Draft version October 12, 2023
Typeset using LaTeX **twocolumn** style in AASTeX631

Solving the Traveling Telescope Problem with Mixed Integer Linear Programming

Luke B. Handley [1], Erik A. Petigura [1] and Velibor V. Mišić [2]

[1] *Department of Physics & Astronomy, University of California Los Angeles, Los Angeles, CA 90095, USA*
[2] *Decisions, Operations and Technology Management, Anderson School of Management, University of California Los Angeles, Los Angeles, CA 90095, USA*

ABSTRACT

The size and complexity of modern astronomical surveys has grown to the point where, in many cases, traditional human scheduling of observations are tedious at best and impractical at worst. Automated scheduling algorithms present an opportunity to save human effort and increase scientific productivity. A common scheduling challenge involves determining the optimal ordering of a set of targets over a night subject to timing constraints and time-dependent slew overheads. We present a solution to the 'Traveling Telescope Problem' (TTP) that uses Mixed-Integer Linear Programming (MILP). This algorithm is fast enough to enable dynamic schedule generation in many astronomical contexts. It can determine the optimal solution for 100 observations within 10 minutes on a modern workstation, reducing slew overheads by a factor of 5 compared to random ordering. We also provide a heuristic method that can return a near-optimal solution at significantly reduced computational cost. As a case study, we explore our algorithm's suitability to automatic schedule generation for Doppler planet searches.

Keywords: methods: observational

1. INTRODUCTION

Maximizing the scientific yield of expensive and often oversubscribed astronomical instrumentation requires meticulous planning of each night's observations. However, determining the optimal (or even near-optimal) sequence of observations is challenging and time consuming task. Schedulers must incorporate temporal accessibility windows while factoring in slew and acquisition overheads that can themselves be time-variable. In this paper, we refer to the task of determining the optimal ordering of a set of observations by a telescope as the 'Traveling Telescope Problem' or 'TTP' given its similarities to the 'Traveling Salesman Problem' or 'TSP'.

The scientific benefits of intelligently sequenced observations can be significant, especially for programs with many targets spread over the entire sky. As an example, the Doppler planet searches at the Keck-I telescope observe up to 100 targets per night (Howard et al. 2010). As we show below, a quasi-random sequence of 100 targets requires over 3 hours of slew during a 10 hour night while an optimized sequence can reduce this to 0.5 hours.

Automated solutions to the TTP offer a number of opportunities. As is the case for the TSP, a skilled human scheduler can generate a observation sequence that significantly outperforms a random ordering. However, such script generation takes significant human effort that could be devoted elsewhere. In addition, a number of ongoing and forthcoming surveys are or will be scheduled automatically. The Zwicky Transient Facility (ZTF; Bellm et al. 2019a) and the Legacy Survey of Space and Time (LSST; Ivezić et al. 2019) are two examples. Automated solutions to the TTP are necessary for automated surveys.

While a rich literature exists on the TSP, standard solutions do not directly transfer to the TTP for several reasons. First, targets are often only accessible for a fraction of the night either due to their position on the sky or scientific need for time-critical observations. Thus, a large fraction of the $N!$ target sequences are infeasible. Second, slew time between targets is *itself* a function of time. As an example, Figure 1 shows the trajectory of two targets above the Keck-I telescope atop Maunakea. Two targets cross the meridian north and south of zenith, respectively. Like most large telescopes, Keck-I moves in the altitude and azimuth directions and target slews are dominated by differences in azimuth. Figure 1 shows the variation in slew time over the course of the night, which grows as the targets first approach the meridian and then straddle the telescopes cable wrap limits.

Previous efforts have addressed the TTP under a number of simplifying assumptions. The ZTF scheduler divides the night into short intervals and solves a standard TSP while treating the set of accessible targets and target-to-target slew times as constant with in the interval (Bellm et al. 2019b). The *James Webb Space Telescope's* scheduling architecture (Giuliano & Johnston 2008), which is built upon the *Hubble Space Telescope's* SPIKE software (Johnston & Miller 1994), also treats target-to-target slew times as constant. At present, we are not aware of any global solutions to the TTP that capture both variable accessibility windows and slew overheads.

This paper is organized as follows. We describe the TTP in Section 2 and present a formulation using Mixed-Integer Linear Programming that can be solved to global optimally range of problem sizes. In Section 3 we conduct a suite numerical experiments to determine the performance and computational cost of our algorithm over various problem sizes. Section 4 discusses the limitations of our global approach and the potential of heuristic solutions. We conclude in Section 5.

2. FORMULATION OF TRAVELING TELESCOPE PROBLEM

2.1. *Problem Setup*

For a given set of targets and an observing interval, we seek the tour that completes all exposures in the shortest possible time. Targets must be accessible for at least part of the observing duration. The slew time between any pair of targets must be computed in advance, but the slew time may *itself* be a function of time. The formulation presented below was inspired by Sun et al. (2018) who developed a framework to optimize the profitability of parcel pickup and delivery with variable time windows and travel times.

2.2. *Slot Framework*

Following the TSP literature, we refer to targets to be traversed in the TTP as *nodes* since that work emphasizes that the solution is a directed graph connecting all targets. With a list of N targets to be scheduled, we define the total set of nodes to be $\{0, 1, ..., N, N+1\}$. The nodes 0 and $N+1$ are the anchoring start and end nodes; their location is arbitrary, i.e. not associated with a celestial source. Their purpose is explained in Section 2.4. Each node i has an associated exposure time $\tau_i^{\rm exp}$, and accessibility window $[t_i^{\rm e}, t_i^{\ell}]$. The values $t_i^{\rm e}$ and t_i^{ℓ} are the earliest and latest times the tour can depart node i, i.e. the time the observation concludes (see Figure 2). We summarize the symbols from the main body of this text in Appendix A.

Next, we break the scheduling interval into M subintervals or 'slots', within which travel time is treated as constant. M is a free parameter and impacts the computational load (see Section 3). The slots may have uneven durations if desired. The boundaries of the slots are $[w_m, w_{m+1}]$ where m indexes each slot.

We then construct the slew tensor $\tau_{ijm}^{\rm slew}$ that specifies the travel time between any two nodes during every slot (see Figure 1). This depends on the telescope slew speed in altitude and azimuth directions as well as cable-wrap considerations. For a concrete example, $\tau_{4,6,2}^{\rm slew}$ specifies the computed travel time between the $i = 4$ node and $j = 6$ node during the bounds of the slot $m = 2$.

2.3. *Decision Variables*

Our MILP formulation involves both binary and continuous variables. The following variables trace the flow through the nodes and the times of node departures:

- X_{ijm} is a binary variable equal to 1 if the tour traverses the arc from node i to node j during slot m and 0 otherwise.
- Y_i is a binary variable equal to 1 if the node i is entered at some point during the tour and 0 otherwise.
- t_i is a continuous variable denoting the departure time from node i.
- t_{ijm} is a continuous variable and equal to t_i if the tour departs the node i toward the node j during the time slot m, 0 otherwise.

The non-zero elements of X_{ijm} describe the tour. The tensor dot product of X_{ijm} and $\tau_{ijm}^{\rm slew}$

$$\sum_{i,j=1}^{N} \sum_{m=0}^{M-1} \tau_{ijm}^{\rm slew} X_{ijm} \tag{1}$$

is equal to the total travel time.

