AAS 19-899

GTOC X: Results and Methods of National University of Defense Technology and Xi'an Satellite Control Center

Ya-Zhong Luo,[1*] Hong-Xin Shen,[1†] An-Yi Huang,[‡] Tian-Jiao Zhang,[§] Yue-He Zhu,[**] Zhao Li,[§] Peng Shu,[**] Zhen-Jiang Sun,[**] Jian-Hui Li,[§] Zhen-Yu Li,[††] Jian-Jun Shi,[††] Bing Yan,[††] Xiang-Nan Du,[††] Zhen Yang[‡‡]

This paper describes the methods used and the results obtained by team NUDT&XSCC (a collaboration team between National University of Defense Technology and Xi'an Satellite Control Center) for the 10th edition of the Global Trajectory Optimization Competition. The resulting trajectory won the 1st place in the competition, achieving a final mission value of J = 3101.15. The methods used by our team are described. These methods mainly include star-targeting technique, allowing one to flyby 2 to 3 stars using a single impulse or two impulses; targets layout optimization technique, enabling one to select the targets to get optimal spatial distribution based on the insights into the ideal configuration; ant colony optimization, permitting one to construct feasible settlement trees for the selected optimal targets.

I. INTRODUCTION

The 10th global trajectory optimization competition (GTOC X) problem is the settlement of the galaxy. This was accomplished by three mother ships and two fast ships flying around the galactic center. The goal was to settle as many of the one hundred thousand star systems as possible, in as uniform a spatial distribution as possible, while using as little propulsive velocity change as possible. A performance index, which depended on the number of targets, the distribution of the targets, the ΔV ratio, must be maximized subject to a variety of constraints. Propulsive maneuvers are impulsive, with ΔVs magnitude and number limits; the tour should last less than 90 Myr (million years) in a 10 Myr launch window. The detail mathematical formalization can be read in the work written by the competition organizers[1].

Since the problem proposed by JPL is quite complex, embedding a rationale in the problem is preferable to our team (Team7), so that an adequate mission (i.e., a high-score solution for the competition) can be devised without exploiting any "standard" global optimization technique. The complexities of this problem are to manage starting settlement targets set and select suitable

[1]These authors contributed equally to this work.
[*]Professor, National University of Defense Technology, 410073, Changsha, China. luoyz@nudt.edu.cn
[†]Assistant Professor, Xi'an Satellite Control Center, 710043, Xi'an, China. hongxin.shen@gmail.com.
[‡]Ph.D. Candidate, National University of Defense Technology; Aerospace Engineer, Xi'an Satellite Control Center.
[§]Aerospace Engineer, Xi'an Satellite Control Center.
[**]Ph.D. Candidate, National University of Defense Technology.
[††]Graduate Student, National University of Defense Technology.
[‡‡]Assistant Professor, National University of Defense Technology.

followed targets to rendezvous. The basic idea of team7 consists in the exploration of stars performed by keeping the settlement routine moving as close as a square spiral with respect to the galactic center. Besides, the targets are sought globally in the optimal spatial distribution before their rendezvous sequence is generated.

Trajectory design is conceptually split in four parts as follows. First, a number of multiple impulsive trajectories for mother ship are generated to provide databases of settlement pods by means of differential evolution (DE)[2]. Generally, it is not difficult to obtain two Pods release through one impulse of shooting, but only one/zero Pod is released in order to make the mother ship go further as fast as possible. Second, suitable rendezvous targets are selected in the reachable region of each Pod and fast ship using generic algorithm, and the rendezvous targets are connected to build up to 14 settlement trees using ant colony optimization (ACO)[3-4]. Actually, this step only concerns the ideal spatial distribution (or the error functions to be minimized) rather than the real performance index. Third, the settlement trees built by ACO is re-adjusted by replacing, deleting, and adding each node, as long as the performance index could be improved the most. Finally, the arrival time of each star in the resulting settlement trees are re-optimized using SNOPT[5] as well as DE as a preprocessing technique.

II. APPROXIMATE AND ACCURATE SOLVER FOR THE BVP

This problem introduces a totally new dynamics. The current conclusions on the motion of a ship or a star in general two-body systems are not available any more. Therefore, the initial and boundary value problems should be dealt with carefully.

An efficient boundary value problem solver is necessary for the whole solution design. In many cases, given transfer time of any pair of bodies, the corresponding velocity increments need to be rapidly estimated. For this purpose, a linear approximation is presented for the boundary value problem.

It is found that the orbital period of a star is usually over 100 Myr in this problem. While transfer trajectory usually lasts for only a few number of Myr, which is much shorter than the orbital period. That is to say, the transfer trajectory can be simply approximated as a straight line. Assume that the initial position and velocity vectors $\boldsymbol{r}_1$ and $\boldsymbol{v}_1$ of a ship at time t_1 are known, a ship is transferred to state $\boldsymbol{r}_2$ and $\boldsymbol{v}_2$ at time t_2, by means of two impulses $\Delta\boldsymbol{v}_1$ and $\Delta\boldsymbol{v}_2$. Suppose the transfer trajectory is a straight line, then the velocity of ship along the line should be constant, which equals

$$\boldsymbol{v}_t = \frac{\boldsymbol{r}_2 - \boldsymbol{r}_1}{t_2 - t_1} \tag{1}$$

The two impulses can be calculated as

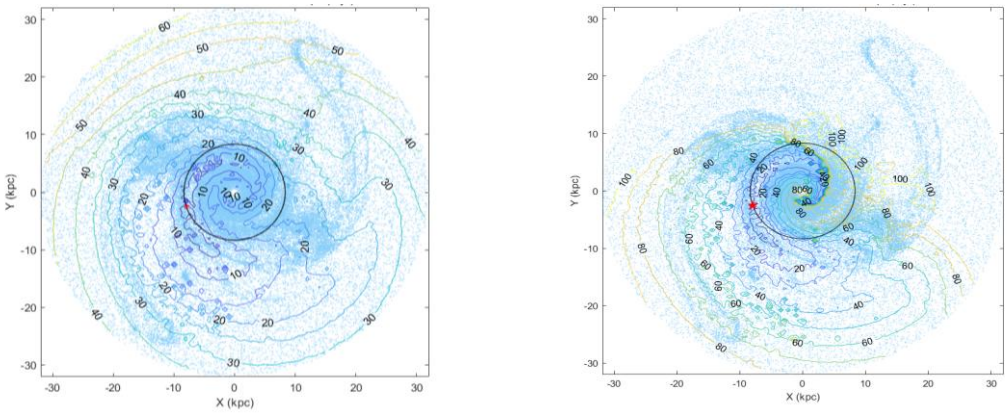
$$\begin{cases} \Delta\boldsymbol{v}_1 = \boldsymbol{v}_t - \boldsymbol{v}_1 \\ \Delta\boldsymbol{v}_2 = \boldsymbol{v}_2 - \boldsymbol{v}_t \end{cases} \tag{2}$$

The approximated solution based on linear model could provide good initial guess for accurate solving of BVP. The relative error between approximated solution and accurate solution is often around 10%, thus the approximate solution guarantees fast convergence of accurate optimization.

