases are presented to prove the efficiency of the method proposed.

1. Introduction

Multiple gravity assist in deep space exploration mission design can greatly reduce the fuel expense. In general, all the planets in solar system can be used to design the gravity assist trajectories. At the same time, each planet can be used more than once. So, there are so many possible planet sequences for the gravity assist trajectories design. For each possible gravity assist planet sequence, there also must be an outstanding algorithm which can find an optimal decision vector of spacecraft transfer time schedule. A global optimization method, such as differential evolution, is suggested for seeking solutions which are likely to be close to the global optimum. However, the optimal solution may not be found only with standard differential evolution, so there must be some improvements on it. How to design the gravity assist trajectories has attracted a lot of researchers to work on it, and lots of research achievements have been published [1–5]. Many multiple gravity assist missions such as Cassini, Rosetta, and Messenger have been the benchmarks announced by ACT (Advanced Concepts Team of European Space Agency) [1]. In addition, a combination of stochastic search algorithms and search algorithm is proposed to solve the problem of multiple gravity assist trajectory design by Vasile [2]. In his recent research, he also applies bionics to solving the problem [3]. Izzo et al. put forward the method of pruning search space based on reducing the search space with a series of constraints [4]. Hennes and Izzo also use Monte Carlo Tree to find a solution [5]; at the same time, Monte Carlo Tree search method has been the hottest technology for optimization, especially after the excellent performance of AlphaGo [6]. Hartmann et al. use stochastic search algorithms to optimize the space trajectories design [7]. However, search space pruning proposed by Izzo et al. also might delete a better solution, though it is an effective algorithm. In paper [8], clustering was used to decide where to branch. In the study of the problem, it is difficult to get a better result if we only use DE and other search algorithms to search the optimal trajectory. In Cassini, it seems to be hard when we try to find a better solution than 5.3 km/s if only using DE algorithm [9]. So, we also use combinatorial method to deal with multiple gravity assist trajectories design.

In addition, in this paper, breadth-first search was also mentioned. It is true that BFS can lead to an optimal solution if there is enough calculation expense. Frequently, some pruning methods must be used for BFS. However, usually there is no suitable pruning method that can be found easily.

So some computer calculating methods are used here. Korf and Schultze propose the method of how to design large-scale parallel breadth-first search which can get a result in less calculation time [10]. Similarly, Zhu and Cheung and Awerbuch and Gallager are devoted to a new distributed breadth-first search algorithm [11, 12]. Bisson et al. present the results obtained by using an evolution of CUDA-based solution, via a breadth-first search, for the exploration [13]. Katsuta et al. propose a tightly coupled accelerator which can accelerate breadth-first search [14]. Of course, in order to improve the efficiency of BFS, Liu et al. finish some improvements to it [15].

When designing the multiple gravity assist trajectories, we must find the most possible gravity assist planets sequence and furthermore find the optimal solution of gravity assist time sequence [16]. In this progress, one possible planets sequence is usually determined by deterministic algorithms, such as BFS algorithm, and the optimal gravity assist time solution is frequently obtained by using heuristic algorithms [2]. But for heuristic algorithms including DE, GA, and PSO, because of the complexity of search space, there is no universally valid method which can deal with the multiple gravity assist trajectories optimization. Therefore, in this paper, the EP_DE algorithm (we also can call it EP_DE I algorithm, because there may be some improved variants in the future research) is proposed based on DE algorithm, whose purpose is to reduce the global search space and improve the performance of DE algorithm. Also, this method of reducing search space is also available for other heuristic algorithms, which we will prove from the experiments presented in this paper.

We are devoted to finding a heuristic search algorithm based on machine learning, and EP_DE algorithm proposed in this paper can be seen as a basic version of this idea. And the feasibility of the algorithm has been proved by the benchmarks Cassini and GTOC1. With the development of hardware technology, heuristic algorithms based on machine learning will become a new research direction. So, EP_DE algorithm has important reference meaning for the development of heuristic search algorithm based on machine learning.

Obviously, compared to previous studies, the advantages of the method proposed in this paper are the guidance strategy based on data analysis. In prior studies, basic aerospace dynamics knowledge is usually used to reduce the global search space [4], while, in this paper, statistical features of a large number of data pieces are used to guide seeking the local space of optimal value. Therefore, EP_DE algorithm makes full use of the historical data generated in the process of optimization, rather than simply depending on some basic pruning method, such as angle constraint.

In this paper, a method combining BFS (breadth-first search) and EP_DE (differential evolution algorithm based on search space exploring and principal component analysis) is proposed. Firstly, find the possible multiple gravity assist planet sequences with pruning BFS; here, some thresholds have been set to prune the search space. In order to judge the efficiency of the planet sequence, standard differential evolution algorithm is applied to find the initial decision of the planet sequence. And it should be noted that the reason why we use standard differential evolution algorithm here is that it can cost less time. Then, select out the better planet sequence from all the possible sequences. Finally, find the optimal decision vector of spacecraft transfer time schedule for the planet sequence selected with EP_DE algorithm.

The rest of the paper is organized as follows. Section 2 briefly describes multiple gravity assist calculation model. In Section 3, the proposed method is presented in detail. Two experiments based on Cassini and GTOC1 are introduced in Section 4 which can prove the efficiency of EP_DE algorithm. Experiment result analysis and trajectory simulation are in Section 5. Finally, Section 6 is devoted to conclusion and future work.

2. Multiple Gravity Assist Calculation Model

In general, there are two kinds of MGA (multiple gravity assist) trajectory design: PMGA (Powered Multiple Gravity Assist) and Unpowered Multiple Gravity Assist. In this paper, assume that the final planet is a fixed destination, and dynamical model in the numerical test cases is MGA with powered flyby. PMGA is the method of applying a pulse to the spacecraft when it flies by a planet. And Unpowered Multiple Gravity Assist is the method of applying a pulse to spacecraft in heliocentric coordinate system.

In the trajectory design of PMGA, the trajectory can be seen as a combination of several partial trajectories which are solved as Lambert problems. It can be seen from Figure 1. In order to introduce the calculation model, we now define some mathematical symbols.

The position vector of spacecraft in the heliocentric coordinate system is $r_{sc}(\mathbf{t})$. The velocity vector of spacecraft in the heliocentric coordinate system is $v_{sc}(\mathbf{t})$. The position vector of gravity assist planet in the heliocentric coordinate system is $p_p(\mathbf{t})$. The velocity vector of gravity assist planet in the heliocentric coordinate system is $v_p(\mathbf{t})$. The time before spacecraft flies by the gravity assist planet is t_{GA}^-, and the time after spacecraft flies by the gravity assist planet is t_{GA}^+.