2.4. *Constraints*

Next, we introduce the following constraints:

Constraint 1. Tour must depart the starting node. We require that flow be non-zero from 0 to some arbitrary target node j during some slot m in time to begin the tour.

$$\sum_{j=1}^{N} \sum_{m=0}^{M-1} X_{0,j,m} = 1 \tag{2}$$

Constraint 2: Tour must conclude at the ending node. Similarly, we anchor the end of the tour with the

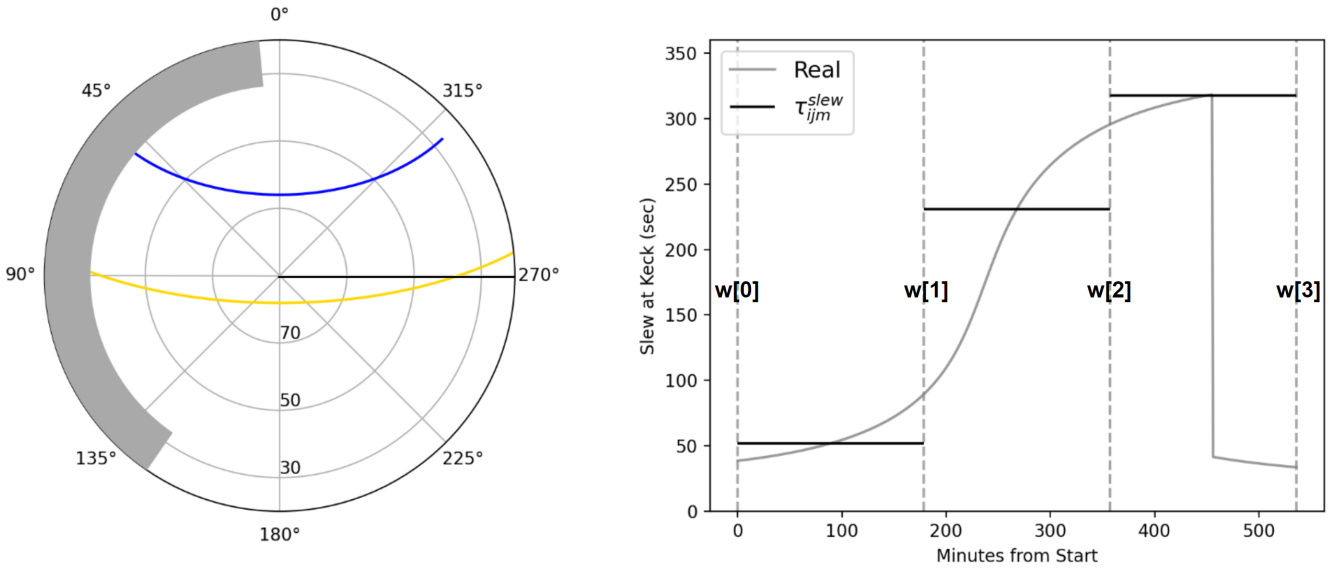

Figure 1. Time dependent slew overheads at Keck Observatory. The left plot shows the motion of two targets in the alt/az frame across the accessible region at Keck Observatory (latitude = 19.8° N). The top target sits at $\delta = 43.8°$, the bottom at $\delta = 11.7°$. As time progresses, the azimuthal separation dominates the slew and steadily increases to a maximum of over five minutes. Late in the night, the lower declination target crosses azimuth = 270°, which corresponds to the assumed telescope cable wrap limit. At this time, a direct slew becomes available, and the slew time drops significantly. For any two arbitrary targets i and j, τ_{ijm}^{slew} approximates the pairwise slew time within the bounds of each time slot as dictated by M. The right plot shows the interpolated slew curve (grey) with the travel time tensor τ_{ijm}^{slew} (black) over-plotted for the full night with $M = 3$. In this extreme case, τ_{ijm}^{slew} often misrepresents the slew time by several minutes in the second and third time slot.

end node $N+1$. We must traverse the arc $(i, N+1)$ from some arbitrary target node i.

$$\sum_{i=1}^{N} \sum_{m=0}^{M-1} X_{i,N+1,m} = 1 \tag{3}$$

Constraint 3: Y must indicate the visitation of a node. In the simplest possible case, we have a trivial solution traversing from 0 to $N+1$, stopping at a single target node along the way. The objective (see Section 2.5) will reward the visitation of additional nodes between these two constrained anchors. This requires first defining the variable Y to track whether nodes are being visited. Y_i will be activated if the tour flows from the node i during any slot in time, into any subsequent node j.

$$\sum_{j=1}^{N+1} \sum_{m=0}^{M-1} X_{ijm} = Y_i \quad \forall\, i = 0, \ldots, N \tag{4}$$

Constraint 4: A slew into any target node must be accompanied by a slew away from that node. Excluding the starting and ending nodes, we require that the slew into any target node k must be accompanied by a slew out of k.

$$\sum_{i=0}^{N} \sum_{m=0}^{M-1} X_{ikm} - \sum_{j=1}^{N+1} \sum_{m=0}^{M-1} X_{kjm} = 0 \quad \forall\, k = 1, \ldots, N \tag{5}$$

If X_{ijm} is 1 for any arc (i, k) in the sequence, it will also hold 1 for some (k, j) arc at an arbitrary time. If the left term is 0 (the tour never enters node k) then the tour will never traverse a departing arc originating at k. The chronology of these events will next be enforced using the time variable t.

Constraint 5: Define the selection variable t_{ijm}. Now we enforce the proper constraints to define the selection time variable t_{ijm} relative to our generic time variable t_i. By summing over all potential destinations j and time slots m, we recover the value stored in t_i.

$$t_i = \sum_{j=1}^{N+1} \sum_{m=0}^{M-1} t_{ijm} \quad \forall\, i = 0, \ldots, N \tag{6}$$

The second time variable t_{ijm} is critical for the following two constraints. It behaves like the product of t_i and X_{ijm}, but works within a linear program.

Constraint 6: Departure time from a node is greater than the departure from the previous node plus the slew and exposure time. Say we begin to traverse from one node to another on the arc (i, j)