For accurate optimization, an integrator DE/STEP/INTRP described in the book of Shampine and Gordon [6] was used for propagating the dynamics. To solve the shooting function, Minpack-1[7], a package of FORTRAN subprograms for the numerical solution of systems of nonlinear equations and nonlinear least-squares problems, is used.

According to the limits, the Fast Ship's impulses is less than 1500 km/s. They could be applied in twice, each shouldn't be more than 1500 km/s. Then we calculated the shortest time contour to arrive those stars by Fast Ship, which is shown in Figure 1(a).

A Mother Ship can applies 3 impulses at most, each of which is less than 200 km/s. In consider of the Mother Ships should arrive targets as soon as possible, we suppose that the first two impulses are applied in the first two Myr after they leave from Sol. The Settlement Pod released from Mother Ships has an impulse no more than 300 km/s. In order to analyze the reachable regions, we simplified the first two impulses of the Mother Ship as one. Thus it was assumed to apply two impulses, the first one is less than 400 km/s and the second is less than 300 km/s. Then, with the same way of the Mother Ships, we get the shortest arrive time contour of every star, shown in Figure 1(b).



(a) Fast ship (b) Mother ship

Figure 1. The minimum arrival time of the fast ships and mother ships.

III. FLYBY TRAJECTORIES FOR THE MOTHERSHIPS

The flyby trajectories for the mother ships can be composed of many types because a total of three impulses is available. For convenience, a three-number code is used to represent the type of flyby trajectories. For example, "111" means that each impulse aims at a star and "022" denotes that the second and the third impulses each aim at two stars, respectively. Flying by a star using one impulse is easy because the flyby trajectory can be directly obtained by solving a two-point boundary-value problem. In the following part, we mainly introduce the approach to simultaneously fly by two stars using one impulse and the approach to simultaneously fly by three stars using two impulses.

A. Approach to Fly by Multiple Stars

The core idea of the approach to fly by multiple stars is summarized as "Shooting+Optimizing". Algorithm 1 presents the approach to simultaneously fly by two stars using one impulse, where step 1 to 5 are the procedure of "Shooting" and step 6 and 7 are that of "Optimizing". Figure 2 illustrates the flight trajectories of "Shooting" and "Optimizing". Essentially, "Shooting" is used to find two candidate flyby stars and "Optimizing" is used to transform a piece rendezvous trajectory to the corresponding flyby one by gradually reducing the middle impulse (Δv_3) to 0. Note that "Shooting" may be implemented many times before "Optimizing". The optimization model to obtain two-star flyby trajectory is presented in part B. Once the objective is converged to 0 and all the constraints are satisfied, a piece of feasible two-star flyby trajectory is achieved.

Algorithm 1 The flow to simultaneously fly by two stars using one impulse

1: Randomly given a sliding time ΔT_1, and propagate the mothership from (r_0, v_0) to (r_1, v_1)

2: Randomly given an impulse $\Delta \boldsymbol{V}_1$ and a transfer time ΔT_2, and propagate the mothership from (r_1, v_1) to (r_2, v_2)

3: Collect the stars that are able to cross the trajectory during ΔT_2, set $N = 0$

4: **for** $1 : M$ (number of the collected stars)
Calculate the minimal distance $d_{\min}$ between the mothership and the star
if $d_{\min} < 0.1\ rpc$
Put the star into the candidate pool, $N = N + 1$
end if
end for

5: **if** $N < 2$
Return to step **1**
else
Randomly select two stars from the candidate pool as the flyby stars
end if

6: Optimizing the rendezvous trajectory from (r_0, v_0) to Star 1 and Star 2 based on the two-star flyby optimization model

7: **if** $\Delta V_2 < 10^{-3}$ && all the constraints are satisfied
Successfully fly by two stars using one impulse
else
Return to step **1**
end if

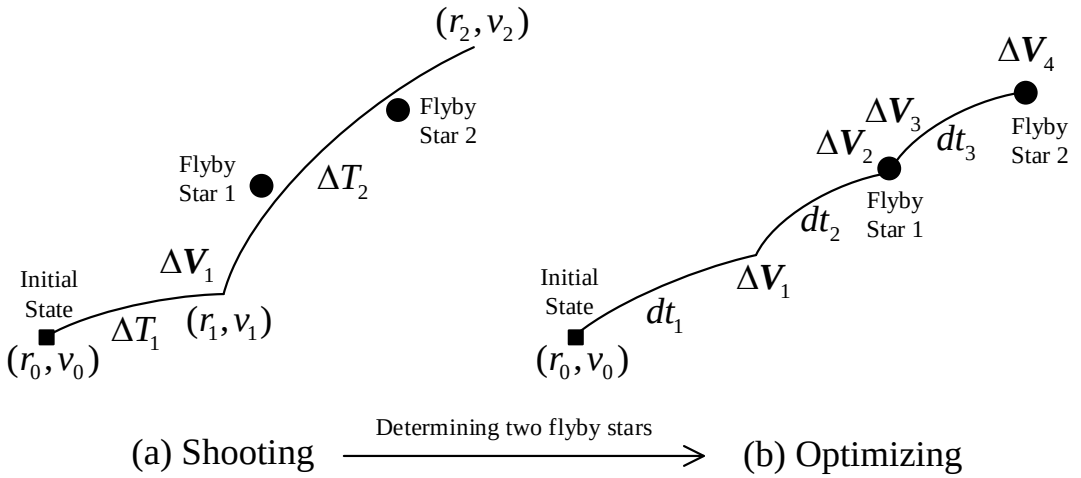


Figure 2. Approach to simultaneously fly by two stars using one impulse.

The approach to simultaneously fly by three stars using two impulses is similar to the one to fly by two stars. The flow and the flight trajectory are presented in Algorithm 2 and Figure 3, respectively. Two impulses are required in this case because using only one impulse cannot guarantee to simultaneously fly by three stars in our test. The optimization model to obtain three-star flyby trajectory is presented in part C. The three-star flyby trajectory can be feasible only when the middle impulses $\Delta \boldsymbol{V}_4$ and $\Delta \boldsymbol{V}_6$ both reduce to 0 with all the constraints satisfied.