Before the spacecraft flies by the gravity assist planet, the velocity of the spacecraft relative to the velocity of planet is

$$v_\infty\left(t_{GA}^-\right) = v_{sc}\left(t_{GA}^-\right) - v_p\left(t_{GA}^-\right). \tag{1}$$

After the spacecraft flies by the gravity assist planet, the relative velocity is

$$v_\infty\left(t_{GA}^+\right) = v_{sc}\left(t_{GA}^+\right) - v_p\left(t_{GA}^+\right). \tag{2}$$

Of course, the value of the two relative velocities is the same. The value can be represented to be

$$v_\infty = \left|v_\infty\left(t_{GA}^-\right)\right| = \left|v_\infty\left(t_{GA}^+\right)\right|. \tag{3}$$

If the angle between $v_\infty(t_{GA}^-)$ and $v_\infty(t_{GA}^+)$ is represented by δ, the relationship between them is

$$v_\infty\left(t_{GA}^-\right) \cdot v_\infty\left(t_{GA}^+\right) = v_\infty^2 \cos\delta. \tag{4}$$

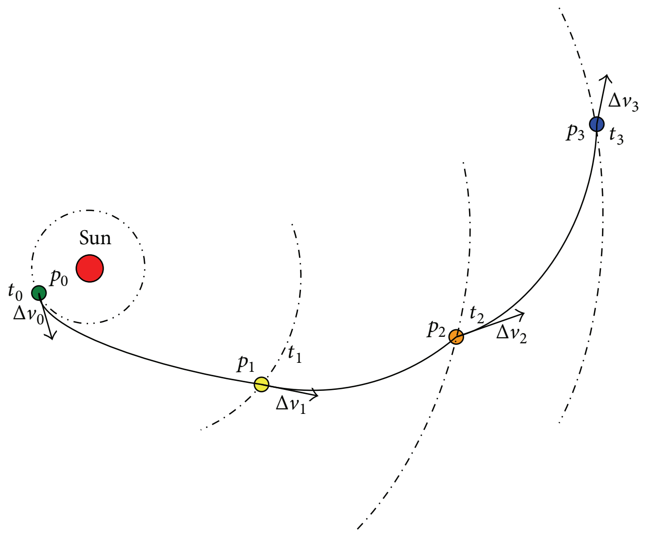

FIGURE 1: Multiple gravity assist trajectory.

Assume that the planetary gravitational constant is μ_p, and the distance between spacecraft and planet center is R_{GA}. δ can be obtained by

$$\delta = 2 \cdot \arcsin \frac{\mu_p}{v_\infty^2 R_{\mathrm{GA}} + \mu_p}. \tag{5}$$

The value range of δ is $[0^\circ, 180^\circ]$.

Multiple gravity assist spacecraft trajectory design relies on two partial decision vectors. One is the sequence of planet; the other is decision vector of spacecraft transfer time schedule. Determine the sequence of planets and then solve the sequence of launch window and transfer duration. The calculation model is described as follows.

One partial decision vector is

$$\mathbf{p} = \left[p_0, p_1, p_2, \ldots, p_{N+1}\right]^{\mathbf{t}}. \tag{6}$$

p_0 is Earth; p_i is each flyby planet. p_{N+1} is the objective planet. And another partial decision vector is

$$\mathbf{t} = \left[t_0, t_1, t_2, \ldots, t_{N+1}\right]^{\mathbf{t}} \in I = I_0 \times I_1 \times \cdots \times I_{N+1}. \tag{7}$$

t_0 is the time when spacecraft is launched from Earth, t_i is the time when it flies by a planet, and t_{N+1} is the time when arrives at the objective planet. $t_{i+1} - t_i$ is the transfer time interval from p_i to p_{i+1}.

Decision vector is

$$\mathbf{x} = \{\mathbf{p}, \mathbf{t}\}. \tag{8}$$

In each partial trajectory solved as a Lambert problem, there is a delta-V. Based on the decision vector, the objective function can be expressed as follows:

$$\begin{aligned} \min \quad & f(x) = \sum_{i=0}^{N+1} \Delta V_i(x) \\ \text{Subject to} \quad & r_{p,i}(\mathbf{x}) \geq r_{p,i}^{\min}, \quad i = 1, \ldots, N \\ & \Delta V_i(\mathbf{x}) \leq \Delta V_i^{\max}, \quad i = 0, 1, \ldots, N \\ & \Delta V_{N+1}(\mathbf{x}) \leq \Delta V_{N+1}^{\max}, \end{aligned} \tag{9}$$

where ΔV_0 is the delta-V which can help spacecraft escape from Earth. $\Delta V_0^{\max}$ is the max velocity constraint.

Meanwhile the delta v applied here is

$$\begin{aligned} \Delta v_i(\mathbf{x}) = & \sqrt{\left|v_\infty\left(t_{\mathrm{GA}}^{+}\right)\right|^2 + \frac{2\mu_p}{R_{\mathrm{GA}}}} \\ & - \sqrt{\left|v_\infty\left(t_{\mathrm{GA}}^{-}\right)\right|^2 + \frac{2\mu_p}{R_{\mathrm{GA}}}}. \end{aligned} \tag{10}$$

So we can know that $\Delta v_i(\mathbf{x})$ is decided by R_{GA}, also can find the answer to $\partial \Delta v_i(\mathbf{x}) / \partial R_{\mathrm{GA}} = 0$, and go on to obtain the minimal $\Delta v_i(\mathbf{x})$.

The partial trajectory between two planets is solved as a Lambert problem. It is a two-point boundary value problem,

$$\begin{aligned} & \ddot{r}_{\mathrm{sc}} - \frac{r_{\mathrm{sc}}}{\left|r_{\mathrm{sc}}\right|^3} = 0 \\ \text{Subject to} \quad & r_{\mathrm{sc}}(t_i) = r_p(t_i) \\ & \qquad (i = 0, 1, \ldots, N, N+1). \end{aligned} \tag{11}$$

When time is t_i, the position of spacecraft is $r_{\mathrm{sc}}(t_i)$, and the position of gravity assist planet is $r_p(t_i)$.