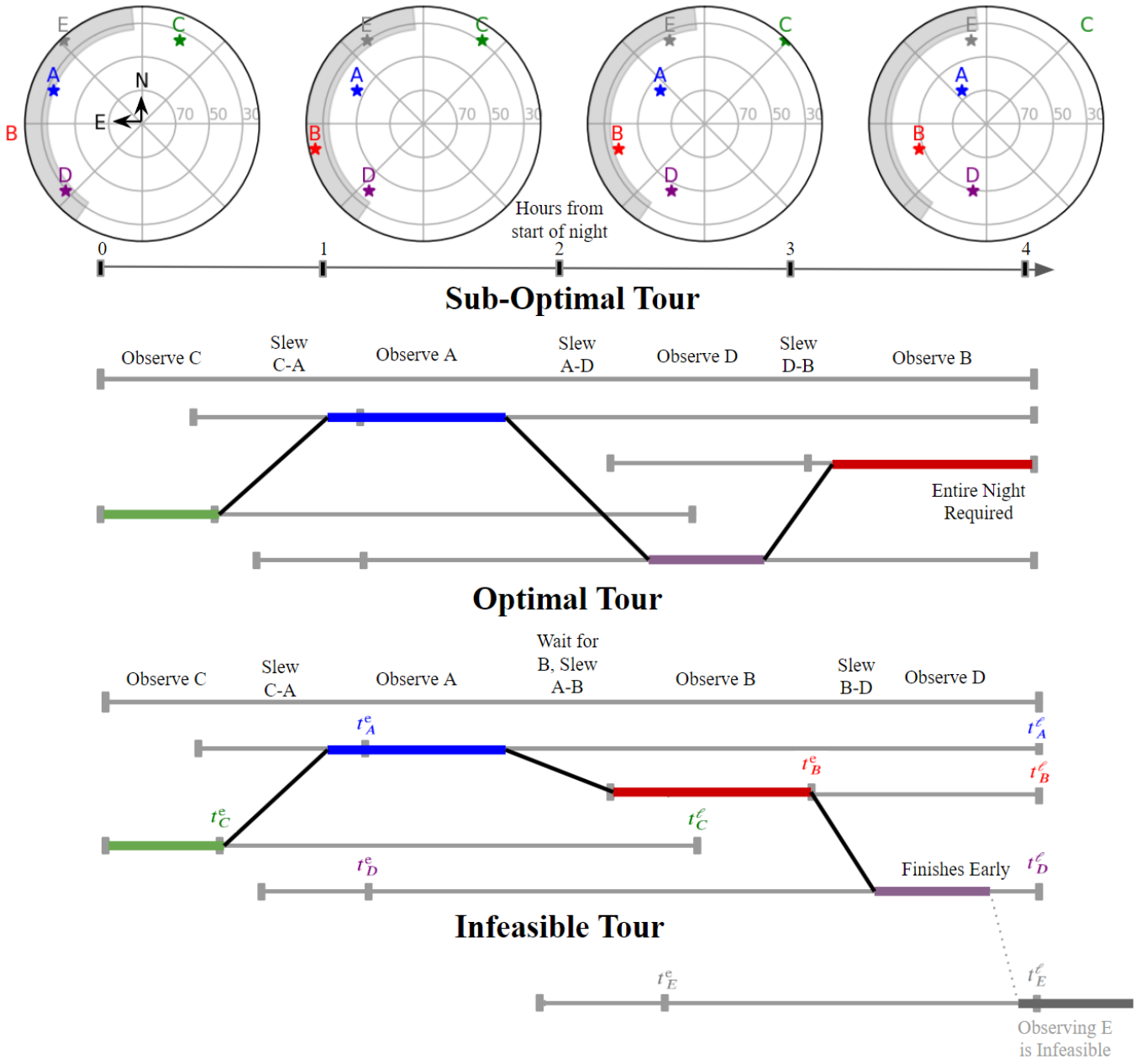

Figure 2. Schematic of sub-optimal, optimal, and infeasible tours in the travelling telescope problem. The four polar plots on top show the sky above Keck Observatory (latitude = +19 deg) in alt/az coordinates. During a duration of four hours, targets A–E move across the sky as the Earth rotates. The shaded region shows an inaccessible region of alt/az (here, the Keck-I Naysmth platform). We have exaggerated slew times to clearly illustrate the differences between the three tours. In the top **sub-optimal tour**, we include a horizontal timeline for targets A–D. The windows of accessibility are shown as vertical ticks. For example, the earliest the telescope can *complete* observations of target A is labeled with t^e_A while t^l_A, denotes the latest time. Note that t^e_A depends on both the rise time and exposure duration, and t^ℓ_i is the either the set time or the end of the observing interval, whichever comes first. This scheduler uses a greedy approach; i.e., once the observation A of completes, the telescope immediately slews to the unobserved target with the shortest slew time. The tour is feasible, but the entire observing duration is required in this example. The middle **optimal tour** is scheduled with a global optimization. In our example, the slew between A and D is so long that it is advantageous for the telescope to wait until target B rises to observe it, before proceeding to D. This is the optimal tour to observe targets A–D. Finally the bottom plot shows the inclusion of a fifth target E to illustrate an **infeasible tour**. There is not enough time to observe all targets.

within the bounds of the slot m. Before the telescope may depart from the next node j, a minimum amount of time must pass, equal to the slew experienced on the journey from i to j plus the exposure time at the new node, i.e. $\tau^{\rm slew}_{ijm} + \tau^{\rm exp}_{j}$. The minimum time value of our departure from j is the time that the previous node i was departed t_{ijm} plus this minimum 'passing time'. The inclusion of the variable X_{ijm} will ensure only the proper values of i and m are considered for the journey to the new node j.

$$t_j \geq \sum_{i=0}^{N} \sum_{m=0}^{M-1} \left(t_{ijm} + (\tau^{\rm slew}_{ijm} + \tau^{\rm exp}_{j}) X_{ijm} \right) \quad \forall\, j = 1, \ldots, N+1 \tag{7}$$

Constraint 7: Ensure departure times are consistent with slot bounds. We must enforce constraints on the departure times t_{ijm} using bounds of the time windows defined in w. If we exit the node i during the time slot m, then the departure time must be within the bounds of the slot window.

$$\begin{aligned} w_m X_{ijm} \leq t_{ijm} \leq w_{m+1} X_{ijm} \\ \forall\, i = 0, \ldots, N+1 \\ \forall\, j = 0, \ldots, N+1 \\ \forall\, m = 0, \ldots, M-1 \end{aligned} \tag{8}$$

Constraint 8: Departure times must respect node accessibility. Finally, we enforce the time window constraints on the departure times t_i for all the visited target nodes.

$$t^{\rm e}_i Y_i \leq t_i \leq t^{\ell}_i Y_i \quad \forall\, i = 1, \ldots, N \tag{9}$$

2.5. *Objective*

We seek to maximize the the number of scheduled exposures while minimizing total slew time

$$\text{Max} \left(\sum_{i=1}^{N} Y_i - C \sum_{i,j=1}^{N} \sum_{m=0}^{M-1} \tau^{\rm slew}_{ijm} X_{ijm}) \right) \tag{10}$$

Here, C is a small constant such that the slew penalisation (second term) is always less than unity. Notice that our anchor nodes do not contribute to the total slews with this summation convention, and are therefore used only for constraining the flow of the tour (a physical location need not be set, and the values in the first row and column of $\tau^{\rm slew}_{ijm}$ are set to 0).

The objective function does not require all nodes be visited. In similar TSP literature such as Sun et al. (2018), each target node may be assigned a scalar priority p_j included as a coefficient in the left summation term in Equation 10. In such a formulation, lowest priority targets are preferentially excluded when total completion is infeasible. We note that global optimality becomes less intuitive when targets have different numerical priorities.

In the TTP, the optimization will work to remove the targets that contribute *most* to the total slews in every case. The objective is clear: observe as many targets as possible as quickly as possible. We note that the formulation above describes a simple TTP where targets are observed once and no additional constraints are placed on the timing between observations. Appendix B describes small modifications to the algorithm presented above that can accommodate such constraints.

3. PERFORMANCE

3.1. *Numerical Experiments*

We evaluated our algorithm's ability to solve the TTP through a suite of simulated target lists. We explored problem complexity along the following three axes: number of targets N, number of time slots M, and duration of the scheduling interval D. We designed TTPs in the following manner:

1. We specified a random calendar night at Keck Observatory.

2. We selected an observing duration D.

3. We selected N stars from the California Legacy Survey (CLS; Rosenthal et al. 2021), a collection of 719 nearby stars that have been observed from Keck observatory for several decades as part of an extrasolar planet search. These stars comprise a good TTP test set because they are nearly uniformly distributed on the sky with declination $\delta \gtrsim -40$ deg and are thus observable for $\gtrsim 7$ months per year. When sampling the targets, we required they be accessible for more than 50% of duration D. We removed a few close binaries that have negligible slew speeds.

