AAS 19-897

GTOC-X: OUR PLAN TO SETTLE THE GALAXY (ESA-ACT)

Dario Izzo[*], Marcus Märtens[†], Ekin Öztürk[‡], Mate Kisantal[§], Konstantinos Konstantinidis[¶], Luís F. Simões[‖], Chit Hong Yam[**], Javier Hernando-Ayuso[††]

The 10th edition of the Global Trajectory Optimization Competition (GTOC-X) invited participants all over the world to compete against each other to design efficient missions with the goal to settle our galaxy. Leveraging concepts of interstellar space travel like generational ships, the participants were tasked to develop settlement plans as each newly settled star system could spawn new settlement ships. This work presents the solution strategies developed by the ESA-ACT during the month long competition. In particular, we unfold the mathematical structure of the objective function that demanded a uniform spread throughout the galaxy and discuss its implications. We describe a tree search that is able to grow settlement trees concurrently over time starting from multiple initial points. Furthermore, we deploy several techniques to rearrange already established settlement trees in order to reduce the overall propulsive velocity change required.

INTRODUCTION

In about ten thousand years from the present, humanity will reset its counting of years to zero. Year Zero will be the year when humanity decides the time is ripe for the human race to boldly venture into the galaxy and settle other star systems.[1]

This visionary text introduced the problem released by the Jet Propulsion Laboratory for the tenth edition of the Global Trajectory Optimization Competition (GTOC-X). The problem, described in full details in[1] and also dubbed "Settlers of the Galaxy", has a highly complex and unusual mathematical structure containing elements of dynamics, optimization, graph theory, and trajectory design. The dynamics is that of motion in a central force field designed to approximate observed galactic rotation curves and thus is far from the familiar and more comfortable (even when perturbed) Keplerian dynamics. The merit function of the optimization element involves, rather creatively, to settle stars so that their spatial distribution matches some assigned probability distribution, making the fitness landscape also rather unfamiliar. The choice of interstellar vessels, and in particular the division between mother, fast and settler ships, as well as the possibility to launch 3 spaceships

[*]Scientific Coordinator, Advanced Concepts Team, European Space Agency, Keplerlaan 1, 2201 AZ, Noordwijk, NL.
[†]Research Fellow, Advanced Concepts Team, European Space Agency, Keplerlaan 1, 2201 AZ, Noordwijk, NL.
[‡]Young Graduate Trainee, Advanced Concepts Team, European Space Agency, Keplerlaan 1, 2201 AZ, Noordwijk, NL.
[§]Young Graduate Trainee, Advanced Concepts Team, European Space Agency, Keplerlaan 1, 2201 AZ, Noordwijk, NL.
[¶]Research Fellow, Advanced Concepts Team, European Space Agency, Keplerlaan 1, 2201 AZ, Noordwijk, NL.
[‖]Lead Data Scientist, ML Analytics, 2670-560 Loures, Portugal.
[**]Mission Analysis Lead, Mission Analysis and Flight Dynamics Team, ispace-inc, Sumitomo Shibakoen Building 10F, 2-7-17, Shiba, Minato-ku, Tokyo, Japan 105-0014.
[††]Flight Dynamics Engineer, Mission Analysis and Flight Dynamics Team, ispace-inc, Sumitomo Shibakoen Building 10F, 2-7-17, Shiba, Minato-ku, Tokyo, Japan 105-0014.

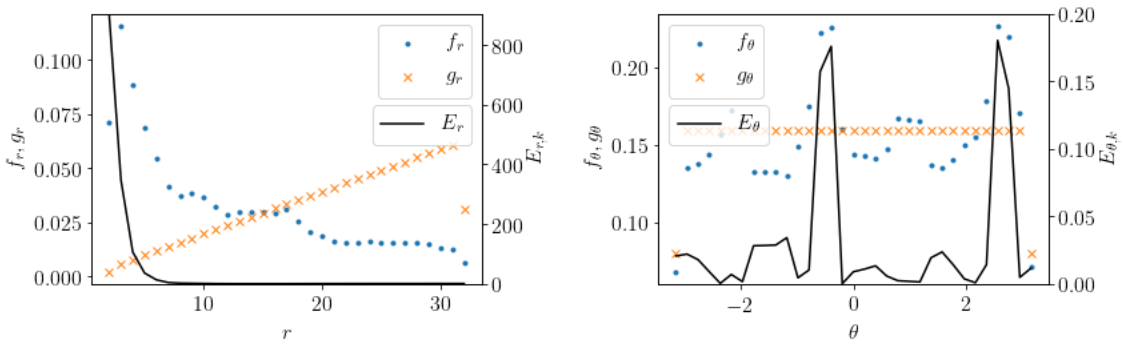

Figure 1. Actual and target distributions when the entire galaxy is settled. The resulting contributions to the errors E_r, E_θ are also shown.

from each settled star, makes the resulting graph also quite peculiar to this problem and (at least to us) unfamiliar. Finally, the trajectory design element, while maintaining some familiar approximations such as impulsive ΔV manoeuvres, also involves manoeuvre spacing constraints and multiple impulses capabilities that, when in connection to the underlying galactic dynamics, make the intuition gained in studying similar problems in the solar system rather untrustworthy, or anyway less accurate.

This report is structured to follow the logic used by the authors (team ESA-ACT) during the one month long GTOC-X competition which lead up to their final solution. First, we analyze the merit function given to us by the problem statement and discuss a few of its interesting properties. Then, we introduce the building blocks needed to design, in this context, single multiple impulses generational interstellar missions. The choice of mother ships and fast ships to get the galaxy settlement started is described in the following section. Then we describe the algorithm developed to grow the settlement trees, that is the self-replicating cycle of sending out new settler ships from previously settled stars to further spread mankind. In the following section, we describe a set of refinement techniques that further improve the settlement trees in particular with the goal to reduce the ΔV usage by rewiring the tree topology and adapting the epochs of settlement. Finally, we summarize the complete process that resulted in our best solution and conclude with an evaluation of its properties.