Algorithm 2 The flow to simultaneously fly by three stars using two impulses

1: Randomly given a sliding time ΔT_1, and propagate the mothership from (r_0, v_0) to (r_1, v_1)

2: Randomly given an impulse $\Delta \boldsymbol{V}_1$ and a transfer time ΔT_2, and propagate the mothership from (r_1, v_1) to (r_2, v_2)

3: Randomly given an impulse $\Delta \boldsymbol{V}_2$ and a transfer time ΔT_3, and propagate the mothership from (r_2, v_2) to (r_3, v_3)

4: Collect the stars that are able to cross the trajectory during ΔT_3, set $N = 0$

5: **for** $1 : M$ (number of the collected stars)

Calculate the minimal distance $d_{\min}$ between the mothership and the star

if $d_{\min} < 0.1\ rpc$

Put the star into the candidate pool, $N = N + 1$

end if

end for

6: **if** $N < 3$

Return to step **1**

else

Randomly select three stars from the candidate pool as the flyby stars

end if

7: Optimizing the rendezvous trajectory from (r_0, v_0) to Star 1, Star 2 and Star 3 based on the three-star flyby optimization model

8: **if** $\Delta V_3 + \Delta V_4 < 10^{-3}$ && all the constraints are satisfied

Successfully fly by three stars using two impulses

else

Return to step **1**

end if

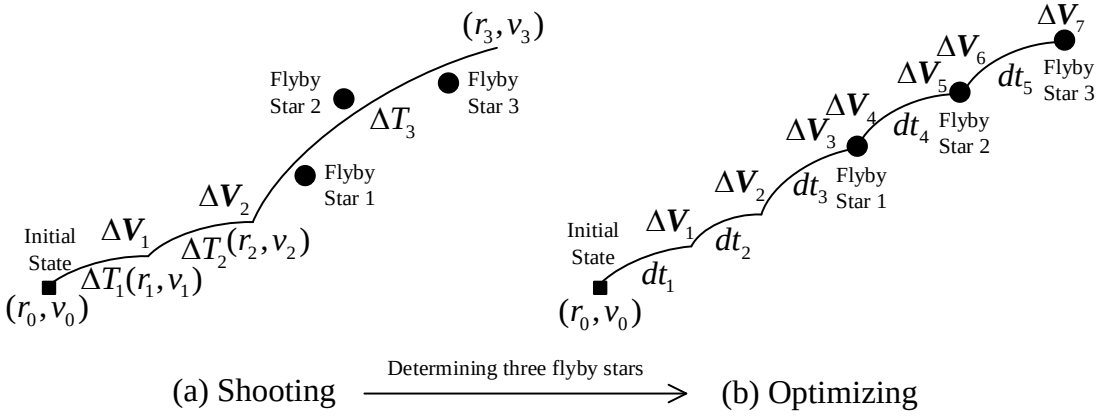


Figure 3. Approach to simultaneously fly by three stars using two impulses.

B. Two-Star Flyby Optimization Model

1) Design variables

There are three design variables in the two-star flyby optimization model $\boldsymbol{D} = [dt_1, dt_2, dt_3]$, where dt_1 is the sliding time, dt_2 is the transfer time from the first impulse to Star 1 and dt_3 is the transfer time from Star 1 to Star 2.

2) Objective function

The objective is to minimize the velocity increment of $\Delta \boldsymbol{V}_3$,

$$\min J = \left\| \Delta \boldsymbol{V}_3 \right\| \tag{3}$$

3) Constraints

The velocity increments of $\Delta \boldsymbol{V}_1$, $\Delta \boldsymbol{V}_2$ and $\Delta \boldsymbol{V}_4$ should be no larger than their maximal values

$$\begin{cases} \left\| \Delta \boldsymbol{V}_1 \right\| \le 200 \text{ km/s}, \\ \left\| \Delta \boldsymbol{V}_2 \right\| \le 300 \text{ km/s}, \\ \left\| \Delta \boldsymbol{V}_4 \right\| \le 300 \text{ km/s} \end{cases} \tag{4}$$

C. Three-Star Flyby Optimization Model

1) Design variables

There are eight design variables in the three-star flyby optimization model

$$\boldsymbol{D} = [dt_1, dt_2, dt_3, dt_4, dt_5, \Delta V_{1x}, \Delta V_{1y}, \Delta V_{1z}] \tag{5}$$

where dt_1 is the sliding time, dt_2 is the transfer time from the first impulse to the second impulse, dt_3 is the transfer time from the second impulse to Star 1, dt_4 is the transfer time from Star 1 to Star 2 and dt_5 the transfer time from Star 2 to Star 3. $\Delta V_{1x}, \Delta V_{1y}, \Delta V_{1z}$ are three components of $\Delta \boldsymbol{V}_1$.

2) Objective function

The objective is to minimize the total velocity increment of $\Delta \boldsymbol{V}_4$ and $\Delta \boldsymbol{V}_6$

$$\min J = \left\| \Delta \boldsymbol{V}_4 \right\| + \left\| \Delta \boldsymbol{V}_6 \right\| \tag{6}$$

3) Constraints

The velocity increments of $\Delta \boldsymbol{V}_1$, $\Delta \boldsymbol{V}_2$, $\Delta \boldsymbol{V}_3$, $\Delta \boldsymbol{V}_5$ and $\Delta \boldsymbol{V}_7$ should be no larger than their maximal values

$$\begin{cases} \left\| \Delta \boldsymbol{V}_1 \right\| \le 200 \text{ km/s}, \\ \left\| \Delta \boldsymbol{V}_2 \right\| \le 200 \text{ km/s}, \\ \left\| \Delta \boldsymbol{V}_3 \right\| \le 300 \text{ km/s}, \\ \left\| \Delta \boldsymbol{V}_5 \right\| \le 300 \text{ km/s}, \\ \left\| \Delta \boldsymbol{V}_7 \right\| \le 300 \text{ km/s} \end{cases} \tag{7}$$

D. Database Generation

Based on the two-star flyby and three-star flyby approaches introduced above, many types of flyby trajectories can be generated. However, these approaches can only achieve partial flyby trajectory of a mother ship. Running the approach multiple times is necessary to achieve a complete piece of flyby trajectory. For example, we need to repeat the procedure of flying by two stars three times to achieve a piece of "222" flyby trajectory and implement both the procedures of flying by two stars and that of flying by three stars to achieve a piece of "032" flyby trajectory. The optimization for achieving each partial flyby trajectory is implemented by an improved DE algorithm. Parallel computing is applied for generating the database. Based on "Tianhe" supercomputer (about 4000 available cores), millions of pieces of flyby trajectories were achieved during the competition, including "222", "032", "122", "022" and "012" flyby trajectories.

Note that the first impulse in "022" and "012" flyby trajectories does not aim at any star, but only to combine with the second impulse to provide larger escaping velocity for the mother ship. The time interval between the first and second impulses is set to 1 million year constantly. "022" and "012" flyby trajectories are considered because the mother ship is expected to fly by higher-

orbit stars (i.e. the star with bigger cycling radius) with earlier arriving time, thereby saving more time to settle higher orbit stars.

