Acta Futura 11 (2018) 91-97
DOI: 10.5281/zenodo.1139268

Acta Futura

GTOC 9: Results from the University of Colorado at Boulder (team CU Boulder)

NATHAN L. PARRISH*, DANIEL J. SCHEERES, SIMON TARDIVEL,
CHANDRAKANTH VENIGALLA, JONATHAN AZIZ, MARIELLE PELLEGRINO,
OSCAR FUENTES, STIJN DE SMET

UNIVERSITY OF COLORADO, COLORADO CENTER FOR ASTRODYNAMICS RESEARCH
ECNT 320, 431 UCB, BOULDER, CO 80309-0431

Abstract. This paper describes the strategies and results of the GTOC9 competition for the team from the University of Colorado, Boulder. The goal of the competition was to remove 123 pieces of space debris for the lowest cost, with cost defined in Euros as a function of fuel mass, number of launches, and time into the competition window. The overall strategy for this team was: 1) Find the set of all possible low-cost chains of debris to visit, 2) Pick from those to define a series of missions that visit most of the debris, 3) Stitch the remaining 20-30 debris onto the existing chains, 4) Add 1-4 more launches to reach the 5-10 debris that remain after stitching, 5) Adjust the dates of each mission slightly to minimize ΔV, and 6) Find a series of four maneuvers to transfer from each debris to the next in the fully-integrated dynamics. The final solution removed all 123 pieces of debris with 17 launches for a cost of 1150.8 MEUR.

1 Introduction

The GTOC9 competition defines a set of 123 pieces of debris that are in approximately sun-synchronous orbits, with inclination near 98°and semimajor axis near 7,000 km. The motion of spacecraft is defined by numerically integrating Earth point mass gravity perturbed by the J_2 effect. The dynamics of debris are the analytical approximation of J_2 dynamics, with the state at any time given by analytically propagating mean motion, and constant secular drift rates of Right Ascension of the Ascending Node (RAAN) and argument of perigee.

We find that the secular drift of the RAAN due to Earth's J_2 is a primary driver for solutions. The node drift rate is given by

$$\dot{\Omega} = -\frac{3}{2} J_2 \left(\frac{r_{eq}}{p}\right)^2 n \cos i \tag{1}$$

with semilatus rectum $p = a(1-e^2)$ and mean motion $n = \sqrt{\frac{\mu}{a^3}}$. Since the eccentricity of all the debris pieces is nearly zero, the node drift rate $\dot{\Omega}$ is mostly driven by inclination and semimajor axis. For initial searches to find chains of debris that can be efficiently visited by a single spacecraft, we choose to ignore the drift of argument of perigee. The small eccentricity means that the cost of changing argument of perigee with fuel is very small compared to the cost of changing RAAN.

The goal of the competition is to remove all of the debris for the minimum cost, where cost in MEUR is

*Corresponding author, napa0706@colorado.edu

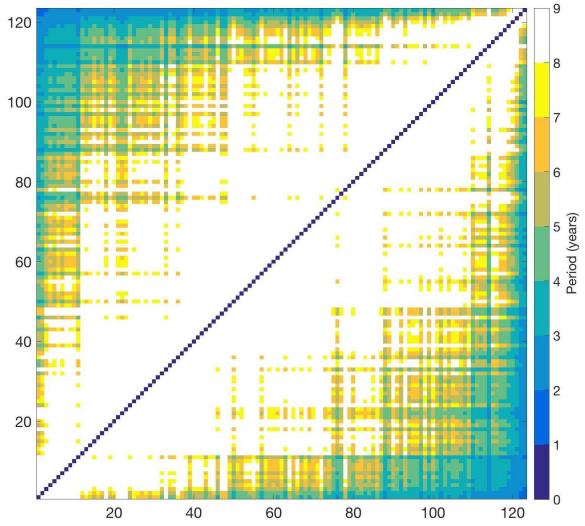

FIGURE 1. *The synodic periods (in years) of the RAAN of each debris object relative to all other debris objects. The debris are sorted by inclination from low to high. Debris that differ significantly in inclination have opportunities for transfers as frequently as every 1-3 years. Debris with very similar inclinations have synodic periods much longer than the 8 year time limit for the competition.*

given by the sum of each launch:

$$J = \sum_{i=1}^{N} \left[c_i + 2 \times 10^{-6} (m_{0_i} - m_{dry})^2 \right]. \quad (2)$$

The launch vehicle cost c_i grew linearly from 45 to 55 MEUR over the contest month. The term $(m_{0_i} - m_{dry})^2$ favors lower ΔV per mission. More details on the competition rules are given in [1].