UNDERSTANDING THE MERIT FUNCTION

The merit function to maximize (discounted for the time bonus factor here not considered) has the form:

$$J = N\eta\sigma = N \underbrace{\left(\frac{1}{1+10^{-4}N(E_r+E_\theta)}\right)}_{\eta} \underbrace{\left(\frac{\Delta V_{\text{max}}}{\Delta V_{\text{used}}}\right)}_{\sigma}$$

where N is the number of stars settled, $E_r > 0$ and $E_\theta > 0$ are the errors made in matching two assigned target distributions g_r and g_θ for the star locations in the galaxy, ΔV_{max} is the maximum ΔV available and ΔV_{used} the one used cumulatively by all transfers. The term $\eta < 1$ is here called settlement efficiency and describes how close the r and θ distributions of the settled stars (f_r, f_θ) are

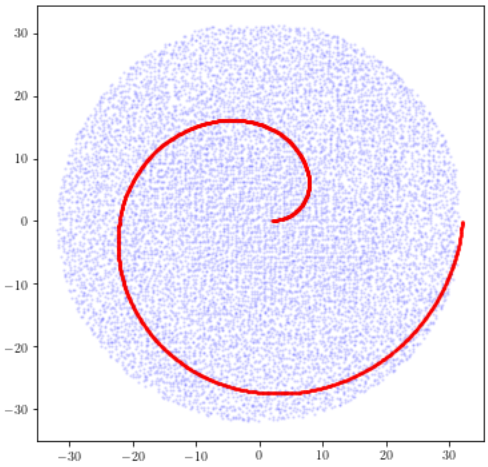

Figure 2. A uniform two dimensional distribution and an equivalent spiral having the same settlement efficiency.

to the target ones (g_r, g_θ). The last term $\sigma > 1$ is the ΔV efficiency and rewards low ΔV transfers on average. The first thing to note in this merit function is that the three terms, N, η and σ are coupled in nontrivial ways. It is difficult to increase N without reducing σ as the time to establish the settlements is limited to 90 million years (=90 Myr), and short interstellar transfers results in a larger ΔV. Increasing N also provokes, eventually, a drastic η decrease as the star distribution in the galaxy is far from the target one. This last point is illustrated in Figure 1 where f_r, g_r and f_θ, g_θ are shown for the case of the entire galaxy being settled, together with the resulting error contributions to E_r and E_θ. The excessive number of stars in the inner galaxy correspond to errors contributions of the order of hundreds driving η to 7.023×10^{-5}. On the other hand, the coupling between η and σ is less obvious and it is indeed possible, to some extent, to increase σ keeping both η and N fixed.

We note that the errors E_r and E_θ are independent: Given N points r_i, θ_i with a settlement efficiency η corresponding to a certain geometry in the two dimensional space, different geometries can be achieved by the points r_i, θ_{p_i}, where p_i is any permutation of the N indexes, while maintaining the same η. If, for example, we choose p_i to produce a monotonically increasing series of θ and r values, the points will be placed on a spiral spanning 2π. This fact is important as it informs us that it is not necessary to settle the galaxy uniformly in its two dimensions: it is also possible (according to the value of N), to find stars, for example, along some one dimensional spiral and obtain the very same settlement efficiency η. An example of such two iso-η spatial distributions is given in Figure 2. While settling along such a spiral would have benefits in terms of ΔV and available ships, it is not given that there is sufficient amount of stars close enough to achieve high J.

With regards to the computational complexity of the merit function we note that a single computation of η requires $\mathcal{O}(N)$ operations. For the design of our algorithms it is of interest to compute the single individual contributions to η, for example to select which stars to add and which to remove as to increase η. This operation can be implemented in constant time: Consider the case where the first N stars have orbital radius r_i and an associated efficiency η'. Say we want to add one more

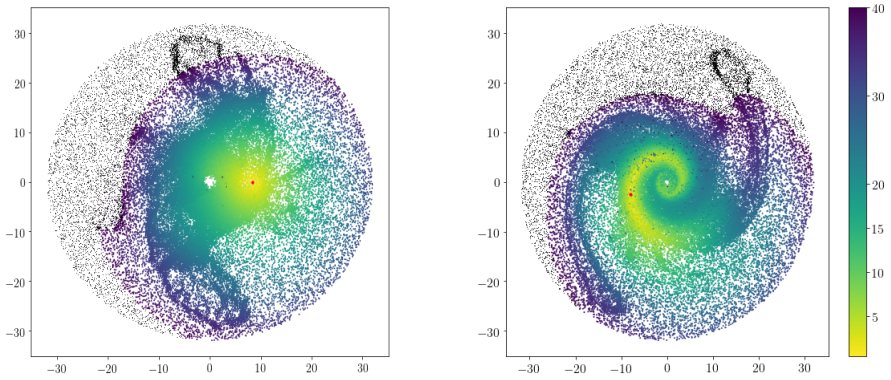

Figure 3. Minimum time trajectories of fast ships to all stars within 40 Myr. Star positions are visualized at 0 Myr (left) and 90 Myr (right). The red dot indicates the Sol system.

star at radius R and compute $\delta\eta = \eta - \hat{\eta}$. As part of the merit computation we need to compute the functions $f_r(r), f_\theta(\theta)$, an operation that is $\mathcal{O}(N)$. Following the original notation in[1] and limiting our argument to only f_r (for the angular equivalent f_θ a similar argument applies), we can write:

$$(N+1)f_r(r) = N\hat{f}_r(r) + f(r; R, s_r)$$

where $\hat{f}_r(r)$ is the f_r function computed for N stars. Thus the cost to determine $\delta\eta$, once η' and thus $\hat{f}_r(r)$ have been computed, is constant. One can thus sort any ensemble of N stars with respect to their individual contributions to η with complexity $\mathcal{O}(N \log N)$ and compute the $\delta\eta$ for all N stars of a given ensemble with in $\mathcal{O}(N)$.