4. We modeled the slew rate of the Keck telescope as 1 deg per second in both altitude and azimuth directions.

5. We assigned uniform exposure times (in minutes) to all targets according to:

$$\tau^{\rm exp}_i = (D - 2N)/N \quad \forall\, i = 1, \ldots, N. \tag{11}$$

Setting the exposure times this way would completely fill the observing duration given a random ordering of targets with an average slew time of

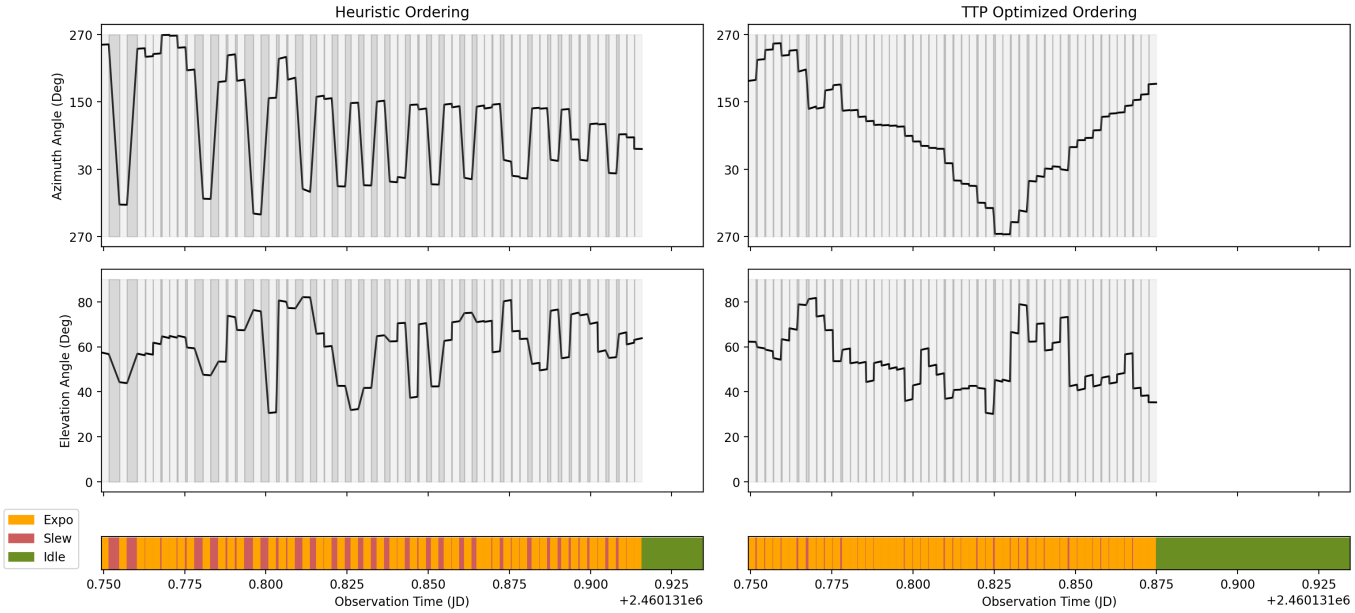

Figure 3. Comparison of a simple heuristic target ordering and a TTP optimized tour of $N = 50$ targets during a simulated half night. The former routine orders the targets by their increasing set times. Top row: the azimuthal coordinate of the simulated telescope in each case. Middle row: the elevation angle of the simulated telescope. Bottom row: time spent exposing, slewing, or idle. The targets have randomized time windows, but are all accessible for at least half of the scheduling duration. The worst case slew estimation from Equation 11 is 3.34 minutes per target. For the left (heuristic) column, the total slew time is 73 minutes, or an average of 91 seconds per slew, leaving 27 minutes idle. In the optimized (right) tour, the total time is a mere 13.8 minutes, or an average of 17.3 seconds per slew, resulting in 86 minutes idle (23 additional targets at the same pace).

120 sec or 2 min. For our experiments, slews in azimuth (which ranges from 0 to 360 deg) dominate over those in altitude (which ranges from 0 to 90 deg). The average distance between two randomly selected values between 0 and 360 is 120. We expect all solutions to the TTP to be significantly more efficient. Thus our experiments are feasible by construction and we report the improvement relative to this random ordering.

6. We selected M uniform slots within D.

7. We calculated the distance tensor τ^{slew}_{ijm} at the midpoint of each slot.

With the experiment specified, we solved the MILP described in Section 2 with one additional constraint.

Constraint 9: All target nodes must be visited.

$$Y_i = 1 \quad \forall\, i = 1, \ldots, N \tag{12}$$

The problem generation process described above in general results in TTP instances in which it is possible to observe all N targets. Given this, we modify the objective function described in equation (10) to focus on minimizing slews only, resulting in the following new objective function.

$$\text{Min} \left(\sum_{i,j=1}^{N} \sum_{m=0}^{M-1} \tau^{\text{slew}}_{ijm} X_{ijm} \right) \tag{13}$$

In our results in Section 3.2, we use TTP-Global to refer to the formulation defined in Section 2 with the objective function in equation (13) and constraint (12).

Before describing our grid-based exploration of problem complexity, we show one example solution to the TTP in Figure 3. We compared it with a simple heuristic solution for a 50 target observing sequence conducted over a half night to emulate a human generated script. In this heuristic, targets are observed in the order of their set times, i.e. the earliest setting target is observed first. Figure 3 shows graphically the slew overheads that TTP-Global eliminates through a more efficient ordering.

3.2. *Computational Results*

We test our formulation using different combinations of the number of targets N, the number of slots M and the duration/scheduling interval D. For the scheduling interval D, we consider quarter nights, half nights and full nights. For quarter nights, we vary N in $\{5, 10, 25\}$; for half nights, we vary N in $\{5, 10, 25, 50\}$; and for full nights, we vary N in $\{5, 10, 25, 50, 100\}$. For each combination of D and N, we vary M in $\{1, 3, 10\}$.

For each combination of N and D, we consider 10 randomly generated sets of targets, and consider the three different values of M, giving rise to a total of $(3+4+5)\times 3\times 10 = 360$ problem instances. We solved TTP-Global using Gurobi version 10.0.1, a state-of-the-art optimization suite that solves MILP problems using the branch-and-bound algorithm (Gurobi Optimization, LLC 2023). We used the Python programming language to generate the input data for TTP-Global and to formulate TTP-Global using the Gurobi Python API. We conducted our suite of numerical experiments on Amazon Elastic Compute Cloud (EC2), on a single instance of type `m6a.48xlarge` (AMD EPYC 7R13 processor, with 192 virtual CPUs and 768 GB of memory). For each experiment, we allocated 8 virtual CPUs and limited computation time to 1800 seconds.

For a few points of reference, a TTP with $N = 25$ targets and $M = 1$ involved an MILP with 1643 rows (constraints) and 1513 columns (variables), and the $M = 10$ case had 14765 rows and 14635 columns. A TTP with $N = 100$ targets had 21518 rows, 21013 columns for $M = 1$, and 208790 rows, 208285 columns for $M = 10$.

For each experiment, we recorded the following information:

- $\texttt{SlewTime}_\tau$: total slew time of the final schedule obtained from Gurobi, calculated using the discretized tensor τ^{slew}_{ijm}.
- $\texttt{RelRed}_\tau$: relative reduction in slew time of the final schedule compared to the randomly ordered value of $2N$; mathematically, it is defined as:

$$\texttt{RelRed}_\tau = \frac{2N - \texttt{SlewTime}_\tau}{2N} \times 100\%. \qquad (14)$$

- $\texttt{SlewTime}_{\texttt{real}}$: real slew time of the final schedule, calculated based on the t_i departure time values of the schedule.
- $\texttt{RelRed}_{\texttt{real}}$: the analog of $\texttt{RelRed}_{\texttt{real}}$ for the real slew time.
- `Runtime`: computation time
- Whether a provably optimal solution was found.
- Whether a feasible solution was found.