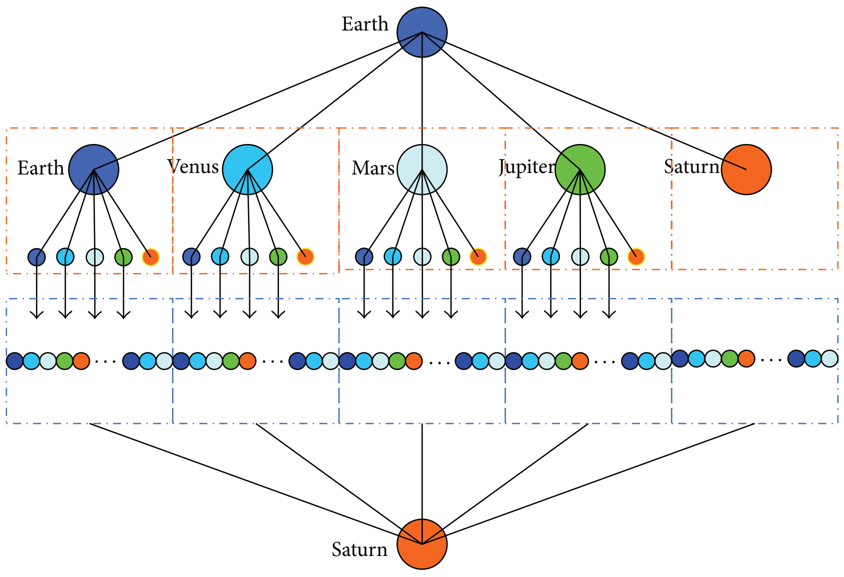

FIGURE 2: Tree searching with BFS.

If there are $N + 1$ partial trajectories, the problem can be solved by using $N + 1$ Lambert calculations.

Finally, when combining two partial trajectories, we should find the suitable $v_{sc}(t_i^-)$ and $v_{sc}(t_i^+)$ subject to

$$v_{sc}(t_i^-) = v_{sc}(t_i^+). \tag{12}$$

If $v_{sc}(t_i^-) \neq v_{sc}(t_i^+)$, there will be

$$\Delta v_i = v_{sc}(t_i^+) - v_{sc}(t_i^-). \tag{13}$$

3. Trajectory Design Based on BFS and EP_DE Algorithm

In this section, we will introduce the method proposed in detail. The introduction of the method is divided into two parts. One is how to use pruning based BFS to seek the gravity assist planet sequence, and the others are the process and efficiency of EP_DE algorithm.

There will be many possible planet sequences when optimizing the gravity assist trajectory if we do not prune the search space. So, some algorithm improvements about BFS must be proposed.

Assume that m planets are available and can be used for the gravity assist trajectory design, and the maximum times of gravity assist are n. So the search space will be a search tree with n layers and m branches in each layer. It can be presented in Figure 2. In this paper, the experiment is based on the trajectory from Earth to Saturn. So, in each possible planet sequence, the objective planet is always Saturn. In general, the available planets are Venus, Earth, Mars, and Jupiter. That is to say, m is 5 when counting in Saturn.

Pruning methods appear to be especially important while there are so many branches. In addition, when the pruning method is not suitable, the optimal solution obtained will be not ideal. The reason why we only obtain some suboptimal solutions will be introduced in this paper.

As for how to prune the search space, the most direct method is to set a threshold value, although it is difficult to determine the threshold. In the experiment, the threshold is determined by experience. The value of each solution is obtained by standard DE algorithm which can estimate the efficiency of the gravity assist planet sequence. Then, pick out the most possible planet sequences, and go on to calculate the optimal value of each planet sequence picked out with EP_DE algorithm. Next, there is the introduction of the EP_DE proposed in this paper. When calculating the optimal value of the mission Cassini at which gravity assist planet sequence is EVVEJS (Earth, Venus, Venus, Earth, Jupiter, and Saturn), we usually get 5.3 km/s.

When using standard differential evolution algorithm to optimize trajectory design problem, it can be found that usually the suboptimal value valley is much larger than a better value valley in spatial size, so it is hard to find a better solution. In order to solve this special problem, the algorithm proposed in this paper is showed in Algorithm 1 EP_DE. An earlier version of this algorithm was presented at the International Congress on Evolutionary Computation. The algorithm uses differential evolution algorithm based on search space exploring and PCA (principal component analysis), so it is called "EP_DE."

Algorithm 1 (EP_DE).

Inputs

Number of initialization samples: Pop0.

Number of samples retained: Pop1.

Number of clusters: Pop2.

Number of dimension division: Pop3.

Outputs

Solution vector: $\mathbf{x} = \{\mathbf{p}, \mathbf{t}\}$.

Table 1: Value range of each variable in Cassini.

Variable	t_0 (MJD2000)	t_1 (d)	t_2 (d)	t_3 (d)	t_4 (d)	t_5 (d)
Min	−1000	30	100	30	400	1000
Max	0	470	400	2000	2000	6000

Table 2: Grid-initialization number for each variable in Cassini.

t_0 (MJD2000)	t_1 (d)	t_2 (d)	t_3 (d)	t_4 (d)	t_5 (d)
30	30	30	100	100	200

Step 1. Initialize Pop0 population samples and calculate the function values of all samples. Rank all the values of the function ascending order. Reserve the top Pop1 samples.

Step 2. Use K-mean clustering algorithm to cluster Pop1 samples into Pop2 communities. In general, we need C communities, and the value range of it is usually from 3 to 5.

Step 3. Choose the community where we can find a better value. And continue to find the densest direction of samples distribution by PCA.

Step 4. Divide the community reserved into Pop3 parts in the densest direction of samples distribution. Use differential evolution algorithm to search optimal value of the small space in every partition obtained.

The main idea of algorithm is to explore the relationship between function value and variables. For one thing, estimate the general position of function value valley by initializing P0 samples in global search space; for another thing, learn the densest dimension of samples distribution in local search space.

In order to show the validity of method proposed in this paper, we use it to find the optimal solution in benchmark Cassini1 and GTOC1.

4. Efficiency of EP_DE Algorithm

4.1. Search Space Exploring. The meaning of search space exploring is to know the change of the function value in the search space by mathematical statistics.

There are two methods of search space exploring: method one is grid-initializing by using a large number of samples to know the relationship between search space function value and the variables change, which we call GI method. Method two randomly initializes a few parts of samples to know about the search space well, which is called method RI. For example, we initialize 25682962 samples in the experiment when using method RI and select the 500 best samples eventually. Apparently, the feature of method one in experiment results is low speed and high repeatability. When using method two, randomly initialize 2000 samples and pick out 500 excellent samples whose function values are outstanding. Apparently, the feature of method two in experiment results is high speed and low repeatability. In addition, the time system is JMD2000.

In the experiment of Cassini1, we use method one to explore the search space, and value range for each variable is in Table 1. Grid-initialization number on each variable is in Table 2. The grid number on each variable is determined according to the length of the variable. In the experiment of GTOC1, we use method two to explore the search space, and value range for each variable is in Table 3.