DESIGNING FAST SHIPS, MOTHER SHIPS AND SETTLER SHIPS

In order to design multi-impulse trajectories to transfer between stars in the galaxy, two main building blocks are needed: a) an efficient numerical integrator for the differential equations describing the motion of the interstellar ships in the galactic medium and b) a solver for the galactic Lambert problem.[2] For the first block we implement a Taylor propagator,[3] while for the second we make use of Powell hybrid method[4] to find the starting velocity v_1 that reaches, in T, the prescribed final position r_2 from r_1. We quickly introduce the different types of ships:

Fast Ships

Fast ships have a total ΔV budget of 1500 km/sec and $n = 2$ impulses to perform their interstellar jump. The design of fast ship trajectories to a given star s requires a solution to the following problem:

$$\begin{aligned} \text{minimize:} \quad & T \\ \text{subject to:} \quad & \Delta V_1(T) + \Delta V_2(T) \leq 1500 \end{aligned}$$

where the two velocity increments ΔV_1 and ΔV_2 are computed solving the Lambert problem defined by r_{sun} to $r_s(T)$ and T. Given the simple capacities of the fast ships, it was possible to compute their two-impulse transfer to all stars in the galaxy exhaustively. Figure 3 shows at which times each

star is reachable from Sol given a maximum flight time of 40 Myr (to allow settlement trees enough time to grow).

Mother Ships

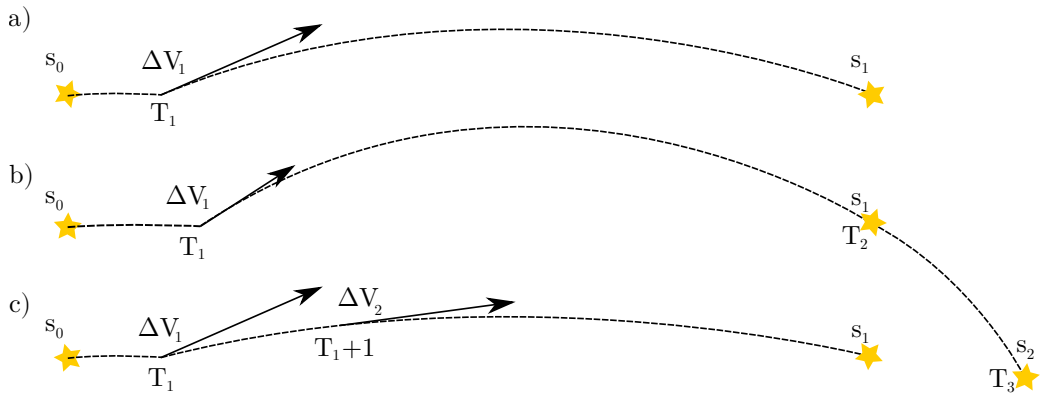

Figure 4. Different leg types for building mother ship trajectories. a) Basic impulsive leg. b) Two stars one impulse. c) One star two impulse

The ΔV budget of a mother ship is 500 km/sec, three times lower than that of a fast ship. While a mother ship may perform up to $n \leq 3$ impulses instead of only two, each impulse is limited to 200 km/sec and restrictions apply on when impulses are allowed. The particularity of the mother ship is that it is capable of releasing up to 10 settlement-pods to settle multiple star systems during its flight. The settlement pod has a ΔV budget of 300 km/sec to rendezvouses with the target star, therefore the mother ship has no need to match the velocity of the star to settle it and can simply pass by. Impulses of the mother ship have to be 1 Myr apart from each other and also 1 Myr between settlement attempts, limiting the maneuverability of this type of ship. To make full use of the available settlement pods, the task is to align the available impulses such that multiple star systems may be settled during each leg.

Below we discuss three different mother ship leg types, that can be combined in trajectories as long as ΔV and impulse constraints are met. A basic impulsive leg targets a single star with an impulsive manoeuvre. An alternative is to target two stars with a single impulse. Lastly, we can use two impulses to reach a single star faster. Figure 4 illustrated all three leg types.

Basic impulsive leg The most basic mother ship trajectory can visit three stars with three impulses, solving three separate Lambert problems. In this case each leg targeting a star s can be treated as an independent optimization problem:

$$\begin{aligned} \text{minimize:} \quad & T \\ \text{subject to:} \quad & \Delta V_{\text{mother}} \leq \min(\Delta V_{\text{left}}, 200) \text{ km/sec} \\ & \Delta V_{\text{pod}} \leq 300 \text{ km/sec} \end{aligned}$$

Two stars one impulse In order to deploy more of the available settlement pods, legs can be extended to target not just one, but two stars simultaneously with a single impulse: We target the first star s_1 by solving a Lambert's problem, and set the time of the first impulse (T_1) and the time-of-flight (T_2) such that a second star s_2 is also passed at a later time (T_3). This results in a

decision vector with three independent time components (T_1, T_2, T_3), which allow us to minimize the distance between the final position of the mother (r_{mother}) and the second target star (D_{s_2}).

$$\begin{aligned} \text{minimize:} \quad & D_{s_2}(T_1, T_2, T_3) \\ \text{subject to:} \quad & \Delta V_{\text{mother}} \leq \min(\Delta V_{\text{left}}, 200) \text{ km/sec} \\ & \Delta V_{\text{pod}_1} \leq 300 \text{ km/sec} \\ & \Delta V_{\text{pod}_2} \leq 300 \text{ km/sec} \\ & r_{\text{mother}}(T_2) = r_{S_1}(T_2) \\ & r_{\text{mother}}(T_3) = r_{S_2}(T_3) \end{aligned}$$

Deploying this technique to each leg, these mother ship trajectories can settle up to six stars.

One star two impulses In some cases we can benefit from firing two subsequent impulses: essentially providing an extra kick that allows the mother ship to reach farther in a given time frame, or arrive to a star earlier. Settling a star earlier has exponential benefits a few generation later. Alternatively, by flying further a mother ship can potentially cover areas that are only reachable with fast ships otherwise.

In these legs we fully constrain the second impulse with regards to its time, direction and the magnitude: it is done 1 Myr after the first impulse with 200 km/sec in the direction of the current velocity vector. Given these constraints, the second impulse can be treated as a property of the propagation, therefore the problem effectively reduces to a simple Lambert's problem, solved with a similar optimization as for the *basic impulsive leg*.

Settler Ships

Settler ships are the main part of the fleet as the majority of settled stars are due to settler ships visiting. Once a star has been settled (which consumes the original settler ship), it will generates up to 3 new settler ships that can be deployed to further advance the settlement tree after 2 Myr.