We found that the greatest constraint on the search space is on time of flight between debris, as this limits the feasible transfers to debris with similar RAAN. Figure 1 shows the synodic periods of the RAAN of all the debris. Most pairs of debris have long synodic periods relative to the 8 year time limit of the competition, so transfers are largely limited to rare natural opportunities.

2 Analytical ΔV

Our first approach to finding low cost transfers between debris objects was to look at the natural crossings of debris RAAN. The RAAN drift rate $\dot{\Omega}$ varies per debris object due to variations in semimajor axis and inclination. The differential drift rate between two debris objects causes natural node crossings, which are opportunities for low ΔV transfers between the two objects at a particular time. The initial goal in exploring the problem was to solely use these natural node crossings to build all missions. However, it soon became clear that this approach was too limiting – there are not enough natural node crossings to solve the problem in a small number of launches, again due to time constraints for the competition. To find more options for cheap transfers, we developed analytical approximations for the cost to transfer between any two debris, then surveyed the available transfers. Transfer ΔVs were approximated by computing three terms: the cost of matching RAAN, the cost of matching inclination and semimajor axis, and the cost of matching the orbit phase (defined as argument of latitude).

The change in RAAN rate $\Delta\dot{\Omega}$ required to force the nodes to cross was calculated by setting a fixed transfer time (in most cases, 20 days) and calculating the difference in RAAN between the two debris objects at the fixed rendezvous time. The ΔV required to achieve the desired $\Delta\dot{\Omega}$ was approximated by assuming circular orbits and small changes in either inclination or semimajor axis. Under these assumptions, we get the following expressions for change of the node rate corresponding to a change in inclination ($\Delta_i\dot{\Omega}$) and to a change in semimajor axis ($\Delta_a\dot{\Omega}$).

$$\Delta_i\dot{\Omega} = \frac{\partial\dot{\Omega}}{\partial i}\Delta i \quad (3)$$

$$= -\dot{\Omega}\tan i\Delta i \quad (4)$$

$$\Delta_a\dot{\Omega} = \frac{\partial\dot{\Omega}}{\partial a}\Delta a \quad (5)$$

$$= -\frac{7}{2}\dot{\Omega}\frac{\Delta a}{a} \quad (6)$$

We can then write a simple relationship between orbit element change and its corresponding ΔV as follows:

$$\Delta i = \sqrt{\frac{a}{\mu}}\Delta V \quad (7)$$

$$\Delta a = 2a\sqrt{\frac{a}{\mu}}\Delta V \quad (8)$$

DOI: 10.5281/zenodo.1139268

Combining these last equations gives us final relationships between a desired $\Delta\dot{\Omega}$ and its corresponding ΔV.

$$\Delta V_i = \Delta_i \dot{\Omega} \frac{2}{3J_2R^2} \frac{a^3}{\sin i} \quad (9)$$

$$\Delta V_a = \Delta_a \dot{\Omega} \frac{2}{21J_2R^2} \frac{a^3}{\cos i} \quad (10)$$

Interestingly, comparing the efficiency of changing $\Delta\dot{\Omega}$ with an inclination change vs with a semimajor axis change by taking the ratio $r_{i/a} = \Delta_i\dot{\Omega}/\Delta_a\dot{\Omega}$ shows that they are equally efficient at $\tan i = -7$, or $i = 98.13°$. For inclinations greater than $98.13°$, a semimajor axis change is more efficient, while for inclinations less than $98.13°$, an inclination change is more efficient. One important caveat here is that for more aggressive maneuvers such as those used in "stitching" (see section 4), the assumptions made here break down. In these cases, a less efficient (but more accurate) approximation was used.

For large maneuvers, ΔV was instead approximated by solving Eqn 1 for either semimajor axis or inclination. The ΔV is then computed as the minimum required to change either semimajor axis or inclination. We did not consider cases where both semimajor axis and inclination were changed. If semimajor axis is used to change RAAN, the ΔV comes from the visviva equation. If inclination is used, then the maneuver cost is given by $\Delta V_i \approx 2V \sin(\frac{\Delta i}{2})$. Both ΔVs were calculated, and the smaller of the two was selected as the optimal transfer.