4.2. Clustering. In the experiment of Cassini, cluster selected 500 samples into five communities. Principal component analysis was performed on the samples in each community we selected. Also there is a mechanism to ensure that there are enough samples in each community for the PCA to be statistically meaningful. If, in a community, there are no enough meaningful samples, we must produce some new samples around the samples reserved by normal distribution. Because the best solution was found in the fourth community, experiment result of the fourth community is showed here. Its samples number is 67, and the value range of community is showed in Table 4.

In the experiment of GTOC1, the number of clusters is 3. Table 5 is spatial range of each community after clustering. Because we use method one to solve Cassini and method two to deal with GTOC1, therefore, in each experiment of GTOC1, the results will be different due to the instability of method two. But for it, if there are a suitable number of random initialization samples, the results also tend to be stable. So, in Table 5, we show two results in (a) and (b) for different experiments when the samples number is suitable. In order to know the location of the selected community clearly, we show the community range in Figures 3 and 4. In addition, the range in Figure 4 is for the community in Table 5(a).

4.3. Principal Component Analysis. Gain the covariance matrix by the fourth community samples, and eigenvalues and eigenvectors of the covariance matrix are calculated. From left to right, the characteristic value is arranged in the successive increasing order. The feature vector is represented as a column vector, and the feature vector corresponding to the characteristic value is also arranged from left to right in Table 6. Here, the sixth component vector with lowest variance and highest "density" is used to divide the fourth community. In fact, the direction of the sixth component is similar to t_3. Because of the calculation convenience, we can directly divide the community space into p3 parts in the direction of t_3.

Optimal value found is in the second community, and two PCA results which mean there are two experiments related to Table 5 are showed in Table 7. For each range of the community we obtain that in every experiment there must be one only PCA result for it. So, Tables 7(a) and 7(b) are the results for Tables 5(a) and 5(b), respectively. We still use feature

Table 3: Value range of each variable in GTOC1.

	t_0 (MJD2000)	t_1 (d)	t_2 (d)	t_3 (d)	t_4 (d)	t_5 (d)	t_6 (d)	t_7 (d)
Min	3000	14	14	14	14	100	366	300
Max	10000	2000	2000	2000	2000	9000	9000	9000

Table 4: Value range of the fourth community in Cassini.

Variable	t_0 (MJD2000)	t_1 (d)	t_2 (d)	t_3 (d)	t_4 (d)	t_5 (d)
Min	−882	148	386	34	898	4504
Max	−697	204	400	97	1220	4891

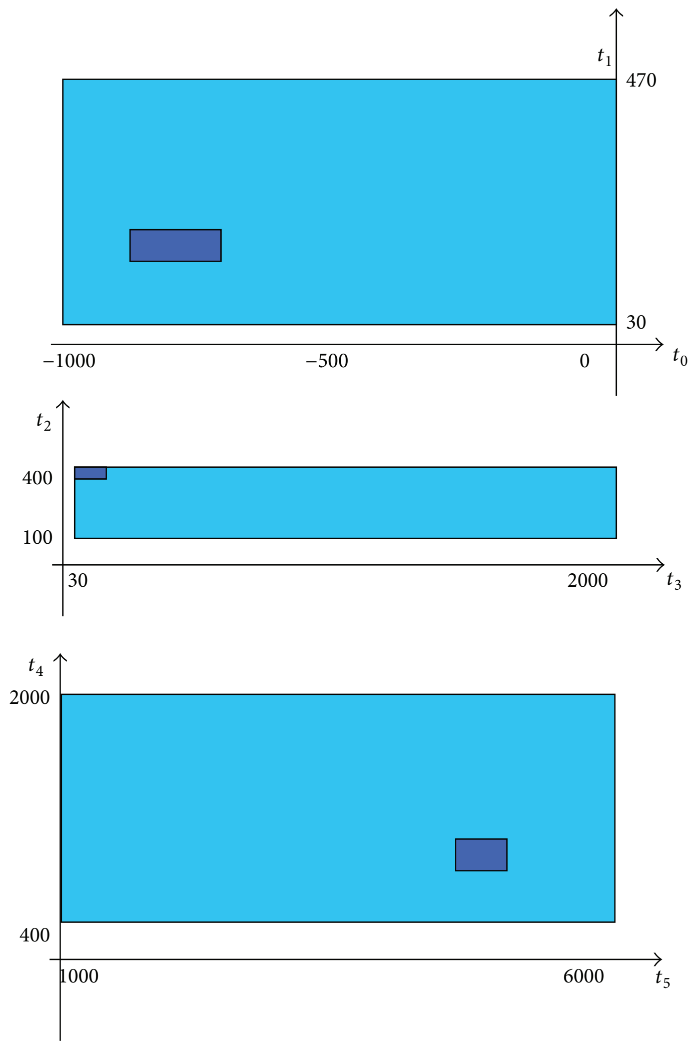

Figure 3: Community range in Cassini.

vector matrix to show the results. The eighth component is the densest dimension of samples distribution. From this direction we might find several function value channels.

4.4. Computational Results. In order to prove the efficiency of the EP_DE algorithm, the different calculation results are compared here by using DE and EP_DE. Based on the idea of improvement for DE, we also try some other improvements for PSO algorithm and GA algorithm, and they are called EP_PSO and EP_GA. 20 independent runs are performed by each algorithm. In Table 8, there are the best results obtained by standard algorithms including DE, PSO, and GA, while in Table 9, there are the best results gained by using improved algorithms including EP_DE, EP_PSO, and EP_GA.

Table 5: The spatial range of each community in GTOC1.