A settler ship has a total ΔV budget of 400 km/sec, $n \leq 5$ impulses to perform the interstellar jumps and a constraint on each impulse magnitude of $\Delta V_i < 175$ km/sec. Consequently, a two impulse strategy is insufficient to exhaust the total ΔV budget. Therefore, in some transfers when the time of flight is sufficiently long, additional impulses may be included.

For most of the time we are interested in time-optimal transfers of settler ships, but when a maximum travel time is defined, corresponding mass-optimal transfers can be generated by minimizing the accumulative ΔV of all impulses.

ESTIMATION OF NUMBER OF SETTLED STARS ACHIEVABLE

Some initial back-of-the-envelope calculations can be made to estimate the maximum number of stars that can be settled given an average time-of-flight between settled stars. A simple exponential growth model was created for that purpose:

$$N_t = 3n_p + 3\sum_{n=1}^{n_p} {n_{sp}}^{G_{\text{set},i}} + 2{n_{sp}}^{G_{\text{set},f}}$$

where N_t is the total number of stars settled, $n_p = 10$ is the number of stars settled by each mother ship, $n_{sp} = 3$ is the number of settlers spawned by each settled star, and $G_{\text{set},i}$ is the number

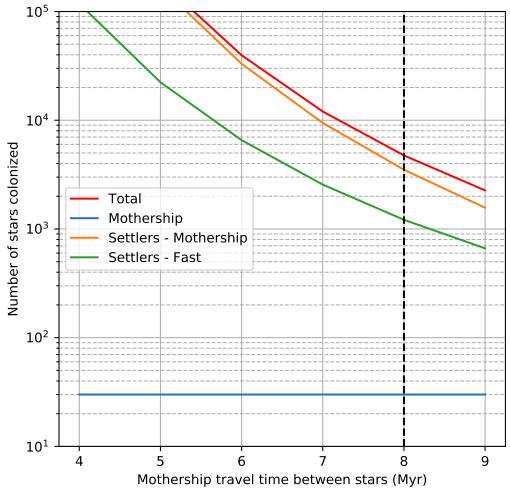

Figure 5. Number of stars settled vs time-of-flight between stars for the mother ships. The various curves represent the number of stars settled by the mother ship pods, by settler ships tracing back to mother ship settlement nodes and by settler ships tracing back to fast ship settlement nodes. The vertical dotted line marks the nominal time-of-flight between stars for a mother ship.

of settler spawning generations originating from a star i originally settled by a mother ship (or a star f settled by a fast ship). The number of settler spawning generations is given by:

$$G_{\text{set},i} = \frac{E_{\text{fin}} - E_{\text{set},i}}{t_{\text{flight}} + t_{\text{wait}}}$$

where $E_{\text{fin}} = 90$ Myr is the end epoch for the problem, $E_{\text{set},i}$ is the epoch of the original settlement of a star i by a mother ship or a fast ship, t_{flight} is the average time-of-flight for settler ships from their spawning star to the star they will settle, and $t_{\text{wait}} = 2$ Myr is the wait time for a settled star to spawn settler ships.

The nominal settlement epochs $E_{set,i}$ for the stars settled by the mother ships (or the time-of-flight between stars for the mother ships) were set 8 Myr apart, so that the ten pods would be able to be deployed within the 90 Myr time limit with some margin. The time-of-flight between stars for the settler ships t_{flight} was equal to that of the mother ships divided by the ΔV ratio between settler and mother ships, i.e. the settler ships were considered as having a time-of-flight between stars of 1.25 times that of the mother ships. For the two stars initially settled by the fast ships, the settlement epochs $E_{\text{set},f}$ were set to 20 Myr. It was also assumed that it would be possible for all three settler ships spawned by a settled star to reach another star to settle at each settlement generation.

The number of stars settled at an epoch of 90 Myr was calculated in the above simple model, as a function of the time-of-flight of the mother ship between stars. The results are given in Figure 5.

For the baseline time-of-flight between stars for the mother ships of 8 Myr, a total of $\sim$ 5000 stars are settled. The majority of stars are settled by settler ships spawned by the mother ship pods, and in particular from the earlier settlement nodes where the exponential settlement trees have had time to grow. Settler ships spawned by fast ship settled stars contribute less, but still significantly.

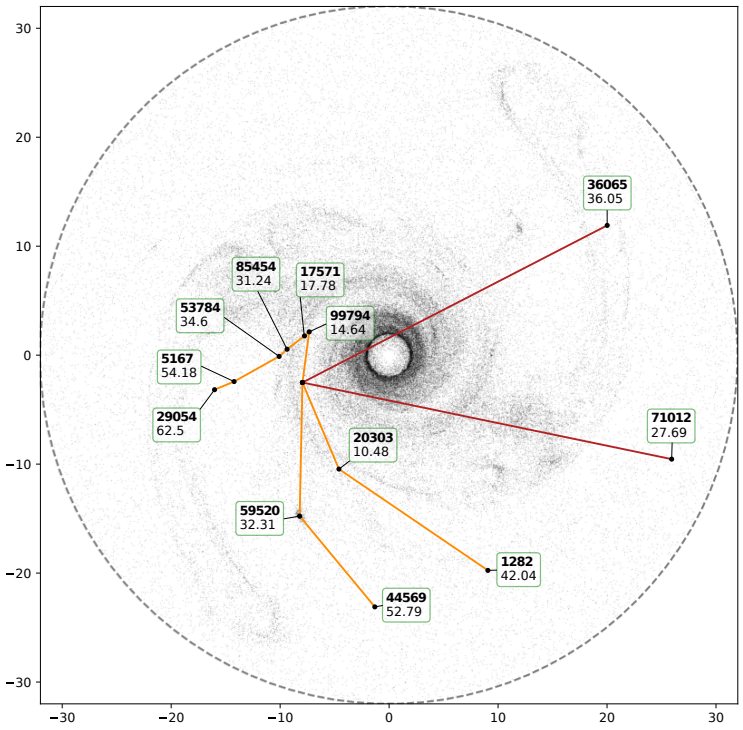

Figure 6. Stars visited by the fast- and mother ships rendered at 90 Myr. Star ID and settling time (in Myr) indicated for each star. Connecting lines show visiting order, not the actual trajectory.