The cost to match inclination and semimajor axis of the target debris was approximated with a Hohmann transfer. The inclination change maneuver is combined with one of the semimajor axis maneuvers, and is chosen to occur at the radius of apogee of the orbit with the larger semimajor axis.

The final term of the ΔV approximation is the cost of matching the target debris' argument of latitude. This term was considered separate from the others and was approximated assuming circular orbits and ignoring the precession of the argument of perigee. Two phasing maneuvers are performed: one at the very start of the transfer time, and one at the very end. This approximation is the least accurate of the three terms, but it is also the smallest component. Adding this term effectively penalized shorter transfers, and it brought the total approximate cost closer to the truth.

Overall, we found that the approximate ΔV was accurate to within $\pm 30\%$ of the final integrated transfer.

3 Building Blocks

With the analytical estimate for ΔV described above, we then pre-computed all the possible chains of debris with an efficient algorithm, subject to the following criteria: Transfer time is exactly 20 days, the ΔV for each debris-to-debris transfer is $\leq 500m/s$, and the average ΔV of all the transfers in a chain is $\leq 200m/s$. This resulted in approximately 500,000 chains of debris with between 5-10 debris per chain.

Every chain was sorted by the rarity of the debris it contains, so that the debris that appear least frequently in all of the pre-computed chains are more likely to be selected early in the algorithm, while the debris that appear most frequently are left to the end. This was guided by the intuition that frequently-appearing debris will be easier to stitch on to existing chains later.

A randomized greedy search was then used to combine pre-computed chains together to find campaigns of missions that each visit unique debris at unique times. By randomizing the search, many possible campaigns of missions were generated, and the most promising were passed on to the next step.

After seeing the success other teams had with various genetic algorithms for this step, we recognize that the greatest improvement in score could be made by choosing a better set of initial chains.

4 The Stitcher

The building blocks algorithm was run several thousand times, and it would typically find 10-15 sets of chains that visit 75-95 of the 123 debris. The remaining debris were then left to the "stitcher" to attach to these chains, one by one.

4.1 The stitcher toolset

The stitcher toolset consisted of many routines, from computation of a single stitching action at the lowest level, to updating an entire campaign at its highest level. At this level, it would take as input a campaign, defined as a set of N missions visiting K debris. The algorithm would then proceed to attach the remaining debris to any of the missions. The stitcher would end with two exit conditions: if all debris were successfully attached, or if it was impossible to stitch some last debris to the already existing missions.

We will describe two important aspects of the stitcher: first how we optimized the low-level stitching

of a single debris attached to a single mission, then how the high-level algorithms handled these possible stitchings into a sorting tree to output the best result that we could find.

4.2 Optimizing the stitching of a single debris to a single mission

At the lowest level, one of the stitcher routines consisted in optimizing the stitching of single given debris to a single given mission. The goal was to minimize the total ΔV for the mission.

Two options were possible. The first one, simple fit, consisted in keeping the original sequence of debris as-is, and simply finding where the debris would fit best. The initial order of the mission was conserved, and the additional debris simply inserted between any two other debris. The complexity of this algorithm was linear of the number of debris already attached to the chain, and made for a very fast computation (usually on the order of a few milliseconds).

The second option was much more powerful but required factorially more time. It would fit the debris anywhere in the chain, but also reorder the chain. Because of the complexity of the constraints between debris, we adopted a brute force method for testing the reordering of the chain: in practice, all possible arrangements of the debris were tested. The second option would thus call the "simple fit" function $K!$ times for a mission visiting K debris. The algorithmic complexity was then quite punitive for long missions. In effect, this option was instantaneous for missions shorter than 6 debris, and would have taken more than a year for missions longer than 13 debris. For this reason, this option was never used until the very end of the competition: if the numerical integrator failed to realize a planned mission, this problematic mission was given back to the stitcher. We would remove a debris and try to stitch it back, with instructions to reorder the mission. The result would generally lower the required ΔV by a few hundreds of meters per second and allow the numerical integrator to make it into a real mission.