IV. CONFIGURATION OPTIMIZATION

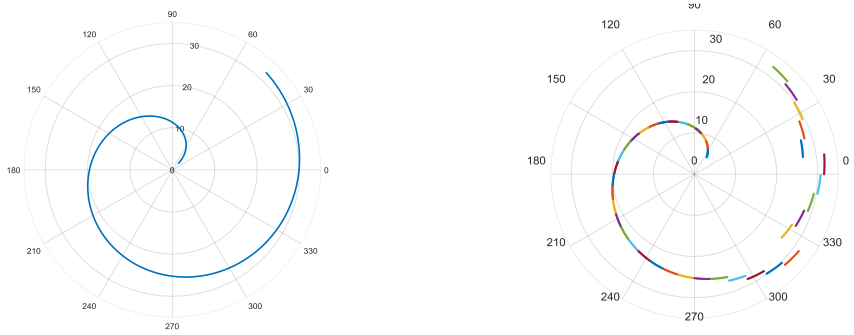
At the early stage of the competition, we constrained the range of settle ships to follow a special spiral line. This method helped us to get a score over 400 with only six starting ships. Then we tried to find the ideal configuration, described by formula $\theta = f^*(r)$. From the merit function, it's clear that both the axial and tangential distribution of settled stars should be uniform. In addition, density of stars in each ring [r, r+dr] is equal to density in each sector of dθ. Thus we can find a formula as follows:

$$\rho\pi\left((r+\mathrm{d}r)^2 - r^2\right) = \rho\frac{\mathrm{d}\theta}{2\pi}\pi\left(r_{\max}^2 - r_{\min}^2\right) \tag{8}$$

which results a formula as

$$\theta = \frac{2\pi}{(r_{\max}^2 - r_{\min}^2)}r^2 + c = f^*(r) \tag{9}$$

where $r_{\max}$=32 kpc and $r_{\min}$=2 kpc. Its shape is shown in Figure 4(a). Actually the line can be divided into several fragments and the order can be rearranged, as shown in Figure 4(b).



(a) Monotonically increasing (b) Rearranged

Figure 4. The ideal configuration based on the spiral assumption.

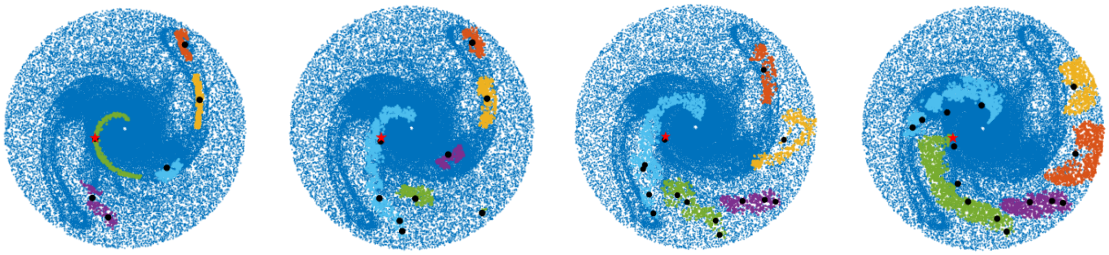


Figure 5. Evolution of the configurations.

Then we select suitable mother ships and fast ships to let the resulting trajectory approach $\theta = f^*(r)$. The evolution of initial layout is shown by red spots in Figure 5.

To design a well-done settle strategy from initial stars which can make J optimal is almost impossible. Firstly, we only consider a part of the real J, that is $J^* = \frac{N_{rem}}{1+10^{-4}N_{rem}(E_r + E_\theta)}$. The aim

of optimization is to select a subset to make the merit function J^* maximum, where N_{rem} is the count of selected stars. We raised a reverse method to generate the optimal settle trees. Firstly all the stars that can be rendezvoused with from initial stars are calculated. Then a global optimization algorithm is used to select stars which make *J* optimal. Finally the settle trees are generated using selected stars as many as possible using the method in Section VI.

As the starting stars are specified, the allowed rest time of each star is also known. Then we can filter all the stars that can be transferred to during the rest time by using impulses less than the maximum total Δv. Assume the count of achievable stars is *N*, then the dimension of encoding is *N*. Boolean variables are used to express the select status of each star. 0 indicates that the star is filtered and 1 indicates that the star is selected. The flow of genetic algorithm is presented in Figure 6. The comparison between preliminary targets and optimal targets is shown in Figure 7.

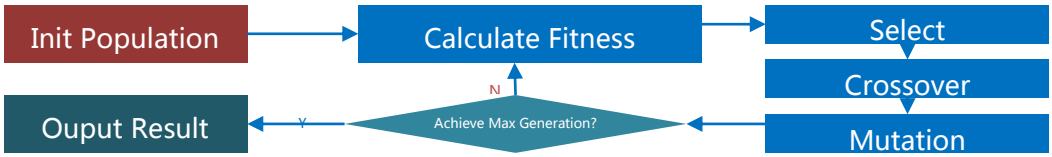


Figure 6. The flow of GA.

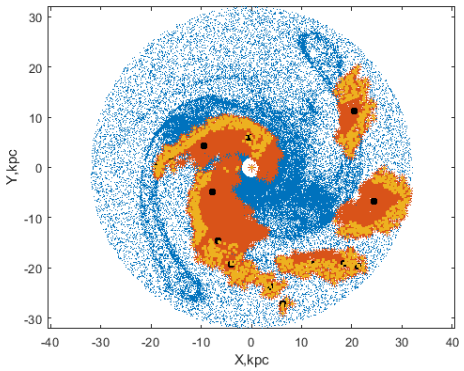


Figure 7. Stars before and after GA selection. Orange points and yellow points indicate preliminary targets and selected targets, respectively.

VI. SETTLEMENT TREE CONSTRUCTION

After a clustering of stars that together contribute a uniform spatial distribution have been selected, the settlement trees should be constructed. The problem of constructing the settlement trees aims to connect a set of chosen stars to a root star (i.e., the star that is settled by the initial settlers including both Mother Ships and Fast Ships) through a tree network for each star cluster, subject to constraints on the propulsive maneuvers and the capacity of Settler Ships.