If the fast ship settled stars can be settled at an earlier epoch, the exponential growth from them will approximate that of the mother ship settled stars. However, this will be at the cost of proper star distribution, as fast ships are expected to be crucial in reaching distant areas of the galaxy. A very high sensitivity to changes in time-of-flight between stars is observed, with a high pay-off in number of inhabited stars if stars can be settled at faster rates. Shorter times-of-flight however will likely result in a worse ΔV efficiency. Therefore, a trade-off emerges between N and σ.

THE CONCURRENT TREE SEARCH

Each settlement created by a fast ship or a mother ship is the root node of a *settlement tree*. Starting from this root node, the settlement tree grows by sending settler ships to other stars, which become new nodes in the tree once a settlement is establishment. All settlement trees together constitute the *settlement forest*, which encapsulates the largest part of a solution for the GTOC-X challenge. This section describes our strategy for generating settlement forests based on the initial conditions.

Initial Conditions

The initial conditions are defined by the combination of mother ships and fast ships used to settle the first stars. It is important to select root nodes that permit the growth of good settlement trees such that the final settlement forest achieves a high merit J. Figure 6 shows an example of valid initial conditions, highlighting the star id of each root node and its epoch of settlement. The

particular initial conditions of Figure 6 are part of our final solution, whose complete generation will be explained at a later section.

Why a Concurrent Search?

In a naive greedy search, a tree is grown by selecting a departure star from the tree and a target star from a set of possible targets, and adding this star to the tree. The target star is selected in order to minimize a cost function like the time of flight (tof) or the ΔV_{used} of the corresponding settler ship transfer.

A closer look reveals that neither of those cost functions supports the goal of achieving a high merit after the time window of 90 Myr ends: A tof-greedy search leads to an exponential growth with high N (up to 25,000 in our preliminary tests), but tends to cluster, reducing η and hence J to values on the order of 1. A ΔV_{used}-greedy search is too conservative and settles only a small number of stars, limiting the merit on the order of 10. Further complications arise from the fact that trees grown independently from different root nodes might lead to overlaps and thus a waste of resources. To mitigate this issue, we design our search strategy to grow the entire settlement forest simultaneously, i.e. by growing the settlement trees *concurrently*.

The concurrent search also needs to be guided by a cost function: While it is tempting to select the next settlement according to the highest possible increase in η this would neglect that the final merit J is dominated by the product of η and N. For example, a distribution with 10 stars and an efficiency of 80% has a merit of 8 whilst a distribution with 1000 stars and an efficiency of 40% has a merit of 400 (assuming both solutions have $\sigma = 1$). An ηN-greedy concurrent search avoids this issue by minimizing the reduction in η at each step while increasing the number of stars in the settlement forest.

Overview of the Algorithm

The pseudo-code in Algorithm 1 describes the essential steps required for the concurrent tree search and the setup of its parameters.

To settle any additional star during the search, a time-optimal transfer needs to be computed. While the computation of a single 2-impulse time-optimal transfers takes, on average, 50 ms on our servers, this is not sufficient for an exhaustive search: Assuming 18 root nodes and $99,982$ possible target stars, the first step of an exhaustive search would require the computation of $18 \times 99,982$ transfers and thus lasting for $1,799,676 \times 50/1000 \approx 25$ hours. The second step would take 26.4 hours, and extrapolating this further implies that for settlement trees with 2500 stars, it would take more than 10 years to find a single solution! From this it is clear that the number of transfer computations must be reduced.

The number of transfer computations required is determined by the number of stars that are in the settlement forest and the number of possible targets. Since the search gradually increases the number of stars in the settlement forest, the only way to reduce the number of transfer computations is to restrict the set of possible targets. We deploy the following strategies to reduce the amount of computations required:

- Choosing a background set of stars: Since one ideal distribution is a grid in the xy-plane, we decreased the total number of stars to $\approx 20,000$ out of the $100,000$ by a selection based on grid points of a finite cell-size (0.25 pc) in the xy-plane.

input : Set of root nodes $\mathcal{D}_0$, set of possible targets $\mathcal{A}_0$
parameters: number of candidates from ranking $m_0 = 10$,
maximum number of candidates $m_{\text{max}} = 200$,
minimum time of flight for 4-impulse $tof_{\text{min}} = 5$ Myr,
minimum generation $g_{\text{min}} = 5$
output : Settlement trees S
begin
  $m \longleftarrow m_0$
  $n \longleftarrow 1$
  $S \longleftarrow \{\}$
  Let G be a hashmap such that $G[d] = 1, \forall d \in \mathcal{D}_0$
  while $m < m_{max}$ **and** $|\mathcal{A}_n| > 0$ **do**
    $\mathcal{T}_n \longleftarrow \{\}$;
    for $d \in$ `GetStarsWithShips(`$\mathcal{D}_n$`)` **do**
      $\mathcal{C} \longleftarrow$ `GetCandidateStars(`d, m, $\mathcal{A}_n$`)`
      $\mathcal{C}' \longleftarrow$ `FilterByEfficiency(`$\mathcal{C}$`)`
      for $t \in \mathcal{C}'$ **do**
        if *transfer t is feasible* **then**
          $\mathcal{T}_n \longleftarrow \mathcal{T}_n \cup \{(d, t)\}$
        end
      end
    end
    if $|\mathcal{T}_n| > 0$ **then**
      $(d, t) \longleftarrow$`FindMaxEfficiencyImprovement(`$\mathcal{T}_n$`)`;
      $\mathcal{D}_{n+1} \longleftarrow \mathcal{D}_n \cup \{t\}$
      $\mathcal{A}_{n+1} \longleftarrow \mathcal{A}_n / \{t\}$
      $G[t] \longleftarrow G[d] + 1$
      if $tof > tof_{min}$ **and** $G[t] < g_{min}$ **then**
        Use 4-impulse transfer
      else
        Use 2-impulse transfer
      end
      $\mathcal{S} \longleftarrow S \cup \{(d, t)\}$
    else
      $m \longleftarrow 2m$;
    end
    $n \longleftarrow n + 1$;
  end
end

Algorithm 1: Concurrent tree search

- Ranking stars and selecting the top m candidates: Instead of computing the transfer from a star to all possible stars, we only compute transfers to the m nearest stars to the departure star as the resources required to reach distant stars become too expensive. To avoid discarding stars that lead to favourable transfers, a suitable ranking criteria that allows for a fast computation needs to be in place. Previous works have shown that proximity in terms of Euclidean distances does not necessarily imply favorable transfer times.[5] Thus, instead of the Euclidean distance we rank our stars according to the orbital phasing indicator that we deployed already in previous GTOC challenges.[6]

- Filtering stars that do not meet the minimum efficiency requirement: As the transfer computations take the largest bulk of the time of the inner loop in Algorithm 1, the stars are filtered based on the selection criteria before the transfer is computed.