4.3 Finding the best stitching

The hard part of the stitching operation was to decide what to stitch where. At this higher level, the algorithm had a campaign of N missions, visiting K debris. Usually N was between 10 and 15 while the K would range between 75 and 95. Among the remaining 30-50 debris, which one would be best to stitch where and when?

Over the last two weeks of GTOC, several versions of the algorithm were created, each attempting to respond to the increasing leaderboard competition. The final version used involved a tree search with partial randomization. Instead of looking at a single campaign, we would create alternate scenarios, depending on which debris was stitched to which mission. The algorithm would be manipulating a number of scenarios (10-20) at any given step. From each of these, it would create many more (20-50) for the next generation by trying out tens of different stitchings to each of the manipulated scenarios. Finally, it would select which scenarios to keep among the best ones with an element of randomness. The best scenarios were determined as the ones with the lowest added ΔV (sometimes this ΔV would even be negative), as it usually output the longest chains and lowest numbers of debris left after the algorithm had run. Finally, at each step, the algorithm would check that all considered scenarios were indeed different from each other, as it was common for multiple scenarios to arrive at the same "best" solutions.

Since there were many scenarios coexisting at a given step in the search, we needed to quicken our computation of the stitching of each debris to each mission. To do so, we would create a stitching matrix that would follow a scenario and its children if they got selected, meaning that many scenarios could be explored at the same time for very little added computation time. This matrix had 123 rows and N columns, where N was the number of missions (from 12 to 15 usually). Each cell (k,n) of this matrix contained the information on the stitching of debris k to mission n. A stitching (cell) was recomputed if and only if a modification to the other missions, through a previous stitching, affected its feasibility.

Although the algorithm certainly dismissed many good solutions too early, it still allowed reaching unexpected solutions that would prove beneficial in the long term. It was however our impression that this exploration was only as good as the criterion used to rank the scenarios: "lowest added ΔV" was a good enough measure of optimality initially but it appeared quite clearly that it was too greedy an approach to capture the best solutions available. We were however unable to find a better measure of optimality. It is likely that, given a subset of each mission of the winning solution and only 40 debris left to place, this algorithm would still have missed

 DOI: 10.5281/zenodo.1139268

the full winning solution unless massive amounts of computation time, unrealistic for GTOC, had been dedicated to it.

In the end, this tree search with randomization would output nearly-complete campaigns. Usually there would still be 2-8 debris remaining. Although it may have been possible to stitch them through reordering, we did not have the computational capabilities to perform this operation for most missions. There was therefore the need to "finish" the campaign with additional missions.

The final solution submitted is shown in Figure 3. We see that, as expected, each mission is largely chosen based on the RAAN of each debris.

5 The Finisher

The algorithm known as the "finisher" aims to obtain a 123 debris campaign given the previously stitched chains and the remaining 2-8 pieces of debris. In a certain way, it repeats the first steps of computation using computational brute force; i.e., considering a big portion of the possible combinations between spares.

First, we find the remaining time gaps in which there are no missions scheduled. Then, applying the time margins to be held due to operational constraints, we now have the time windows to compute transfer maneuvers between spare debris.

We compute links between each pair of sp are debris. This ΔV is computed for varying transfer times, initial times, debris objects and time gaps. The ΔV is then stored and sorted to find the maximum number of possible pairs of single launches that can be combined to launch together.

We sort the sets of debris pairs by total ΔV of the respective maneuvers between the pairs found. The next step is to use the "stitcher" [4] again to reduce the number of missions of every set of pairs of debris by stitching the remaining spare debris to these pairs. The combination of missions that minimizes the total cost of the campaign is chosen.

As an example, a campaign of missions may have 7 spare debris after running the "stitcher" algorithm. Without further work, each of those 7 debris will require separate launches, which is very expensive. The cost of the spare debris can be reduced significantly by combining the debris into fewer launches. The first pass through the "finisher" may find 3 pairs of debris that can be launched together, bringing the number of launches down from 7 to 4 (3 launches which each visit two debris, and 1 single launch). The second pass through the "finisher" will reduce these 4 launches into perhaps 2 or 3 launches. While the cost of removing these debris is still high, it is greatly reduced from the cost of 7 single launches.