Table 1 summarizes these statistics for the ten experiments conducted at each combination of N, M and D. We report the number of experiments where Gurobi found a feasible solution, `NumFeas`, and a provably optimal solution `NumOptimal`. For the feasible set, we report the average values of $\texttt{SlewTime}_\tau$, $\texttt{RelRed}_\tau$, $\texttt{SlewTime}_{\texttt{real}}$, and $\texttt{RelRed}_{\texttt{real}}$. for each combination of N, M and D, where the average is taken over the `NumFeas` instances for which a feasible solution was found. For example, for $(\text{Half}, 50, 3)$, `NumFeas` is 8, indicating that Gurobi found a feasible schedule in only 8 out of the 10 instances; consequently, the value of 12.7 for $\texttt{SlewTime}_\tau$ is the average slew time over those 8 feasible schedules. We show the average `Runtime` and $\texttt{RelRed}_{\texttt{real}}$ values for different problem sizes in Figure 4.

4. DISCUSSION

We may draw a number of conclusions about the suitability of TTP-Global for the TTP from Table 1 and Figure 4. Gurobi generally found a feasible schedule when the number of targets $N \leq 25$. For $N \geq 50$ targets, our ability to find a feasible solution depended sensitively on the number of slots. Gurobi found a feasible schedule for all ten instances when $M = 1$, for some when $M = 3$, and for none when $M = 10$.

Not surprisingly, runtime was a strong function of N and M as can be clearly seen in Figure 4. For example, Gurobi found optimal solution for all $(D, N, M) = (\text{Full}, 25, 1)$ experiments with an average `Runtime` of 4.7 seconds. In contrast, the $(\text{Full}, 25, 10)$ experiments found no optimal solutions in the 1800 second time limit. For the largest $(\text{Full}, 100, 3)$ experiments, not even a feasible solution was found in the time limit. We find that the $M = 1$ case scales well into the realm of $N = 100$, solving even faster than the $N = 50$ case. While this goes against our initial intuition, we suspect this behavior is the result of parameter tuning by Gurobi to accommodate larger models.

When we do find a feasible schedule, it is significantly more efficient relative to the $2N$ baseline. For example, for the $(D, N, M) = (\text{Half}, 50, 3)$ experiments, the average $\texttt{SlewTime}_{\texttt{real}}$ is 22.4 minutes compared to the $2N = 100$ minute benchmark, a reduction of $\texttt{RelRed}_{\texttt{real}} = 77.6\%$. For the $(\text{Full}, 100, 1)$ set of instances, average $\texttt{SlewTime}_{\texttt{real}}$ is 36.1 minutes, which is a reduction of 82.0% relative to the $2N = 200$ minute benchmark. We note that in all cases, $\texttt{SlewTime}_\tau$ is less than $\texttt{SlewTime}_{\texttt{real}}$. For example, for the same $D = \text{Full}$, $N = 100$, $M = 1$ set of instances, $\texttt{SlewTime}_\tau$ is a mere 16.4 minutes, which is a reduction of $\texttt{RelRed}_\tau = 91.8\%$ relative to the 200 minute worst-case value. The difference between $\texttt{RelRed}_{\texttt{real}}$ and $\texttt{RelRed}_\tau$ stems from our piece-wise slew approximation, i.e. the choice of M for each model.

In the quarter night models, we find little improvement from using higher values of M, since the slews vary minimally with respect to time. For the longer durations, we see some benefit for higher M in the simpler $N < 25$ cases. For $N \geq 25$, the higher M models be-

D	N	M	NumFeas	NumOptimal	Runtime (s)	$\texttt{SlewTime}_\tau$ (min)	$\texttt{RelRed}_\tau$ (%)	$\texttt{SlewTime}_{\texttt{real}}$ (min)	$\texttt{RelRed}_{\texttt{real}}$ (%)
Quarter	5	1	10	10	0.0	4.1	59.1	5.3	47.0
Quarter	5	3	10	10	0.0	3.7	63.3	5.3	47.0
Quarter	5	10	10	10	0.1	3.5	64.5	5.1	48.7
Quarter	10	1	10	10	0.0	5.8	70.9	8.2	58.8
Quarter	10	3	10	10	0.3	5.4	73.2	8.1	59.4
Quarter	10	10	10	10	2.1	5.3	73.7	7.7	61.4
Quarter	25	1	10	10	0.5	8.1	83.8	13.0	74.0
Quarter	25	3	10	9	217.7	7.8	84.4	11.3	77.5
Quarter	25	10	8	3	1625.2	9.6	80.8	13.7	72.7
Half	5	1	10	10	0.0	4.9	50.8	7.5	25.4
Half	5	3	10	10	0.0	4.7	53.2	6.3	36.8
Half	5	10	10	10	0.1	4.4	55.5	6.3	37.1
Half	10	1	10	10	0.1	6.5	67.3	12.5	37.6
Half	10	3	10	10	0.5	5.8	70.8	9.8	51.1
Half	10	10	10	10	3.5	5.5	72.4	8.5	57.4
Half	25	1	10	10	9.7	9.2	81.5	15.5	69.0
Half	25	3	10	8	776.8	8.5	83.0	15.2	69.6
Half	25	10	8	0	1800.0	12.7	74.6	20.5	59.1
Half	50	1	10	5	954.7	12.1	87.9	22.8	77.2
Half	50	3	8	0	1800.0	12.7	87.3	22.4	77.6
Half	50	10	0	0	1800.1	–	–	–	–
Full	5	1	9	9	0.0	6.0	39.7	6.8	31.7
Full	5	3	9	9	0.0	5.3	47.0	7.3	27.0
Full	5	10	9	9	0.0	5.3	47.3	7.0	29.5
Full	10	1	9	9	0.1	8.3	58.6	13.1	34.6
Full	10	3	9	9	0.6	7.4	62.8	8.6	56.8
Full	10	10	9	9	1.6	6.5	67.6	8.9	55.3
Full	25	1	10	10	4.7	10.4	79.1	19.9	60.2
Full	25	3	10	5	1379.4	9.6	80.7	16.7	66.6
Full	25	10	10	0	1800.0	12.3	75.3	19.1	61.8
Full	50	1	10	7	1009.6	13.2	86.8	24.5	75.5
Full	50	3	3	0	1800.0	18.8	81.2	27.6	72.4
Full	50	10	0	0	1800.0	–	–	–	–
Full	100	1	10	8	619.8	16.4	91.8	36.1	82.0
Full	100	3	0	0	1800.1	–	–	–	–
Full	100	10	0	0	1800.0	–	–	–	–

Table 1. Computational results of TTP-Global for different values of D, N and M. *Note*: "–" indicates that the metric could not be calculated due to Gurobi not being able to obtain a feasible schedule for any of the ten instances. For D = Full, $N = 5$ and D = Full, $N = 10$, one instance was determined to be infeasible. The goal of our experiment is to devise target lists and exposure times that are feasible in the limit of large N, however for small N this is not strictly guaranteed.

come too complex to be solved to optimality in the time limit (leaving better slews in question), but the $M = 1$ models continue to demonstrate extremely efficient slews despite the higher expected variability.