- Caching: Due to the fact that $\mathcal{T}_{n+1} \cap \mathcal{T}_n \neq \emptyset$, previously computed transfers can be saved in a look-up table to avoid their reevaluation.

Selecting a Star

Adding a star to the current settlement forest changes the efficiency of the distribution by a certain amount that we can compute directly. If we denote the efficiency of a distribution as $\eta(\mathcal{D}_n)$, then the change in efficiency is $\Delta\eta(t) = \eta(\mathcal{D}_n \cup \{t\}) - \eta(\mathcal{D}_n)$.

During the search, we filter out all target stars that do not meet the minimum $\Delta\eta$ threshold. This threshold is set adaptively according to the following formula

$$\Delta\eta(t) > \frac{k - \eta(\mathcal{D}_n)}{|\mathcal{D}_n| + 1}$$

where $k = -0.01$ is a constant that measures the change in $\eta|\mathcal{D}_n|$ and was chosen based on experimentation.

After computing all the transfers to the filtered set of targets, we select the star that maximizes $\Delta\eta$ and use that as a transfer target. Since multiple departure stars may have transfers to this target, we pick the transfer with the shortest tof.

Depending on the hop distance towards the root node of the corresponding settlement tree, we optimize the transfer either for 4 impulses (departing star is not further than 5 hops away from the root) or 2 impulses (otherwise). Exceptions are transfers that have a shorter time of flight than 5 Myr, which are optimized for 2 impulses regardless of their distance. Transfers with 4 impulses can utilize the full ΔV budget of settler ships and thus save time during the early phases of the solution which in turn accelerates the spreading of subsequent settlements.

Note that all settler ship transfers, regardless of their number of impulses, are always optimized for minimum transfer times.

TOPOLOGICAL REFINEMENTS OF SETTLEMENT TREES

Topological refinements are post-processing steps that take a settlement forest (as delivered by the concurrent tree search) and aim to either improve the merit J by means of local modifications or prepare the settlement trees for further optimization. In the following, we discuss these techniques in detail.

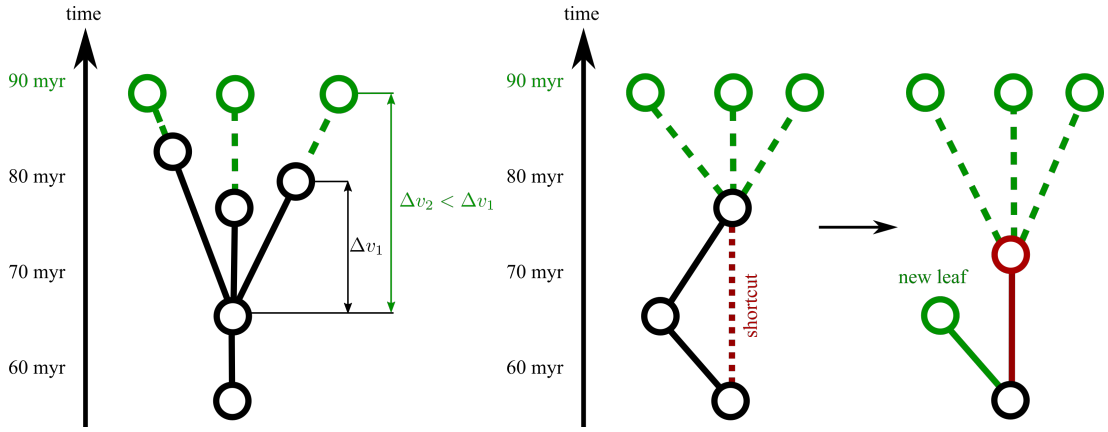

Figure 7. Left: The mass-optimal transfer to the leaf stretches the epoch of arrival to 90 Myr by saving ΔV. Right: An inner node gets rewired by a shortcut, turning its previous parent into a leaf and allowing for further leaf stretching.

ΔV-refinement

This refinement consists of a collection of operations which are aimed to increase σ while keeping the set of settled stars (and thus N and η) unchanged. Since the concurrent tree-search is designed to deliver time-optimal transfers, the ratio σ can often be improved by substituting the time-optimal transfers towards leaf-nodes with mass-optimal transfers (constrained to an arrival before the 90 Myr deadline). We call this operation *leaf stretching* as it stretches, for the majority of cases, the time of the settlement of the leaf node to 90 Myr, resulting in a lower total ΔV.

While transfers to leaf nodes can be stretched, transfers to inner nodes are more challenging to optimize, as the time schedule of inner nodes needs to allow for the successive settlements to take place. However, there exist situations for which inner nodes can be turned into leaf nodes without the risk of violating any scheduling constraints. For example, an inner node can turn into a leaf if all its children become settled by different means, i.e. by utilizing inner nodes that are still capable of sending additional settler ships. These unsent ships may also create opportunities to establish shortcuts within a tree and thus gain some time for further refinement steps. Figure 7 illustrates the principle between node stretching and shortcuts.

Since the out-degree of each node can never be larger than 3 (the maximum amount of new settler ships that are generated 2 Myr after a star has been settled), it follows that each inner node can reach at most 12 other nodes within 2 hops. Fixing this inner node as the root of a settlement-subtree, we can exhaustively enumerate all possible sub-trees that can be constructed to visit these 12 other nodes. If any of these nodes are not leaf nodes and need to spawn new settler ships, we add the corresponding arrival-times as constraints to the optimization problem and minimize the cumulative ΔV for all transfers involved. We call the solution of this problem the *mass-optimal equivalent subtree*, as it effectively settles the same stars but rewires transfers in order to save ΔV in comparison to the original subtree. Figure 8 shows a schematic depiction of this rewiring process.