6 The Wiggler

After creating full campaigns of missions, the "wiggler" tool was used to slightly adjust the date at which each debris was visited. A nonlinear programming (NLP) problem was defined with the times between each debris rendezvous as the optimization variables. The analytical approximate ΔV of the whole mission was minimized, subject to operational constraints from the problem statement. This NLP was solved with MATLAB's fmincon solver, using the Interior Point method. The debris at the beginning of the mission and at the end of a mission were held constant to avoid inadvertently invalidating other missions, while the times between debris within the mission were allowed to vary between 7 and 29 days. Typically, the "wiggler" tool would adjust each date by a fraction of a day, and it would reduce the ΔV of each mission by 2-10%.

7 Final Optimization & Integration

A two-step algorithm was developed to transition from the approximate, analytical model described above to the fully-integrated solution for submission. During the final optimization and integration, the arrival times at each debris were held fixed, and the transfer between each pair of debris was considered separately. The algorithm, variables, and series of maneuvers are shown graphically in Figure 2.

The first step of the algorithm is to choose maneuver $\Delta\vec{V}_1$ at time t_1 so that the RAAN Ω and argument of latitude u of the spacecraft match the corresponding elements of debris $i+1$ at time t_2 (approximately 20 days later). Time t_1 is chosen to be the time when the spacecraft's argument of latitude is equal to zero. We never use the propulsion system to directly change Ω — rather, we change semimajor axis a, eccentricity e, and inclination i to indirectly change Ω by leveraging the natural dynamics. Maneuver $\Delta\vec{V}_1$ is defined in a VNC (velocity, normal, co-normal) frame, with components in the V and N directions. The component in the V direction immediately changes the semimajor axis and

DOI: 10.5281/zenodo.1139268

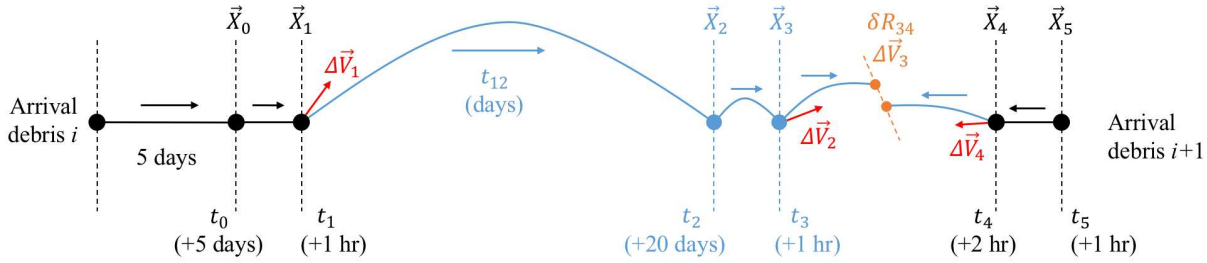

FIGURE 2. *Schematic showing the four maneuvers used in the final integration step to transfer from one debris to another. The dates of arrival at debris i and $i+1$ are held fixed. Black, straight lines indicate motion according to the debris' approximate equations of motion, while blue, curved lines indicate motion according to numerically propagating the Earth J_2 equations of motion. The times t_0 through t_5 are indicated, with typical values given of each relative to the previous time. The horizontal arrows indicate whether a segment was propagated forward or backward.*

eccentricity of the spacecraft's orbit, while the component in the N direction immediately changes the inclination. Only the V component affects the argument of latitude target, while both components have an indirect effect over time on the RAAN target because of the J_2 dynamics. For a given number of orbital revolutions, there is an exact solution for the V and N components of the maneuver that satisfy the constraints on Ω and u. After performing $\Delta\vec{V}_1$, the spacecraft coasts for up to approximately 25 days to state $\vec{X}_2$ at time t_2 (exactly three hours before the nominal rendezvous with debris $i+1$), which can be up to approximately 400 orbital revolutions. The number of orbital revolutions was chosen to minimize the magnitude of $\Delta\vec{V}_1$. Later insights revealed that it would be more optimal to choose the number of orbital revolutions to minimize the total ΔV, but it was not possible to implement this given the time constraints of the competition.