In most cases, increasing the value of M did not demonstrate a significant advantage over the static models. For the high N models, the optimizer was often not able to construct a feasible solution for $M > 1$. For scheduling full nights of observations, the $M = 1$ case demonstrates dramatic improvement in slew times by up to a factor of 5, and increasing M provides more computational complexity than can be handled by our global algorithm in the time limit. For most cases, we

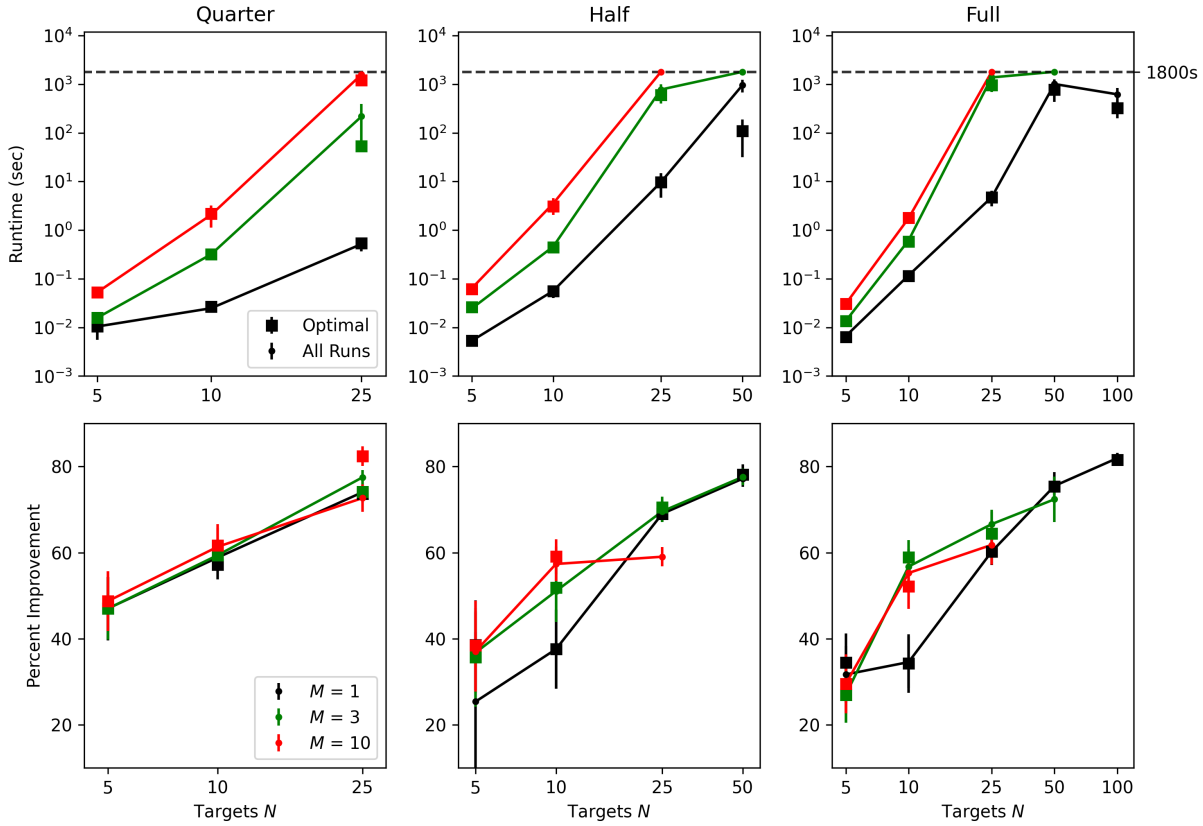

Figure 4. Top row: average value of `Runtime` for TTP-Global when scheduling each duration type D, with varying values of M. In the quarter night case, TTP-Global can schedule all 25 observations for $M < 10$, but struggles to reach optimality at $M = 10$. In the half night case, only the $M = 1$ model achieves optimality for $N = 50$. For the full night, TTP-Global handles $N = 100$ far better than expected for the static case in comparison to $N = 50$. Bottom row: average relative improvement $\texttt{RelRed}_{\texttt{real}}$ for each value of D across the model types. For all D and the respective maximum values of N, TTP-Global can reliably reduce slews by around a factor of 5. For small N, there is some benefit to using $M > 1$. For larger models, the risk of long slews in the $M = 1$ case has little impact on $\texttt{RelRed}_{\texttt{real}}$.

would recommend $M = 1$ be the standard due to its exceptionally fast runtime.

In our numerical experiments we attempted to solve TTP-Global to a provably optimal solution by branch-and-bound. For large numbers of targets or finely discretized slew tensors this approach may not be computationally tractable. We note that many heuristic solutions to the TSP have been developed that achieve near-optimal solutions. We suspect that analogous high-quality heuristic solutions exist for the TTP, which may be equipped to solve the $M > 1$ models for much higher N. We develop one such procedure in the Appendix for comparison with our global solution.

5. CONCLUSIONS

In this work, we addressed the challenge of determining the most efficient ordering of a set of exposures at a telescope. We formulated the Traveling Telescope Problem (TTP) as a mixed-integer linear program TTP-Global and solved it using a standard commercial optimizer for problem sizes as large as 100 targets in ~ 10 minutes using modest computational resources. The speed of TTP-Global means it can be run dynamically throughout the night and respond to real-time changes in target accessibility from weather. Further work is needed to develop algorithms that can solve the TTP for substantially larger sets of targets or substantially finer time-resolution in slew overheads. Local searches initialized with an heuristic solution may prove fruitful. We hope that algorithms like the one described here can assist or automate scheduling, save human effort and increase the scientific productivity of astronomical surveys.

L.H. & E.A.P. acknowledge support from the following sources: Heising-Simons Foundation Grant #2022-3832. V.V.M. acknowledges support from the UCLA Anderson School of Management. We are grateful for enlightening conversations with Eric Bellm.

Software: astropy (Astropy Collaboration et al. 2013, 2018), numpy (Harris et al. 2020), pandas (pandas development team 2020), gurobi (Gurobi Optimization, LLC 2023), matplotlib (Hunter 2007)

APPENDIX

A. VARIABLES

Table 2 lists the variables used in all preceding sections of this paper, along with their first usage:

Table 2. Symbols Used

Symbol	Definition	Section
C	Small normalization constant used in objective function. Ensures the scheduling of additional observations is prioritized	2.5
i, j, k	Indices for arbitrary nodes	2.2
m	Index for an arbitrary time slot	2.2
M	The number of time slots where slews are assumed constant.	2.2
N	The number of celestial targets assigned to the TTP	2.2
t_i^{e}	The earliest time value at which the node i can be departed based on observability constraints	2.2
t_i^{ℓ}	The latest time value at which the node i can be departed based on observability constraints	2.2
t_i	A continuous variable indicating the time value of the departure from node i	2.3
w_m	The time value at which the slot m begins	2.2
X_{ijm}	A binary decision variable indicating the flow state between nodes i, j during the time slot m. Holds 1 if a slew takes place, and 0 otherwise	2.3
Y_i	A binary decision variable that indicates the visitation of the node i. Holds 1 if the node is visited, and 0 otherwise	2.3
τ_i^{exp}	The exposure duration of the target node i	2.2
$\tau_{ijm}^{\mathrm{slew}}$	Travel time between the nodes i, j during the time slot m	2.2

B. VARIANTS OF THE TRAVELLING TELESCOPE PROBLEM

B.1. *Consecutive Targeting*

One may require two exposures i and i' be scheduled back-to-back, such as a science observation and a calibration observation. Such linked observations may be specified via two additional constraints:

$$Y_i + Y_{i'} = 2\left(\sum_{m=0}^{M-1} X_{ii'm} + \sum_{m=0}^{M-1} X_{i'im}\right) \tag{B1}$$

$$Y_i = Y_{i'}. \tag{B2}$$

The first constraint ensures that if both observations take place, the directed tour must pass directly from i to i' or vice versa. The second constraint ensures both observations or neither observation occur.