We start searching for mass-optimal equivalent subtrees beginning from the grandparents of the leaves and working ourselves iteratively deeper until we reach the root of the corresponding settlement tree. Because this rewiring processes allows nodes to change their depth in the tree, it can be applied repeatedly, however with diminishing returns.

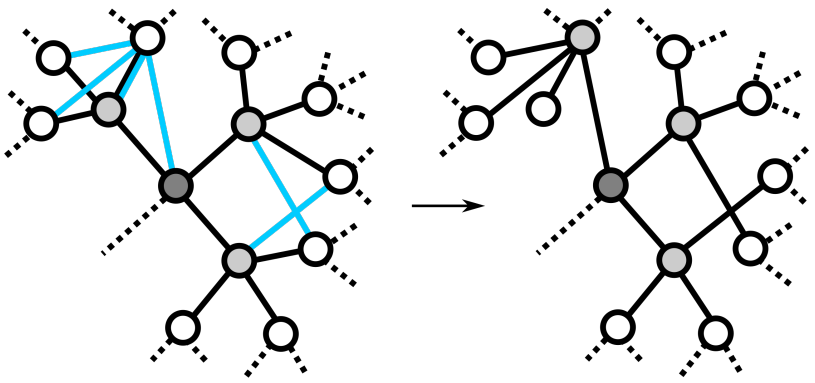

Figure 8. An exhaustive enumeration of all possible subtrees allows to find the mass-optimal equivalent subtree (right) by rewiring inner nodes over two levels of depth while fulfilling arrival and degree constraints.

Reduction

Reduction is an aggressive refinement strategy that checks whether there exists a subset of leaf-nodes whose removal would result into an increase of η. Finding the optimal subset to remove can be a computationally demanding problem, which is why we deploy a heuristic that aims to minimize E_r and E_Θ. The heuristic constructs two histograms of r and Θ and identifies overflowing and underflowing bins of stars with respect to an optimal distribution. It then proceeds to remove random subsets of leaf nodes from an overflowing bin in one of the two histograms while avoiding unbalancing the other histogram. Once removed, nodes may also be reinserted again in case an underflowing bin would benefit from it. We terminate the heuristic if after several trials η does no longer improve. This refinement is particularly aggressive as it decreases N without paying any attention to σ, and consequently can lead to a lower merit J if the improvement in η can not compensate.

Regrowth

Regrowth is a refinement that takes an existing settlement tree, removes branches whose removal improves ηN, and then runs the concurrent tree search on the shrunken settlement forest. The concurrent tree search takes transfers that are ηN-optimal with respect to the settlement forest at each of its iterations, but these transfers may not necessarily be ηN-optimal at later stages of the search. Thus, some earlier transfers can be removed and new branches can be grown that improve ηN. The regrowth procedure grows these new branches concurrently in order to keep the efficiency as high as possible.

Padding

Padding is a refinement strategy which aims to increase N by settling additional stars (preferably with a low ΔV) from nodes that are still capable of spawning settler ships. It is possible that due to the background distribution deployed during the concurrent tree search, some useful transfers have been overlooked, which is why for this refinement the complete local neighborhood of a star is taken into consideration. Padding always improves N but typically leads to a decrease in η, as it tends to produce clusters. Thus, a padding transfer is only kept if its potential increase in σ compensates for any decrease in η.

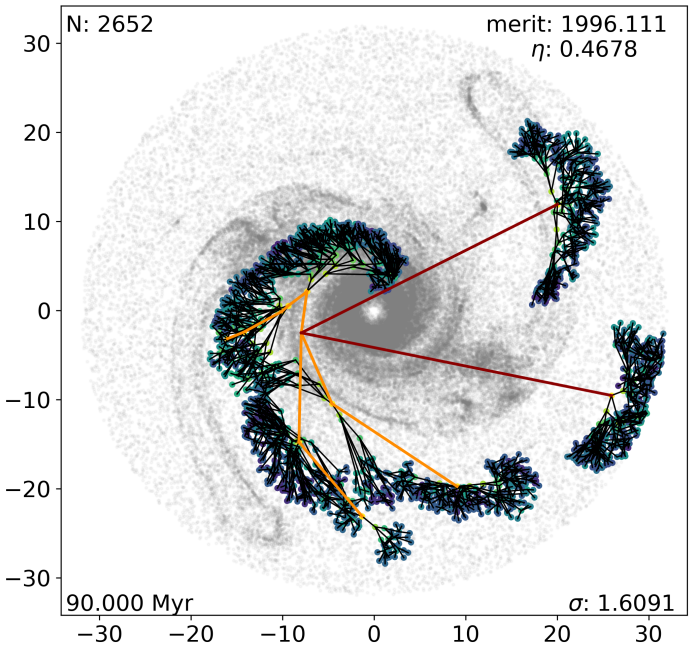

Figure 9. Our best found solution at time $t = 90$ Myr. Dark red lines indicate fast ships, orange lines mother ship legs. Shown is not the actual trajectory but straight line segments between settlement maneuvers.

THE BEST SOLUTION FOUND

During the four week long competition, multiple solutions were generated that allowed our team to learn about the merit function. Based upon our insights, we took the following high-level decisions for the design of our final solution:

1. Both fast ships travel to the outer rims of the galaxy to settle stars with approximately $r > 20$ kpc and $-1 \leq \Theta \leq 1$.

2. One mother ship releases a settlement pod quickly to spread settler ships towards the center of the galaxy $r < 10$ kpc and $1 \leq \Theta \leq 3$. Further settlement pods are targeted for the eastern areas of the galaxy ($\Theta \approx \pm 3$).

3. The remaining two mother ships are spread out to fill in missing regions at the south of the galaxy, roughly corresponding to distances between 10 and 20 kpc and angles $-3 \leq \Theta \leq -1$.

Several candidate trajectories for all three parts of this strategy have been developed, but required further selection. Balancing out our computational resources, we performed a search on several 100 different combinations of fast and mother ships and, which we picked by hand. Each combination resulted in a concurrent tree search that took from 5 to 10 hours on a single core CPU.

Out of those runs, our highest scoring solution achieved a merit of $J = 1177$ without any refinements. Applying the ΔV-refinement improved σ from 1.14 to 1.52, resulting in $J = 1591$. Based on the refined settlement trees, three consecutive optimization cycles were started, with one cycle consisting of the following refinements in that order: reduction, regrow, ΔV-refinement and padding.