The minimum spanning tree problem (MST)[8] is one of the widely studied combinatorial optimization problems. It aims at finding a spanning tree of the given nodes such that the sum of the weights attached to edges is minimal. When the domain constraints are ignored, the settlement tree construction problem is closely analogous to the MST. More specifically, the chosen stars can be viewed as nodes, and the directed edge connecting two stars corresponds to the settlement trajectory between them which is characterized by both the ΔV budgets consumed and the transfer duration. However, the three main differences between these two problems are as follows: 1) Unlike nodes in MST, stars are moving and the ΔV costs for settling a star thus depend on the settlement schedule (i.e., the exact epochs of rendezvous and departure from each star), 2) The

propulsive maneuver limits, as well as the maneuver timing limit, can be converted into the capacity constraints on edges, and 3) The capacity of Settler Ships adds to the degree constraint on each node. Although there are serval practically relevant variants of the MST problem, to the best of our knowledge, our problem does not belong to any existing variant of the MST. Owing to these specialized constraints, we named this new variant capacitated MST with degree-constrained problem (CMSTDC). Since the degree-constrained MST[9] problem has been shown to be NP-complete and therefore computationally expensive to solve in exact ways, the CMSTDC problem is clearly also NP-complete as it can be reduced to a degree-constrained MST when settlers with unlimited fuel budgets are allowed.

Due to the hardness of the CMSTDC problem, it makes heuristic approaches more suitable. To deal with it, ant colony optimization (ACO) based metaheuristic is proposed to obtain a satisfactory solution within reasonable time in this paper. ACO is a swarm intelligence optimization algorithm which has been successfully applied mainly to different routing problems[10]. Convergence proofs for generalized versions of ACO algorithms can be found in Ref. 11.

A. Formulation of the CMSTDC problem

Assuming that the star node set is partitioned into L disjoint nonempty subsets, each of which has one and only one root star. The CMSTDC problem seeks to identify the shortest capacitated spanning tree with degree-constrained for each subset. For the sake of simplicity, only the CMSTDC for a single subset is described in this paper. Formally, the CMSTDC can be defined on a directed graph $G = <V, E>$, where V is a set of nodes comprised of a unique root node 0 and a subset $V/\{0\}$ of n star nodes. E is a set of edges connecting each pair of stars in V. If a fuel transfer between two stars is permissible without violating both maneuver and timing limits, an edge between the corresponding star nodes exists. Thus, the directed edge $(i, j) \in E$ corresponds the orbital maneuver from star $i \in V$ to $j \in V$, associated with which is the required minimum transfer time $\Delta t_{ij}^*(t_i) = \underset{\Delta t_{ij}(t_i)}{\arg\min} \left|\Delta v_{ij}(t_i) - \Delta V_{\max}\right|, \Delta t_{ij}^*(t_i) \in (0, 90)$ Myr, where t_i is the exact epoch of rendezvous from the star i, and $\Delta v_{ij}(t_i)$ denotes the velocity change incurred during the transfer at time t_i. Additionally, up to three Settler Ships can depart from each settled star, as long as at least 2 Myr have been elapsed since the star became settled. Each Settler Ship can rendezvous with a single star, and each star can only be settled once. The objective is to determine how the settlement tree is constructed to settle as many stars as possible while minimizing the average rendezvous epoch of all leaf stars (i.e., the settled star that does not release any Settler Ship).

The decision variable $x_{ij} = 1$ indicates that the edge proceeding from star node i to j is part of the CMSTDC. Clearly, the set of settled stars is $V' = \{v \mid \sum_{\forall i \in V} x_{iv} = 1, v \in V\}$, and the set of leaf nodes can thus be defined as $U = \{u \mid \sum_{\forall i \in V} x_{ui} = 0, u \in V'\}$. For the sake of clarity, the CMSTDC problem can be formulated by the following integer linear programming model:

$$\min \quad 10 * \left|V / V'\right| + \left(\sum_{i \in V} \sum_{j \in V} \Delta t_{ij}^*(t_i) \cdot x_{ij} \right) \Big/ \left(10 * \left|U\right|\right) \tag{10}$$

subject to

$$\text{s.t.} \quad \sum_{i \in V} x_{i0} = 0 \tag{11}$$

$$\sum_{i \in V} x_{ij} = 1, \forall j \in V' \tag{12}$$

$$\sum_{i \in V} x_{ji} \le 3, \forall j \in V \tag{13}$$

$$0 \le t_j \le 90, \forall i \in V' \tag{14}$$

$$\Delta t_{ij}^{*}(t_i) = \left(t_j - t_i\right) \cdot x_{ij} \ge 2x_{ij}, \forall i, j \in V \tag{15}$$

$$x_{ij} \in \{0,1\}, \forall i, j \in V \tag{16}$$

The objective (10) is to settle as many chosen stars as possible, while minimizing the average rendezvous epoch of all leaf stars in CMSTDC. According to the timing constraint that all settlement events must be made up to 90 Myr past Year Zero, the second term in cost function is less than the penalty of losing one star in the settlement tree, that is the first term in (10). Clearly, the objective shows the preference on the number of settled stars. Constraints (11) and (12) ensure that there is only one root star and each star can be settled once by exactly one Settler Ship. Constraint (13) means that the out-degree of each star node is less than or equal to 3. Maneuver timing constraints is given in (14). Constraint (15) guarantees the time gap between two successive settlements is greater than 2 Myr. Eventually, constraint (16) imposes the restriction on the decision variables.

B. Ant Colony Optimization

ACO represents an optimization problem by a construction graph and uses *K* artificial ants to walk on the graph where *K* is the size of ant colony. Each ant constructs a solution iteratively and its behavior is guided by pheromone and heuristic information. The construction graph for our problem is the settlement mission graph defined above, in which the nodes are chosen stars and the edges are solution components. The proposed ACO procedure is summarized in Algorithm 3. The main iterative procedure of ACO consists of three steps. At the first step, each ant builds a solution according to the transition rule. Subsequently a local search procedure is employed to improve every constructed solution. At the last step, pheromone is updated. The iterative process terminates when a stopping criterion is satisfied. In the following, the main components of the proposed ACO are detailed.

Each ant constructs a CMSTDC in the same manner: starting from the empty partial solution $T = \varnothing$, the settlement tree is extended incrementally layer by layer. By means of selecting the stars to be settled in the next layer from the available candidates $Can(T) \subset V$ which is defined as the set of stars can be settled without violating any domain constraints. Consequently, the edge connecting the chosen star node $i \in Can(T)$ and its father $f(i) \in V$ is thus added to the partial solution. As in ACO algorithms the main working principle is to utilize the search memory, let us briefly describe how this memory is implemented. We use a square matrix with $n+1$ columns and rows. Each entry stores the pheromone intensity on each edge as an indicator of the usefulness of connecting these two nodes in previous iterations, based on the evaluation of the solutions found. In detail, an ant makes its decision on which star node to be settled in the next layer according to the following biased probabilistic rule:

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}^{a} \cdot \eta_{ij}^{b}}{\sum_{l \in Can(T)} \tau_{il}^{a} \cdot \eta_{il}^{b}}, & \text{if the star } l \in Can(T) \text{ and } t_l \le 90 \text{ Myr} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

The probability is defined by two factors, the pheromone τ_{ij} of edge (i, j), and the corresponding heuristic information η_{ij} that is defined as $\eta_{ij} = \frac{1}{\Delta t_{ij}^{*}(t_i)}$. In addition, *a* and *b* are the parameters that control their relative importance.