B.2. *Intra-Night Spacing Requirements*

One may wish to enforce a minimum interval between two exposures. A common example occurs in time-series monitoring where multiple observations of the same target occur during the same night, subject to a minimum separation. We accomplish this by letting N correspond to the total number of *exposures* to be collected across all targets. For

simplicity let us assign exposure indices such that exposures of the same target occur consecutively in the total exposure list $\{1, \ldots, N\}$. That is, if the target requires n^{exp} individual exposures, the node indices $\{\kappa, \kappa+1, \ldots, \kappa+n^{\text{exp}}-1\}$ correspond to that target for some value κ.

With the repeat observations specified, we enforce a minimum interval via the following constraint:

$$\sum_{j=1}^{N+1}\sum_{m=0}^{M-1} t_{ijm} \geq \sum_{j=1}^{N+1}\sum_{m=0}^{M-1} t_{i-1,j,m} + Y_i\tau^{\text{sep}}, \quad \forall\ i = \kappa+1, \ldots, \kappa+n^{\text{exp}}-1 \tag{B3}$$

Subsequent exposures of a given target must not end until at least τ^{sep} has passed since the previous exposure ended. For example, if the linked exposures of a target have index $i = 5$ and 6, exposure 6 may not end until at least τ^{sep} has passed since exposure 5 ended.

C. LOCAL SEARCH HEURISTIC

Here, we develop an alternative formulation to the TTP which uses a local search heuristic. We refer to this as TTP-LS to distinguish it from TTP-Global which searches the global solution space.

C.1. *Algorithm Description*

In the TTP, there are two sets of decisions that need to be made simultaneously. One is the sequence in which the targets will be visited. For example, with $N = 5$, we have to choose between $5 \to 1 \to 4 \to 2 \to 3$, $1 \to 3 \to 2 \to 4 \to 5$, and so on. The other set of decisions involves the timing slews. This involves deciding a (continuous) time t_i within the observing duration D to slew and the corresponding slot m. Even with a fixed sequence exposures, this is a non-trival task. As a result, making both sets of decisions under the umbrella of a single MILP formulation is computationally demanding.

This suggests an alternate approach to the TTP that decouples the sequence decision from the timing decision. Suppose that the sequence of targets is fixed. When should the telescope slew in order to minimize slew times? Let σ denote the sequence of targets, which is a bijective function $\sigma : \{1, \ldots, N\} \to \{1, \ldots, N\}$, and let the minimum total slew time be denoted by the function F, so that $F(\sigma)$ is the minimum total slew time that one would obtain from following the sequence σ. Let Σ denote the set of all such sequences. For example, for $N = 5$ targets and the sequence $5 \to 1 \to 4 \to 2 \to 3$, the corresponding σ is

$$\begin{aligned}\sigma(1) &= 5\\ \sigma(2) &= 1\\ \sigma(3) &= 4\\ \sigma(4) &= 2\\ \sigma(5) &= 3\end{aligned}$$

The TTP can then be abstractly formulated as

$$\min_{\sigma \in \Sigma} F(\sigma),$$

which is an optimization problem over sequences in Σ. As written, this is not a problem that can be readily provided to any commercial solver, but because of the discrete nature of Σ, it can potentially be solved using local search. Let $z \in \{1, \ldots, N\}$ and $z' \in \{1, \ldots, z-1, z+1, \ldots, N\}$ be positions in the sequence, and let $\sigma^{z \leftrightarrow z'}$ denote the sequence obtained by swapping the targets in positions z and z'; that is, $\sigma^{z \leftrightarrow z'}$ is the unique sequence such that

$$\begin{aligned}\sigma(z) &= \sigma^{z \leftrightarrow z'}(z'),\\ \sigma(z') &= \sigma^{z \leftrightarrow z'}(z),\\ \sigma(z'') &= \sigma^{z \leftrightarrow z'}(z'') \quad \forall z'' \in \{1, \ldots, N\} \setminus \{z, z'\}.\end{aligned}$$

Let $\mathcal{N}_z(\sigma)$ denote the set of neighboring sequences of σ obtained by swapping the target at position z with a target at any other position:

$$\mathcal{N}_z(\sigma) = \left\{ \sigma' \in \Sigma \;\middle|\; \begin{array}{l} \sigma' = \sigma^{z \leftrightarrow z'} \\ \text{for some } z' \in \{1, \ldots, z-1, z+1, \ldots, N\} \end{array} \right\}$$

With this definition, our local search algorithm can be formally described as Algorithm 1.

In words, we begin from some initial sequence σ. We use $\mathcal{U}$ to denote the set of sequence positions which have not yet tried to modify. As long as there is at least one sequence position we have not tried to change, we pick a sequence position z, and calculate the best neighboring sequence σ^* obtained by swapping the target at position z with the target at any other position. If the objective value F' of the best neighboring sequence improves on the current objective value $F(\sigma)$, we replace σ with σ^*, and we reset $\mathcal{U}$ to be the set of all positions. Otherwise, if we do not make improvement in an iteration of the while loop, then $\mathcal{U}$ will be reduced by one member. If N such iterations occur, then $\mathcal{U}$ will be empty, and we will have ascertained that there is no neighboring sequence we can move to in order to reduce the objective value; in other words, σ is a locally optimal sequence. We note that this heuristic is similar to the 2-OPT heuristic (Croes 1958) for the classical TSP problem, which involves eliminating two edges in a

Algorithm 1 Pseudocode of local search procedure.

Require: Initial sequence $\sigma \in \Sigma$.
Set $\mathcal{U} \leftarrow \{1, \ldots, N\}$.
while $|\mathcal{U}| > 0$ **do**
  Select $z \in \mathcal{U}$; set $\mathcal{U} \leftarrow \mathcal{U} \setminus \{z\}$
  Set $\sigma^* \leftarrow \arg\min_{\sigma' \in \mathcal{N}_z(\sigma)} F(\sigma')$
  Set $F' \leftarrow \min_{\sigma' \in \mathcal{N}_z(\sigma)} F(\sigma')$
  if $F' < F(\sigma)$ **then**
    Set $\sigma \leftarrow \sigma^*$,
    Set $\mathcal{U} \leftarrow \{1, \ldots, z-1, z+1, \ldots, N\}$
  end if
end while
return Locally optimal sequence σ

TSP tour and reconnecting the tour so that the edges are swapped. The main difference comes from the function F, which calculates the minimum total slew time when one assigns the targets in the sequence to slots optimally.

Before this algorithm can be deployed, we must specify how to compute $F(\sigma)$. The function value $F(\sigma)$ for a fixed sequence σ can be calculated by solving a MILP. While this MILP shares some similarities with the TTP problem described in Section 2, it is a simpler because target sequence is fixed and "baked in" to the optimization problem. This smaller MILP is a subroutine in Algorithm 1. We provide further details on this integer program in Section C.2.

We must also consider sequences of targets that are infeasible. In some cases, for a fixed sequence σ of targets, it may not be possible to make the timing decisions and the slot decisions in way that respects the accessibility windows and slot bounds. In such cases, the MILP that defines the function $F(\cdot)$ will be infeasible. We can extend the definition of $F(\cdot)$ so that $F(\sigma) = +\infty$ if the corresponding MILP is infeasible for sequence σ; since Algorithm 1 is always choosing the neighboring sequence with the lowest value of F, this ensures that Algorithm 1 will never replace the current sequence with one that is infeasible.