After the first cycle, J increased to 1740, after the second to 1934 and after the last cycle to 1972. This score was improved further by repeating the ΔV-refinements and adding some padding, resulting in our final solution with $J = 1996$, settling $N = 2652$ stars with an efficiency of $\eta = 0.46778$ and $\sigma = 1.60906$. Figure 9 shows this solution, which we submitted at the end of the competition. An animation of our solution can also be found on Youtube.[7]

ANALYSIS OF OUR SOLUTION

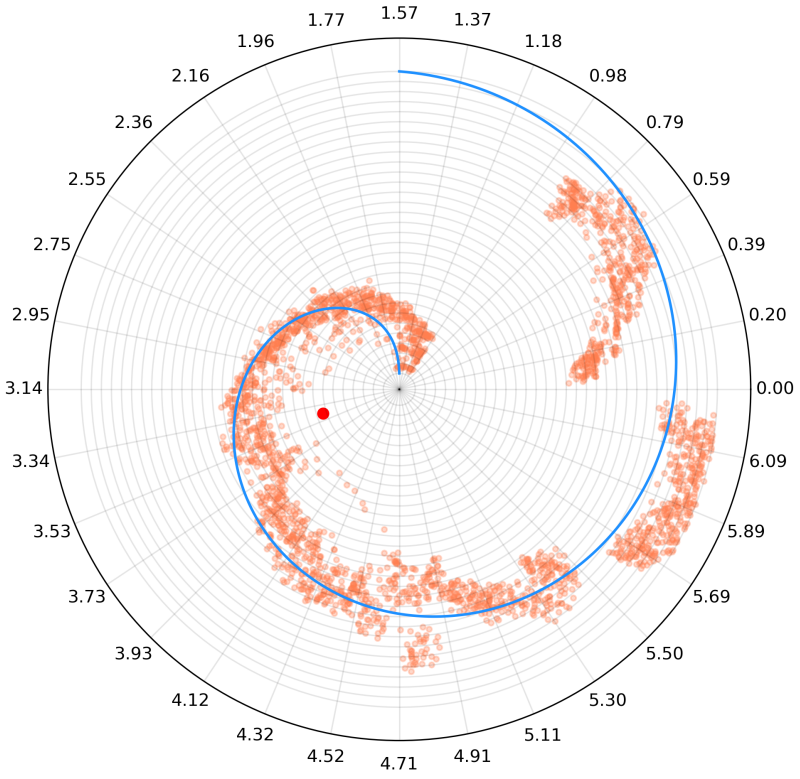

Figure 10. Overlay of our best found solution at 90 Myr with a spiral of optimal efficiency.

There are several points worth noting about our solution: Coming back to the initial analysis of the merit function, we observe that the final settlement forest indeed resembles a spiral-like pattern. Figure 10 shows an overlay of an optimal efficiency spiral with our solution, illustrating the (comparably) high efficiency η that our overall strategy achieved. The emergence of the spiral pattern can be partly attributed to the geometry of the initial root nodes. Figure 6 shows how the root nodes were established: One of the three mothers ship trajectories produces 6 settlements by deploying the two stars one impulse strategy at each leg. The two other mother ships follow the one star two impulse strategy, each only establishing 2 settlements, but travelling larger distances to

target areas of high r. The two fast ships break the continuity of the spiral pattern but still allow for a coverage of large chunks of r and Θ regions, which is essential for high efficiency.

With respect to the settlement trees we note that the concurrent tree search is capable of growing various different shapes, but generally avoids creating dense clusters of stars (which would be, most likely, detrimental for η). Instead we observe that the first transfers in the settlement trees tend to spread out over larger distances to create local branches of settlements. Moreover, the settlement trees grow outwards, as there are more settled stars needed in the outer areas of the galaxy than in the center. This outward growing is induced by the refinement cycles, as inward growing branches are pruned while new transfers to outward regions are favoured.

CONCLUSIONS

In this work we analyzed the galaxy settlement challenge that was introduced as the problem of the GTOC-X. Identifying the efficiency η as one of the major factors influencing the merit, we designed fast ships and mother ship transfers to settle promising stars for further development. While we did not explicitly design our algorithms to generate solutions following a spiral pattern, our exploration of the design space converged to a settlement distribution close to an optimal spiral. The concurrent tree search demonstrated its potential to directly optimize two of the three major factors of the merit function and thus capturing large parts of the complexity of this challenge. Lastly, we believe that the challenge was able to give us fascinating insights into the design of interstellar travel and the capabilities of intelligent life to spread through our galaxy.

REFERENCES

[1] A. E. Petropoulos, E. D. Gustafson, G. J. Whiffen, and B. D. Anderson, "GTOC X: Settlers of the Galaxy Problem Description and Summary of the Results," *Astrodynamics Specialist Conference, Portland, Maine, 11-15 Aug.*, 2019, pp. AAS 19–891.

[2] D. Izzo, "Revisiting Lamberts problem," *Celestial Mechanics and Dynamical Astronomy*, Vol. 121, No. 1, 2015, pp. 1–15.

[3] À. Jorba and M. Zou, "A software package for the numerical integration of ODEs by means of high-order Taylor methods," *Experimental Mathematics*, Vol. 14, No. 1, 2005, pp. 99–117.

[4] M. J. Powell, "A hybrid method for nonlinear equations," *Numerical methods for nonlinear algebraic equations*, 1970.

[5] D. Hennes, D. Izzo, and D. Landau, "Fast approximators for optimal low-thrust hops between main belt asteroids," *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2016, pp. 1–7.

[6] D. Izzo, D. Hennes, L. F. Simões, and M. Märtens, "Designing complex interplanetary trajectories for the global trajectory optimization competitions," *Space Engineering*, pp. 151–176, Springer, 2016.

[7] D. Izzo, M. Märtens, E. Öztürk, M. Kisantal, K. Konstantinidis, L. F. Simões, C. H. Yam, and J. Hernando-Ayuso, "Settling the Galaxy (ACT's entry to GTOCX)," `https://youtu.be/nlnlORplDTI`, 2019. Accessed: 2019-07-29.