Once all ants have gone through solution construction, the pheromone update procedure is applied to these pheromone values. The update strategy is a pure elitist strategy, where only for edges belonging to the best-so-far CMSTDC reinforcement takes place. Moreover, all pheromone trails are decreased uniformly through pheromone evaporation as to allow ants forgetting bad solutions. The ACO iterative process terminates when a stopping criterion is satisfied.

Algorithm 3 ACO for the CMSTDC problem

Set parameters, initialize pheromone trails
while The stopping criterion is not met **do**
  for all ants *k*=1,…,*K* **do**
    $T_k = \varnothing, Can(T_k) = V / \{0\}$
    set the layer indicator *l*=0, the set of nodes in the *l*-th layer is defined as L_l, $L_0 = \{0\}$, $L_l = \varnothing, l = 1, ..., 10$
    for each layer $l = 1, ..., 10$ **do**
      for each star $i \in L_l$ **do**
        while $Can(T_k) \neq \varnothing$ **do**
          for $c = 1, 2, 3$ **do** //the maximum number of Settler Ships can depart from each settled star
            if rand (0,1) < 0.1 **do** //decide how many Settler Ships does the settled star *i* release?
              continue;
            else
              choose a star $j \in Can(T_k)$ with probability p_{ij}
              add the edge (i, j) to the partial solution T_k
              $Can(T_k) = Can(T_k) / \{j\}$, $L_{l+1} = L_{l+1} \bigcup \{j\}$
            end if
          end for
        end while
      end for
    end for
  end for
  for all ants *k*=1,…,*K* **do**
    evaluate the soluton T_k
    update the best-so-far soluton T^*
  end for
  update the pheromone on all edges of the best-so-far CMSTDC solution T^*
end while
return T^*

C. Settlement Tree Re-Adjustment

Actually, many stars near the settle tree aren't settled, and they can be used to improve score by some operations on the settled stars nearby. Therefore, we consider a local forward search based on the spanning tree obtained by ACO. We tried the algorithms of reconstructing, adding points, deleting points, and changing points. Each step is aimed at maximizing the real performance index. The operations are as follows:

1) Reconstruct nodes: Delete the last 2 nodes and re-settle using the depth-first search algorithm. The transfer time of the first leg takes the minimum transfer time, and the transfer time of the other leg reaches 90Myr;

2) Add nodes: If there are less than 3 children nodes in the tree or the arrival time is less than 88Myr, try to immigrate all the stars that can be settled, and choose the star that makes the increase of J the most;

3) Delete nodes: try to delete the leaf node if the score increases;

4) Replace nodes: Try to replace the selected star by a nearby unsettled star, calculate the scores of the settle tree after replacement; if the score after replace increased, accept the replacement. Otherwise, try another unsettled star nearby.

The flow are presented in Algorithms 4~7.

Algorithm 4 Reconstruction nodes

while within computational budget **do**
  select one node *n* from $S_{settled}$ at random
  if number of child nodes of *n* > 0 and number of child nodes of each child node of *n* == 0
    best ← $J\left(S_{settled}\right)$ and *addList* ← null and *deleteList* ← null
    father ← father of *n*
    deleteList ← child nodes of *n* and *n*
    delete child nodes of *n* and *n*
    $S_{settled}$ ← $Can(T_{arrival})$ of *father*
    for each star $c \in S_{candi}$ **do**
      C_{candi} ← $Can(T_{arrival})$ of *c*
      for *loop*=1:1000 **do**
        select *trial*(3,2,1,or 0 stars) from C_{candi} at random
        if $J\left(S_{settled} - deleteList + trail\right)$ > best
          best ← $J\left(S_{settled} - deleteList + trail\right)$
          addList ← *trial*
        end if
      end for
    end for
    if length of *addList* > 0
      $S_{settled} \leftarrow S_{settled} - deleteList + addList$
    end if
  end if
end while

Algorithm 5 Adding nodes

while within computational budget **do**
  select one node *n* from $S_{settled}$ at random
  if number of child nodes of *n* < 3 and arrival time of *n* < 88Myr
    best ← $J\left(S_{settled}\right)$ and *select* ← null
    for each star $s \in Can(T_{arrival})$ **do**

6: **if** $J\left(S_{settled}+s\right)>$ best
7: $best \leftarrow J\left(S_{settled}+s\right)$
8: $select \leftarrow s$
9: **end if**
10: **end for**
11: **if** *select* is not null
12: $S_{settled} \leftarrow S_{settled}+select$
13: **end if**
14: **end if**
15: **end while**

Algorithm 6 Deleting nodes

1: $best \leftarrow J\left(S_{settled}\right)$
2: **for** each star $s \in S_{settled}$ **do**
3: **if** $J\left(S_{settled}-s\right)>$ best
4: $best \leftarrow J\left(S_{settled}-s\right)$
5: $S_{settled} \leftarrow S_{settled}-s$
6: **end if**
7: **end for**

Algorithm 7 Replacing nodes

1: **for** each star $s \in S_{settled}$ **do**
2: $best \leftarrow J\left(S_{settled}\right)$ and $select \leftarrow$ null
3: $S_{candi} \leftarrow$ if transfer time no change and satisfy constraints of velocity increments
4: **for** each star $c \in S_{candi}$ **do**
5: **if** $J\left(S_{settled}-s+c\right)>$ best
6: $best \leftarrow J\left(S_{settled}-s+c\right)$
7: $select \leftarrow c$
8: **end if**
9: **end for**
10: **if** *select* is not null
11: $S_{settled} \leftarrow S_{settled}-s+select$
12: **end if**
13: **end for**

VII. FINAL OPTIMIZATION

The settlement trees achieved by the above processes are not as perfect as enough because the departure and arrival times, as well as the velocity increments of each transfer are not optimal. Further optimization including the overall optimization for determining the best departure and arrival times and the body-to-body optimization for calculating the optimal velocity increments of each transfer are required.