(a)

Community		t_0 (MJD2000)	t_1 (d)	t_2 (d)	t_3 (d)	t_4 (d)	t_5 (d)	t_6 (d)	t_7 (d)
1	Max	9991	1993	1995	1979	1993	8985	8994	8983
1	Min	4070	62	16	26	46	168	3893	3869
2	Max	9989	1996	1991	1984	1986	8991	6338	8982
2	Min	4865	27	17	19	56	3439	428	3007
3	Max	9997	1997	1996	1982	1993	8997	8989	6087
3	Min	5203	21	29	23	18	2930	2376	320

(b)

Community		t_0 (MJD2000)	t_1 (d)	t_2 (d)	t_3 (d)	t_4 (d)	t_5 (d)	t_6 (d)	t_7 (d)
1	Max	9989	1995	1991	1982	1997	8990	8991	8992
1	Min	4265	40	23	17	54	144	3907	3923
2	Max	9976	1994	1993	1979	1992	8995	6344	8992
2	Min	4877	16	22	30	27	3504	445	3102
3	Max	9994	1992	1989	1991	1986	8988	8975	6064
3	Min	5157	32	22	34	27	3011	2476	396

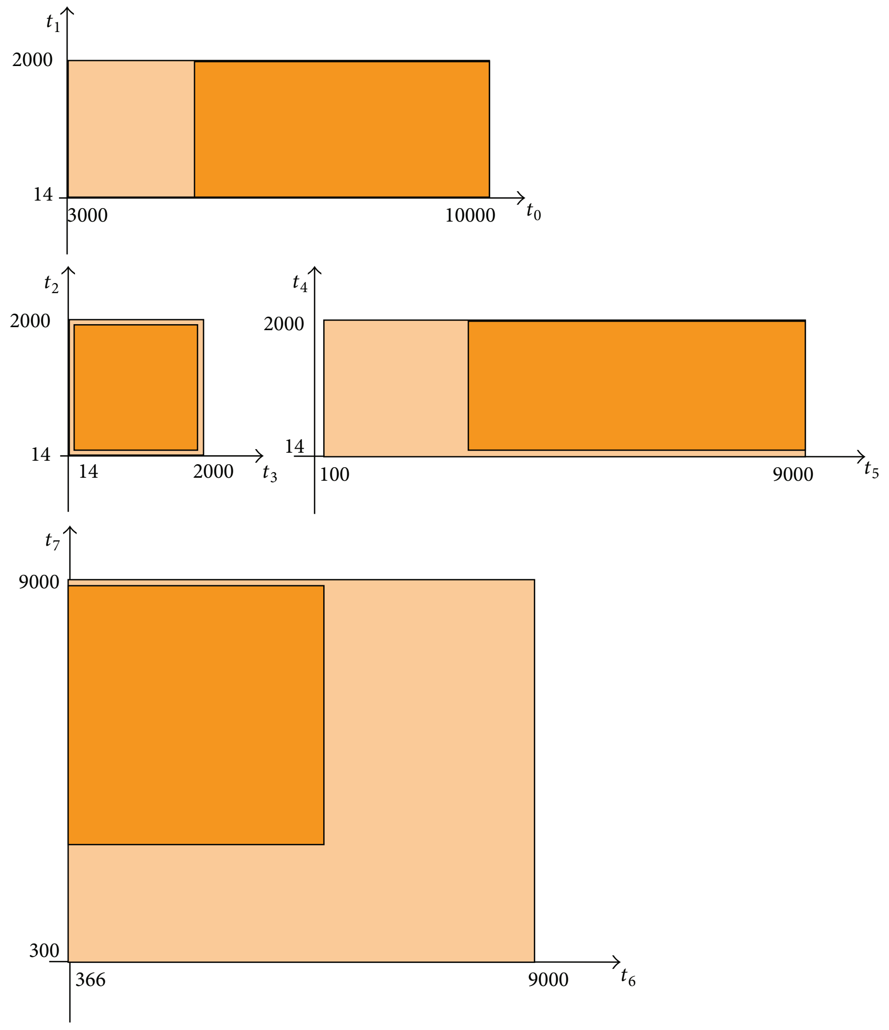

Figure 4: Community range in GTOC1.

Table 6: Feature vector matrix in Cassini.

First component	Second component	Third component	Fourth component	Fifth component	Sixth component
0.0063	−0.0082	−0.9998	−0.0117	0.0015	−0.0119
−0.0041	−0.0150	−0.0121	0.8167	0.5020	0.2839
−0.0025	0.0150	−0.0055	0.5634	−0.8000	−0.2059
0.0009	−0.0121	0.0103	0.1239	0.3279	−0.9364
−0.0076	0.9996	−0.0082	0.0051	0.0235	−0.0041
0.9999	0.0077	0.0062	0.0048	−0.0000	0.0016

Table 7: Feature vector matrix in GTOC1.

(a)

First component	Second component	Third component	Fourth component	Fifth component	Sixth component	Seventh component	Eighth component
−0.3850	−0.2518	−0.2510	0.1105	−0.3493	0.1704	0.2427	0.7093
−0.0492	0.3120	0.3859	−0.6696	−0.3199	0.4230	−0.1061	0.1021
−0.2776	0.6459	−0.5983	−0.1580	0.2905	0.0622	0.1809	−0.0423
0.4842	0.4185	−0.1284	0.3069	−0.6616	−0.0948	0.1762	−0.0452
−0.4723	0.3972	0.5264	0.2188	−0.0206	−0.5224	0.0346	0.1403
0.3660	0.0344	−0.1872	−0.3740	0.0992	−0.5419	−0.3616	0.5057
0.3856	0.0454	0.2862	−0.0729	0.3909	0.0490	0.7312	0.2686
0.1794	0.2930	0.1400	0.4767	0.3008	0.4585	−0.4466	0.3675

(b)

First component	Second component	Third component	Fourth component	Fifth component	Sixth component	Seventh component	Eighth component
−0.1287	0.0755	−0.5779	0.4540	−0.1094	0.2181	−0.1042	−0.6060
−0.0511	−0.0347	0.1628	−0.4753	−0.6551	0.5042	0.1035	−0.2230
−0.0646	−0.1284	−0.0644	−0.1991	0.5942	0.3844	0.6374	−0.1686
0.0696	0.0040	0.0960	−0.2124	0.4270	0.4605	−0.7376	−0.0494
−0.1124	0.0311	−0.1876	−0.6225	0.1176	−0.5663	−0.1099	−0.4659
−0.0825	−0.1345	0.7530	0.3088	0.0690	−0.1201	0.0018	−0.5418
0.7923	0.5644	0.0581	−0.0158	0.0225	0.0072	0.1200	−0.1873
0.5697	−0.7994	−0.1437	0.0071	−0.0664	−0.0574	−0.0402	−0.0799

Table 8: The best results using DE, PSO, and GA.

Algorithm	Optimal value (km/s)	Independent runs
DE	5.30	20
PSO	5.30	20
GA	6.83	20

Table 9: The best results using improved heuristic algorithm.

Algorithm	Optimal value (km/s)	Independent runs
EP_DE	4.93	20
EP_PSO	4.94	20
EP_GA	5.80	20

4.5. The Relation between the Optimal Value and the Step Size. When dividing the community into several partitions on the selected dimension, the step size of the partition determines the optimal value we can find. Set different width of the partition and study the relationship between width of partition and the optimal value. The experiment results are in Table 10. From it, we can find that the calculation result of differential evolution algorithm is more stable than particle swarm optimization algorithm and genetic algorithm by the change of partition width.