The second step of the algorithm is to choose maneuvers $\Delta\vec{V}_2$, $\Delta\vec{V}_3$, and $\Delta\vec{V}_4$ to adjust the spacecraft orbit's inclination, semimajor axis, argument of perigee, and true anomaly to rendezvous with debris $i+1$. To do this, we defined an optimization problem with 8 variables: t_{23} (the forward propagation time after t_2 until performing maneuver $\Delta\vec{V}_2$), t_{45} (the backward extra time from the nominal arrival time at debris $i+1$), the vector elements of $\Delta\vec{V}_2$, and the vector elements of $\Delta\vec{V}_4$. We propagate forward (with numerical integration) from time t_2 to time t_3 and perform maneuver $\Delta\vec{V}_2$ at time t_3. We also propagate backward (according to the debris dynamics) from t_5 to time t_4 and perform maneuver $\Delta\vec{V}_4$ at time t_4. We then shoot forward from time t_3 and backward from time t_4, constraining the position discontinuity $\delta\vec{R}_{34}$ to be $\vec{0}$ and defining the velocity discontinuity to be $\Delta\vec{V}_3$. MATLAB's fmincon optimizer is used with the Interior Point algorithm to minimize the total ΔV for maneuvers 2, 3, and 4 and remove the discontinuity at the midpoint of times 3 and 4.

It was found that the ΔV for the integrated solution and for the analytical estimate agreed well for transfers under 1 km/s. However, the algorithm had difficulty converging to an optimal solution when the total ΔV exceeded 1 km/s. We expect that further refinements to the final optimization algorithm could have improved the cost of these high-ΔV transfers, but we also acknowledge that these high-cost transfers could be removed entirely by better pruning techniques in the analytical search.

8 Discussion & Conclusions

Although the problem was definitely very hard to solve, we think that many constraints actually narrowed the strategy possibilities. For instance, by limiting the number of days between debris to 30, it was not possible to have a single mission waiting for an extended period of time. The total duration and the number of debris already imposed a fast rhythm of encounters (24 days on average), hence allowing to wait between debris would have added an interesting element or risk-reward: maybe a mission can get one more debris if it waits for 60 days, but is the time wasted really worth it?

 DOI: 10.5281/zenodo.1139268

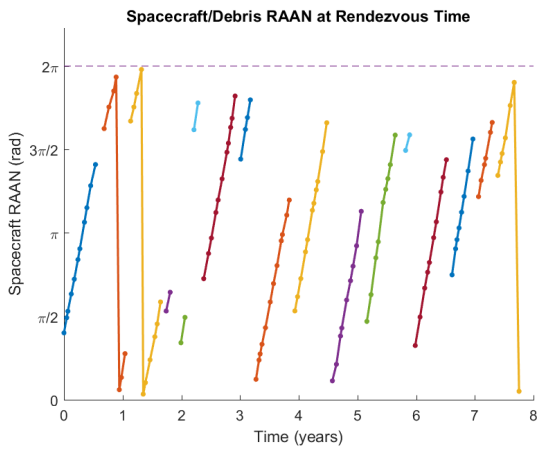

FIGURE 3. *The final solution of team CU Boulder. Each line series is a separate launch, with 17 total. The clear trend apparent in the RAAN of debris visited captures the driving dynamic of the problem.*

The real-time variation of the costs of each mission was an interesting element of the competition, but we feel it was ultimately a distraction that, if anything, only benefited those teams whose time availability happened to coincide with the competition. In the end, almost every team's best solution was submitted in the final hours, so there was no tangible advantage to submitting earlier. In a complete solution, all the missions are tightly related to each other, so it is not practical to submit one single mission early. Future competitions could make better use of this time-varying mechanic by making component parts of the full solution more separable from the whole. For example, if there were thousands of debris, and each team were to remove only some subset, then it would be feasible to design a single mission early in the competition that did not interfere with other missions designed later.

The leaderboard mechanic was very exciting for the competition, and also helped our team know what the ideal solution should look like. Towards the end of the competition, we were able to constrain our search space to be similar to the best solutions on the leaderboard.

9 Acknowledgements

We would like to thank the following individuals for their advice and help: Prof. Natasha Bosanac (CU Boulder), Dr. Jeffrey Parker (Advanced Space, LLC), and Prof. Christoffer Heckman (CU Boulder). This work was supported by a NASA Space Technology Research Fellowship.

Team CU Boulder is grateful to Dr. Dario Izzo for his work organizing a challenging, well-organized competition. Thank you.

References

[1] D. Izzo and M. Märtens. The Kessler Run: On the Design of the GTOC9 Challenge. *Acta Futura*, 11:11–24, 2018.