Actually, body-to-body optimization should be contained in the overall optimization if we want to achieve the global optimal total velocity increments. However, it is too time consuming and computationally difficult to determine the best departure and arrival times while calculating the optimal velocity increments of each transfer when there are tens or even hundreds of stars in a tree. In order to decrease the optimization difficulty, body-to-body optimization is implemented after overall optimization is completed, and the velocity increments of each transfer are estimated according to the corresponding two-impulse transfer during overall optimization. The overall optimization model is presented in the following part. Note that the overall optimization is implemented tree by tree. More than three impulses will be used if the maximum impulse constraint is violated. First, the improved DE algorithm[2] is applied for both the overall and body-to-body optimization, the design variables, objective function and constraints are as follows

1) Design variables

Once a settlement tree is given, the arriving time of each star is expressed as $T_0 = T_{pod}$ and

$$T_{ij} = \begin{cases} 90 & \text{if the current star is an end-node star} \\ T_{ij}^M + 2 + \delta_{ij} \cdot [90 - (T_{ij}^M + 2)] & \text{otherwise} \end{cases} \tag{18}$$

where T_{pod} is the time that the mothership releases the pod; T_0 is the arriving time of the root star; T_{ij} is the arriving time of S_{ij} (the *j*th star in the *i*th level); T_{ij}^M is the arriving time of S_{ij}'s mother star; δ_{ij} is the coefficient that determines the transfer time from the mother star to the current star.

δ are exactly the design variables, where all of the values are within [0, 1]. There are $N - N_{end} - 1$ design variables in overall optimization model, where N is the number of the total settled stars and N_{end} is the number of the total end-node stars.

2) Objective function

The objective is to minimize the total velocity increments of the whole tree.

$$\min J = \sum_{i=1}^{N-1} \left\| \Delta \boldsymbol{V}_{i1} \right\| + \left\| \Delta \boldsymbol{V}_{i2} \right\| \tag{19}$$

3) Constraints

The velocity increments of each transfer must be no larger than the maximal value

$$\left\| \Delta \boldsymbol{V}_{i1} \right\| + \left\| \Delta \boldsymbol{V}_{i2} \right\| \le 350 \text{ km/s} \quad i = 1, 2, ..., N-1 \tag{20}$$

Although DE tends to provide a global optimum, it is found that is generally slow to converge; because there are too many variables to deal with for DE. So we also improve the solution obtained by DE by re-optimizing of transfer times with SNOPT, and smooth converge can be obtained.

VIII. RESULTS

Figure 8 provides a summary of all the complete missions sets submitted by NUDT&XSCC during the competition, including the team's winning final submission of 3101.15. In general, score increases along with the climbing of the number of settled stars (see Figure 9) and the decrease of the error function (see Figure 10), and flattens near about 2 of fuel coefficient (see Figure 11). It is interesting noting that the score increases obviously faster than the certain decline due to the fading of the bonus function.

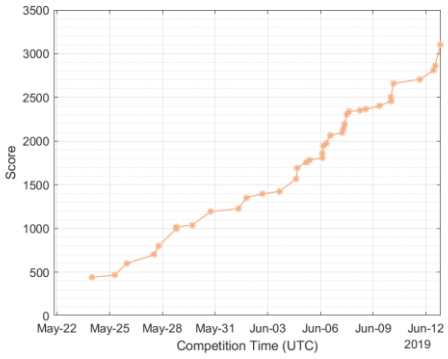


Figure 8. Evolution of score during competition.

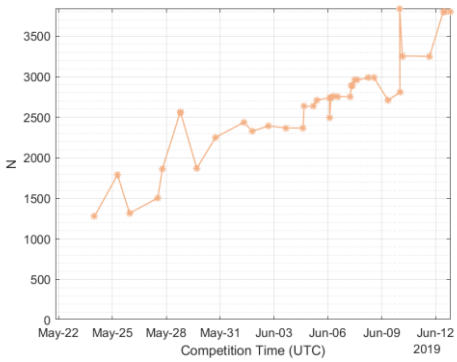


Figure 9. Evolution of the number of settled stars.

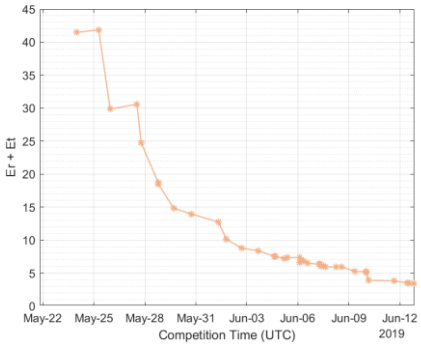


Figure 10. Evolution of the error function.

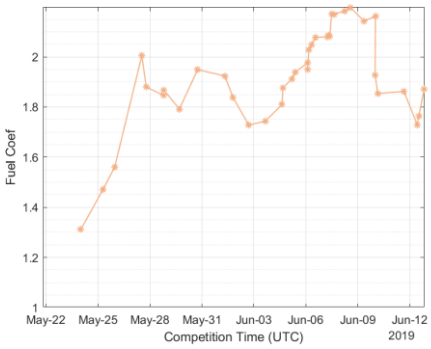


Figure 11. Evolution of the fuel coefficient.

At the early stage of the competition, we increased the fuel coefficient significantly by optimizing the maneuver time. But the rapid decline of time bonus counteract some scores. Then, with the better understanding of the error function, effective measures were taken to ensure the continuous reduction of Er and Etheta. At the last days, the number of settled stars are increased by regenerating the settlement tree (described in Section VI.C). Through the combined use of various means, our score finally reached J=3101.15 (primary objective), and 3798 different stars (secondary objective).

The stars arrived by the mother ships in the final solution is shown in Table 1, where only one/zero Pod is released in order to make the mother ship go further as fast as possible. The settlement trees depth generally grows up to 7, as shown in Figure 12; which also tells the complexity of this GTOC X problem due to the enormous tree size. In particular, it seems that the search in breadth is much more complex than the search in depth. A latest research of the authors using Monte Carlo tree search algorithm was conducted in the context of GTOC X suggesting that its use may be competitive to the current state of the art technique.

The total ΔV used performed by each ship and pod is 812701km/s (the second secondary objective). The final distribution of the reached stars is shown in Figure 13a. The error functions Er and Etheta, which represent how uniform of the spatial distribution in the galaxy, are 2.4 and 1.1 (see Figure 13b, Figure 13c)), respectively.

It is also given an improved solution found by NUDT&XSCC, completed shortly after the competition ended, scoring 3139. The main focus is to cut down Er in 3101 solution particularly in the highest region. Figure 14(a) provides the distributions of the solutions. The different color stars are explored by different Mother Ships and Fast Ships. It's obvious that the yellow stars are changed. Figure 14(a) and (b) show the magnitude of Etheta in each angle range. Etheta in 22.5 ° range becomes worse, but Er is decreased dramatically, i.e., the maximum Er value is below 0.2. Thus, the total score is better in the post-competition solution.

Table 1. Settlement stars arrived by three Mother Ships.