Dividing the space in the selected dimension, we can find that step size of partition determines the optimal value. Set different partition width and study the relationship between partition width and optimal value. Experiment result is in Table 11. First of all, divide each community into several partitions with grid width of 1000 days and use differential evolution algorithm to find optimal value in each partition of community. Then because the best solution is found in the second community, we go on dividing the second community with grid width of 500 days, which can contribute to a better solution.

Table 10: Relationship between width of partition and optimal value.

Algorithm	Width (days)	Optimal value (km/s)	Independent runs
EP_DE	20	4.93071	5
EP_DE	16	4.93071	5
EP_DE	12	4.93071	5
EP_DE	8	4.93071	5
EP_PSO	20	4.97	5
EP_PSO	16	4.97	5
EP_PSO	12	4.97	5
EP_PSO	8	4.938560	5
EP_GA	20	5.83	5
EP_GA	16	5.81	5
EP_GA	12	5.80	5
EP_GA	8	5.80	5

Table 11: Relationship between partition width and optimal value.

Width (days)	Community	Optimal value (change in semimajor axis)
1000	1	−1050910.520645
1000	2	−1213981.217776
1000	3	−1054529.474693
500	2	−1335213.053407

4.6. Verification. In the experiment of Cassini1, we ever tried to use second screening to find the optimal value, but the optimal value was lost. So there might be rapid growing of function value near the optimal value. When exploring search space, the optimal value is also removed. We fix four of all variables and draw the relationship between function value and the remaining two variables t_2 and t_3 in Figure 5.

As can be seen from Figure 5, there is rapid growing of function value. When value range of t_2 is set from 449.3857 to 449.3860, value range of t_3 is set from 54.7115 to 54.7119, function value is mainly affected by t_2. Still use this method to view the relationship between the function value and the other variables in Figures 6, 7, 8, and 9. These graphs can also verify the analytical results in PCA.

Differential evolution algorithm factors have a great influence on results. The results of experiments are summarized and analyzed here. When there are 200 samples and 800 generations, best result we found is −473311.67197 km. When there are 2000 samples and 2000 generations, best result we found is −885276.164762 km. After clustering, use differential evolution algorithm to find the optimal value in each community. Each optimal value is showed in Table 12. The optimal value we found with the method proposed is in Table 13. Use ten experiments to count the repeatability of the algorithm; result is in Table 14. If there are more variables, the repeatability of the algorithm needs to be improved.

The global search space is estimated by the function value of samples in the global space. The community is estimated by PCA in each community. EP_DE algorithm can obtain the

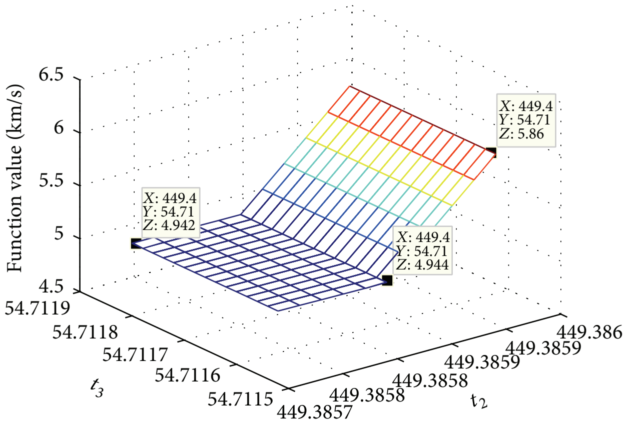

Figure 5: Relationship between function value change and variables t_2 and t_3.

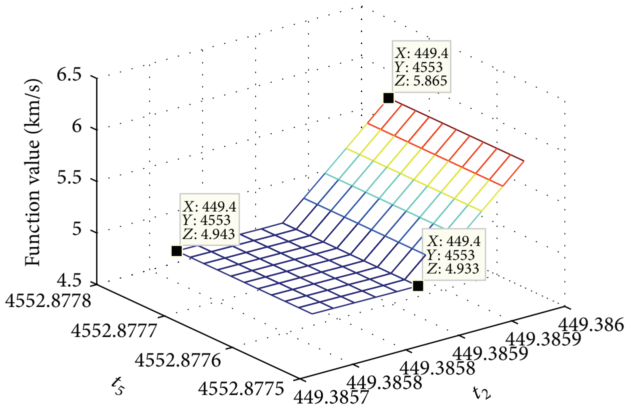

Figure 6: Relationship between function value change and variables t_2 and t_5.

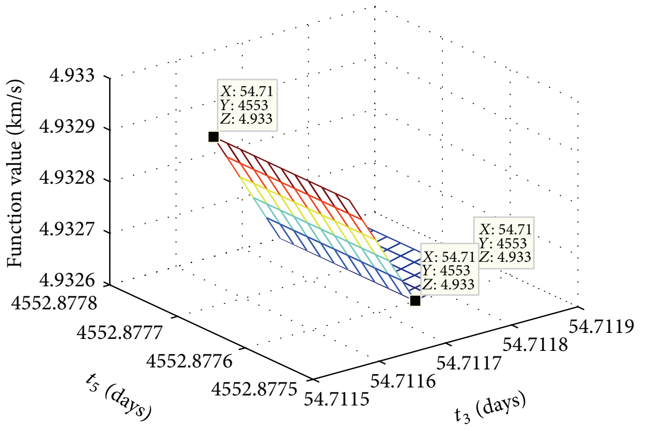

Figure 7: Relationship between function value change and variables t_3 and t_5.

distribution of search space through this statistical method. Also, its efficiency in different problems is also different. In the calculation of the Cassini1, the results are relatively stable, while, in the calculation of the GTOC1, experiment result is still far from the optimal value that had already been

TABLE 12: Optimal value of each community.

Algorithm	Community number	Optimal value (change in semimajor axis)	Independent runs
DE	One	−951717.294079	5
DE	Two	−949732.022191	5
DE	Three	−892186.731365	5

TABLE 13: Optimal value of each community.

Algorithm	Community number	Optimal value (change in semimajor axis)	Independent runs
DE	One	−1050910.520645	5
DE	Two	−1213981.217776	5
DE	Three	−1054529.474693	5

TABLE 14: Stability of the algorithm.