Mother Ship	Star id	Time (Myr)	R (kpc)	θ_f (deg)	Flyby velocity (km/s)
1	2097	21.935	6.036	93.712	269.555
	4478	38.223	10.416	156.137	299.159
	4855	58.294	16.266	172.100	290.947
	36794	68.013	18.636	179.303	276.667
2	10672	4.588	9.177	-148.361	201.937
	40977	29.285	16.029	-114.667	298.950
	99421	42.234	19.621	-101.625	276.512
	13002	58.086	23.994	-80.995	280.229
	12384	71.813	27.826	-76.972	273.760
3	23374	41.979	22.983	-57.319	300.000
	21650	56.789	26.457	-46.121	239.596
	62350	68.670	29.002	-42.711	232.424

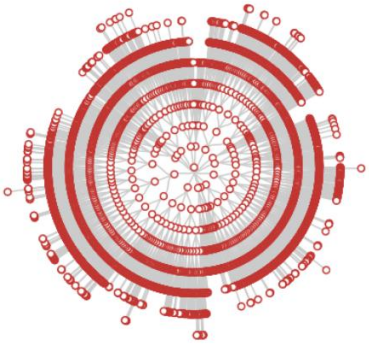
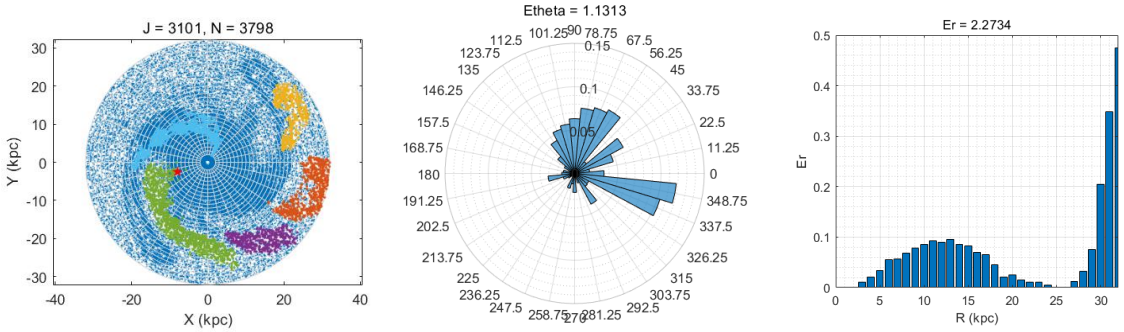
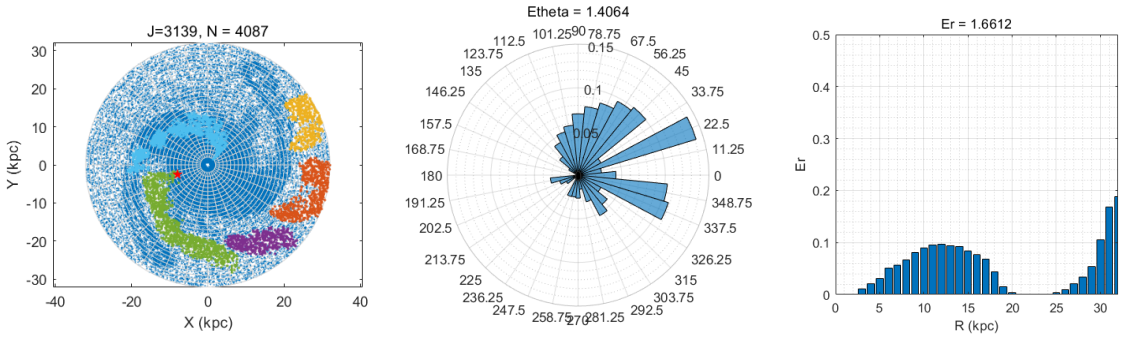


Figure 12. The settlement tress for solution 3101. The central point stands for the Sol.



(a) Targets distribution (b) Etheta (c) Er

Figure 13. Spatial distribution for solution 3101.



(d) Targets distribution (e) Etheta (f) Er

Figure 14. Spatial distribution for solution 3139.

VIII. CONCLUSION

The galaxy stars rendezvous problem posed for this edition of the GTOC was a challenging time-dependent combination problem. Especially, the tree-shape transfer routine was the first time introduced into space trajectory optimization problem. The expansion of settlement trees makes the search space become extremely huge. The reduction of solving difficulty was benefited greatly from the insights into ideal configuration; actually, the winner solution readily yields the assumed configuration. Also, we constructed the settlement trees after selecting the optimal targets. Thus the time-dependent combination problem can be solved using the existing state-of-the-art techniques, i.e., generic algorithm and ant-colony optimization-based approaches. However, the performance must be discounted by going around the time-dependent problem, so it is confidently expected that there is still considerable improved room for the current best solution.

ACKNOWLEDGMENTS

We thank JPL for posing this fascinating and challenging problem. This research was supported by the National Natural Science Foundation of China (No. 11702330).

REFERENCES

[1] Anastassios E. Petropoulos, Eric D. Gustafson, Gregory J. Whiffen, and Brian D. Anderson, "GTOC X: Settlers of the Galaxy Problem Description and Summary of the Results", Paper AAS 19-891, Astrodynamics Specialist Conference, Portland, Maine, 11-15 Aug. 2019.

[2] Y. Zhu, Y. Luo and J. Zhang. "Packing Programming of Space Station Spacewalk Based on Bin Packing Theory and Differential Evolution Algorithm," in Proc. IEEE World Congr. Evol. Comput., Vancouver, BC, Canada, Jul. 2016.

[3] H.-X. Shen, T.-J. Zhang, A.-Y. Huang, and Z. Li, "GTOC9: Results from the Xi'an Satellite Control Center (team XSCC)." Acta Futura. Vol. 11, 2018, pp. 49–55.

[4] H.-X. Shen, T.-J. Zhang, L. Casalino, et al. "Optimization of active debris removal missions with multiple target," Journal of Spacecraft and Rockets, Vol. 55, No. 1, 2017, pp. 181-189.

[5] P. E. Gill, W. Murray, and M. A. Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization," SIAM Journal on Optimization, Vol. 12, 2002, pp. 979-1006.

[6] L. F. Shampine, M.K. Gordon. Computer solution of ordinary differential equations: the initial value problem. 1975.

[7] https://github.com/devernay/cminpack

[8] Graham R L, Hell P. "On the history of the minimum spanning tree problem," Annals of the History of Computing," Vol. 7, No. 1, 1985, pp. 43-57.

[9] Narula S C, Ho C A. "Degree-constrained minimum spanning tree," Computers & Operations Research, Vol. 7, No. 4, 1980, pp. 239-249.

[10] Neumann F, Witt C. "Ant colony optimization and the minimum spanning tree problem," Theoretical Computer Science, Vol.411, No.25, 2010, pp. 2406-2413.

[11] Dorigo M, Stutzle T. "A short convergence proof for a class of ACO algorithms," IEEE Transactions on Evolutionary Computation, 2002, 6.