Population	Generation	Optimal value (change in semimajor axis)	Worst value	Standard deviation	Independent runs
2000	2000	−1271203.511630	−1004813.552603	89121	10

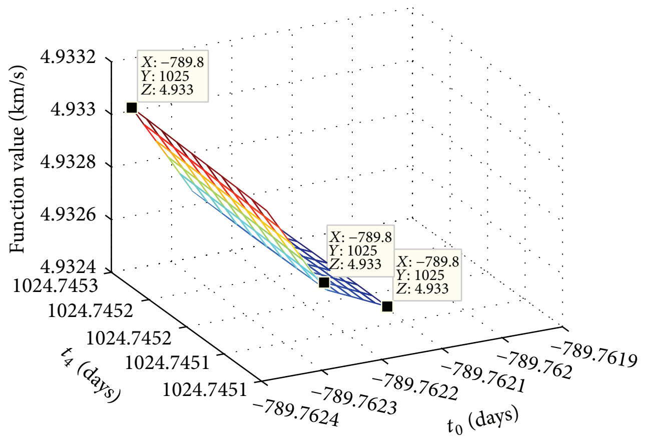

FIGURE 8: Relationship between function value change and variables t_0 and t_4.

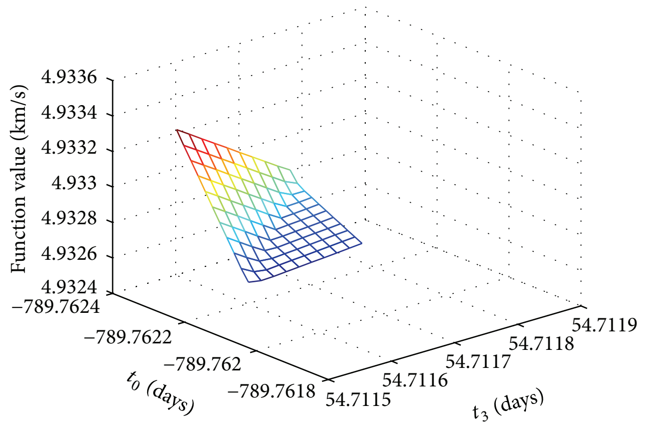

FIGURE 9: Relationship between function value change and variables t_0 and t_3.

announced. We will test the efficiency of EP_DE by other benchmarks in the next work.

4.7. Trajectory Simulation. Show the trajectory simulation based on the solution we have found. The trajectories simulations of Cassini and GTOC1 are in Figure 10.

5. Seek the Optimal Gravity Assist Planet Sequence for Cassini

5.1. Set Transfer Durations. The gravity assist planet sequence of above Cassini experiment mentioned is EVVEJS announced by ACT in ESA. And the best value found in Cassini with EVVEJS is 4.93 km/s. Of course, there may be a better gravity assist planet sequence. Next, we will try to seek a better solution with the method proposed in this paper.

In general, the value of the solution could be affected by transfer duration between two planets. In Tables 15 and 16, there is transfer duration between random two planets.

5.2. Set Pruning Threshold. The search computation expense will be high if there is no pruning threshold. However how to find the suitable threshold is difficult. But there is some prior knowledge that can be used by us. Firstly, while seeking the optimal value in Cassini with DE algorithm, the result found usually comes to 5.3 km/s. Then, the maximum energy expense of direct transfer trajectory is no more than 15 km/s. So, in the former search process, in order to prevent the pruning of possible solution, the threshold we set is 20 km/s while in the later search process, in order to reduce the computation expense, the threshold we set is 6 km/s. In addition, sometimes, it cannot hold true that more gravity assist planets can reduce the total fuel expense. For example, in the experiment, we find that the fuel expense of planet

Table 15: The minimum transfer duration between random two planets.

	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Mercury	10	25	25	50	200	100	1000	2500
Venus	25	50	15	150	250	200	1500	3000
Earth	25	15	50	200	200	50	50	50
Mars	50	150	200	150	500	500	2500	2500
Jupiter	200	250	200	500	1000	500	2500	2500
Saturn	100	200	50	500	500	250	500	500
Uranus	1000	1500	50	2500	2500	500	5000	20000
Neptune	2500	3000	50	2500	2500	500	20000	10000

Table 16: The maximum transfer duration between random two planets.

	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Mercury	300	600	450	1200	9000	1500	75000	120000
Venus	600	705	600	1200	9000	1500	75000	120000
Earth	450	600	750	1500	3000	900	750	750
Mars	1200	1200	1500	1500	12000	9000	75000	120000
Jupiter	9000	9000	3000	12000	7500	9000	90000	135000
Saturn	1500	1500	900	9000	9000	3000	75000	120000
Uranus	75000	75000	750	75000	90000	75000	75000	300000
Neptune	120000	120000	750	120000	135000	120000	300000	120000

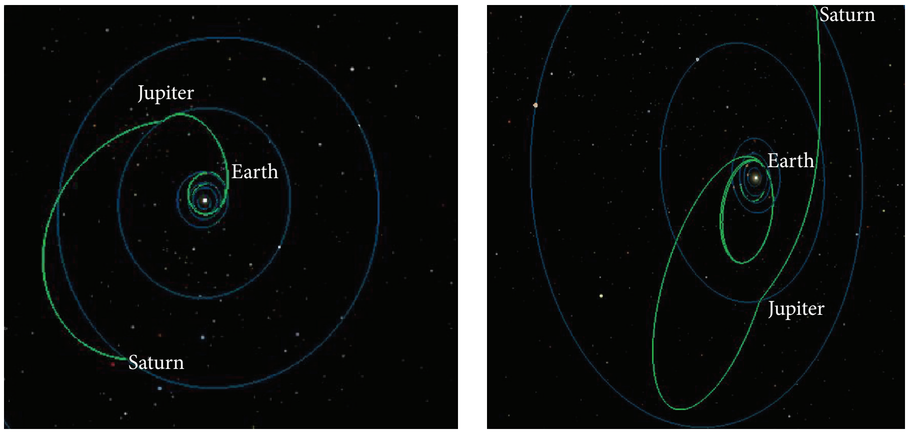

Figure 10: Trajectories simulation of Cassini (a) and GTOC1 (b).

sequence EVVS is higher than that EJS, while the fuel expense of planet sequence EVVEJS is lower than that EJS.

5.3. Experiment Result. The planet sequence we gain with method proposed is EVVEEJS. Every time the experiment result obtained by EP_DE algorithm seems to be not stable. In several experiments we have finished, there are three different results shown in Table 17.

5.4. Analysis. The feature of the optimal solution is obvious when seeking the optical solution of multiple gravity assist trajectory. It looks like looking for a needle in a sea; that is to say, it is so hard to find the optimal solution. In this paper, when fixing the gravity assist planet sequence and looking for the optimal solution with EP_DE algorithm, the method proposed was proved to be effective, although sometimes there may not be perfect performance. The value of Cassini obtained in this paper is better than that announced by ACT in ESA. Likewise, this method can be used for any problems of multiple gravity assist trajectory design.

6. Conclusions

All of the above experiments can illustrate the efficiency of the method proposed in this paper, especially the availability of EP_DE algorithm. Experimental result of Cassini has the

Table 17: Different results of Cassini for EVVEEJS.

Algorithm	Planet sequence	Total Δv	Experiments t_i (day)	Δv_i (km/s)	R_p (km)
EP_DE	EVVEEJS	4.425189 km/s	t_0: −194.933259	Δv_0: 3.592163	R_{p1}: 10438.997785
			t_1: 175.412664	Δv_1: 0.000084	R_{p2}: 10138.617395
			t_2: 435.455190	Δv_2: 0.000193	R_{p3}: 13664.926677
			t_3: 58.615710	Δv_3: 0.001033	R_{p4}: 6780.248744
			t_4: 623.697390	Δv_4: 0.376078	R_{p5}: 7803345.772169
			t_5: 2480.085580	Δv_5: 0.000065	
			t_6: 2915.843899	Δv_6: 0.455572	
EP_DE	EVVEEJS	4.424657 km/s	t_0: −194.342712	Δv_0: 3.592019	R_{p1}: 10447.177963
			t_1: 174.935300	Δv_1: 0.000311	R_{p2}: 10056.969935
			t_2: 435.396489	Δv_2: 0.000238	R_{p3}: 13639.312831
			t_3: 58.609530	Δv_3: 0.000083	R_{p4}: 6778.125213
			t_4: 623.794914	Δv_4: 0.378360	R_{p5}: 7833005.169348
			t_5: 2481.150014	Δv_5: 0.000131	
			t_6: 2933.858016	Δv_6: 0.453515	
EP_DE	EVVEEJS	4.439864 km/s	t_0: −196.156429	Δv_0: 3.580373	R_{p1}: 10411.922288
			t_1: 175.857199	Δv_1: 0.015845	R_{p2}: 10600.022696
			t_2: 436.073458	Δv_2: 0.003641	R_{p3}: 13675.546437
			t_3: 58.762976	Δv_3: 0.001209	R_{p4}: 6798.849556
			t_4: 623.817716	Δv_4: 0.383698	R_{p5}: 7851748.989280
			t_5: 2481.402774	Δv_5: 0.002542	
			t_6: 2944.544453	Δv_6: 0.452555	

same function value as that announced in the database of ACT in ESA, though the specific solution is different from that. Also, the experimental result of GTOC1 also performs well, even if the accuracy of the calculation depends on the step size. For Cassini, if minimum energy is used as the optimization objective, the planets sequence EVVEJS in the benchmark may not be the best selection, while the function value 4.425189 km/s based on the planets sequence EVVEEJS found in this paper is better than 4.93 km/s which is announced by ACT.

It is a new research trend for heuristic algorithms based on machine learning. EP_DE algorithm combining search space exploring and PCA proposed in this paper is the basic research work for it. But, in general, there are also some limitations of EP_DE algorithm. For example, the processes of data statistics and analysis will cost lots of time and computing resources. However, this problem can also be solved by means of the distributed computing and development of hardware technology.

The purpose of the machine learning here is to obtain more information from less data. In the future research work, in order to improve the calculation accuracy of EP_DE algorithm, deep learning, decision tree, neural network, and reinforcement learning can also be added into the EP_DE algorithm.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by Natural Science Foundation of China under Grant nos. 41571403, 61472375, and 61103144.

References

[1] T. Vinkó, D. Izzo, and C. Bombardelli, “Benchmarking different global optimisation techniques for preliminary space trajectory design,” in *Proceedings of the 58th International Astronautical Congress*, pp. 24–28, 2007.

[2] M. Vasile, “A global approach to optimal space trajectory design,” *Advances in the Astronautical Sciences*, vol. 114, pp. 629–647, 2003.

[3] M. Vasile, J. M. R. Martin, L. Masi et al., “Incremental planning of multi-gravity assist trajectories,” *Acta Astronautica*, vol. 115, pp. 407–421, 2015.

[4] D. Izzo, V. M. Becerra, D. R. Myatt, S. J. Nasuto, and J. M. Bishop, “Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories,” *Journal of Global Optimization*, vol. 38, no. 2, pp. 283–296, 2007.

[5] D. Hennes and D. Izzo, “Interplanetary trajectory planning with Monte Carlo tree search,” in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI ’15)*, pp. 769–775, AAAI Press, Buenos Aires, Argentina, July 2015.

[6] D. Silver, A. Huang, C. J. Maddison et al., “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] J. W. Hartmann, V. L. Coverstone-Carroll, and S. N. Williams, “Optimal interplanetary spacecraft trajectories via a pareto

genetic algorithm," *Journal of the Astronautical Sciences*, vol. 46, no. 3, pp. 267–282, 1998.

[8] M. Vasile and M. Locatelli, "A hybrid multiagent approach for global trajectory optimization," *Journal of Global Optimization*, vol. 44, no. 4, pp. 461–479, 2009.

[9] D. Izzo, "Global optimization and space pruning for spacecraft trajectory design," *Spacecraft Trajectory Optimization*, vol. 1, pp. 178–200, 2010.

[10] R. E. Korf and P. Schultze, "Large-scale parallel breadth-first search," in *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI '05)*, vol. 3, pp. 1380–1385, AAAI Press, 2005.

[11] Y. Z. Zhu and T. Y. Cheung, "A new distributed breadth-first-search algorithm," *Information Processing Letters*, vol. 25, no. 5, pp. 329–333, 1987.

[12] B. Awerbuch and R. G. Gallager, "A new distributed algorithm to find breadth first search trees," *IEEE Transactions on Information Theory*, vol. 33, no. 3, pp. 315–322, 1987.

[13] M. Bisson, M. Bernaschi, and E. Mastrostefano, "Parallel distributed breadth first search on the Kepler architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 2091–2102, 2015.

[14] Y. Katsuta, T. Miyajima, S. Nomura et al., "Accelerating breadth-first search using Tightly Coupled Accelerator," *IEICE Technical Report Dependable Computing*, vol. 114, no. 21, pp. 63–68, 2014.

[15] G. Liu, A. N. Hong, L. I. Xiao-Qian et al., "Study on optimizing techniques of breadth-first search algorithm on graphic processing unit," *Journal of Chinese Computer Systems*, vol. 35, no. 5, pp. 1074–1079, 2014.

[16] G. Tan, *GAO. Global Search of Multiple Gravity Assist Transfer Trajectories for Deep Space Probes*, Spacecraft Engineering, 2012.