However, even with this fix, one problem that still remains is if the initial sequence σ and all neighboring sequences of that initial sequence are infeasible. In this case, Algorithm 1 will not return a feasible sequence, as it will simply terminate with the current sequence. This is a serious issue, because it is not straightforward to identify a sequence of targets for which the TTP problem constraints can be perfectly satisfied. In Section C.3, we present a feasibility heuristic for identifying such a sequence.

C.2. *Calculating Minimum Total Slew Time for a Fixed Sequence of Targets*

As noted in the previous section, a key component of Algorithm 1 is the function F, which maps a sequence σ to a minimum slew time $F(\sigma)$. We use z to denote the index of a position in this sequence σ. For targets, z will range in $\{1, 2, \ldots, N\}$. With a slight abuse of notation, we will use $z = 0$ to denote the start node of the telescope, and assume that $\sigma(0) = 0$; similarly $\sigma(N+1) = N+1$ at the final node. Thus, z can take any value in $\{0, 1, \ldots, N+1\}$.

Let $Y^{\text{at}}_{z,m}$ be a binary decision variable that is 1 if the telescope departs the target at position $z \in \{0, 1, \ldots, N+1\}$ in the sequence in slot m, and 0 otherwise. Let $Y^{\text{by}}_{z,m}$ be a binary decision variable that is 1 if the telescope reaches slot m by position $z \in \{0, 1, \ldots, N+1\}$ in the sequence and 0 otherwise. Let t_z denote the departure time of the telescope from the node at position z. The function F is obtained by solving the following MILP:

$$\text{minimize} \quad \sum_{z=1}^{N} \sum_{m=0}^{M-1} \tau^{\text{slew}}_{\sigma(z),\sigma(z+1),m} Y^{\text{at}}_{z,m} \tag{C4a}$$

$$\text{subject to} \quad Y^{\text{by}}_{0,0} = 1, \tag{C4b}$$

$$Y^{\text{by}}_{z,m} \leq Y^{\text{by}}_{z+1,m}, \quad \forall\, z = 0, 1, \ldots, N,\ m = 0, 1, \ldots, M-1, \tag{C4c}$$

$$Y^{\text{by}}_{z,m+1} \leq Y^{\text{by}}_{z,m}, \quad \forall\, z = 0, 1, \ldots, N+1,\ m = 0, 1, \ldots, M-2, \tag{C4d}$$

$$Y^{\text{at}}_{z,m} = Y^{\text{by}}_{z,m} - Y^{\text{by}}_{z,m+1}, \quad \forall\, z = 0, 1, \ldots, N+1,\ m = 0, 1, \ldots, M-2, \tag{C4e}$$

$$Y^{\text{at}}_{z,M-1} = Y^{\text{by}}_{z,M-1}, \quad \forall\, z = 0, 1, \ldots, N+1, \tag{C4f}$$

$$t_z \geq t_{z-1} + \sum_{m=0}^{M-1} \tau^{\text{slew}}_{\sigma(z-1),\sigma(z),m} \cdot Y^{\text{at}}_{z-1,m} + \tau^{\text{exp}}_{\sigma(z)}, \quad \forall z = 1, 2, \ldots, N+1, \tag{C4g}$$

$$t_z \geq \sum_{m=0}^{M-1} w_m \cdot Y^{\text{at}}_{z,m}, \quad \forall z = 0, 1, \ldots, N+1, \tag{C4h}$$

$$t_z \leq \sum_{m=0}^{M-1} w_{m+1} \cdot Y^{\text{at}}_{z,m}, \quad \forall z = 0, 1, \ldots, N+1, \tag{C4i}$$

$$t_z \geq t^e_{\sigma(z)}, \quad \forall\ z = 0, 1, \ldots, N+1, \tag{C4j}$$

$$t_z \leq t^\ell_{\sigma(z)}, \quad \forall\ z = 0, 1, \ldots, N+1, \tag{C4k}$$

$$Y^{\text{by}}_{z,m} \in \{0,1\}, \quad \forall\ z = 0, 1, \ldots, N+1,\ m = 0, 1, \ldots, M-1, \tag{C4l}$$

$$Y^{\text{at}}_{z,m} \in \{0,1\}, \quad \forall\ z = 0, 1, \ldots, N+1,\ m = 0, 1, \ldots, M-1. \tag{C4m}$$

In order of appearance, the constraints have the following meaning. Constraint (C4c) requires that if we have reached slot m by position z, then it must be the case that we have reached slot m by position $z+1$. Constraint (C4d) requires that if we have reached slot $m+1$ by position z, then we must have reached slot m by position z. Constraints (C4e) and (C4f) link the Y^{by} and Y^{at} variables; constraint (C4e) means that we are in slot m at position z if and only if we have reached slot m by position z ($Y^{\text{by}}_{z,m} = 1$) and have not reached slot $m+1$ by position z ($Y^{\text{by}}_{z,m+1} = 0$). Constraint (C4f) similarly requires that we are in slot $M-1$ at position z if and only if we have reached slot $M-1$ by position z. Constraint (C4g) requires that the departure time from the target in position z is at least the slew time that is realized departing from the target in slot $z-1$ plus the exposure time of the target in position z. Constraints (C4h) and (C4i) ensure that the departure time of each position z is within the lower and upper bounds of that position's assigned slot, while constraints (C4j) and (C4k) ensure that the departure time from each position $z \in \{1, \ldots, N\}$ is within the rise and set times for the target in that position ($t^e_{\sigma(z)}$ and $t^\ell_{\sigma(z)}$ respectively). The last two constraints enforce that the Y^{at} and Y^{by} variables are binary.

The MILP problem (C4) is essentially the TTP problem of Section 2, restricted to a particular sequence σ. Essentially, this formulation decides when each target's departure time will be and to what slots the different positions in the sequence will be allocated. Importantly, the sequence of targets is not a decision variable as it is in the original TTP model; it is a fixed input that is provided by the user.

Many of the constraints are direct analogs of constraints that appear in the TTP-Global formulation. For example, (C4j) and (C4k) model the rise and set time constraints for each target in each position, mirroring constraint (9) of the TTP MILP. As another example, (C4i) and (C4h) model the lower and upper bounds of each slot, similarly to constraint (8). Note that because the sequence of targets is fixed, many of the constraints from the full TTP MILP can be simplified, and other decisions, such as the departure times and in which slot each target is being departed from, can be expressed more efficiently using different decision variables. Specifically, the decision of which slot each position is assigned to is captured by the $Y^{\text{by}}_{z,m}$ decision variables. Here, we remark that these variables are an example of the incremental encoding technique in integer programming, which enhances the efficiency of branching in the branch-and-bound algorithm that is the cornerstone of integer programming solvers. We refer interested readers to the review paper of Vielma (2015), and to Bertsimas et al. (2011), Bertsimas et al. (2019) and Mišić (2020) for examples of applications of this technique in air traffic control, vehicle routing and optimization over trained machine learning models.

As a result, problem (C4) is much easier to solve than the full TTP MILP. We found that that Gurobi could determine the exact optimal solution to this problem in under a second with a single thread.

C.3. *Feasibility Algorithm*

The local search approach described above may fail if it initialized at an infeasible sequence whose neighbors are also all infeasible. In order for Algorithm 1 to return a sequence that can be implemented, the initial sequence must be one for which the minimum slew problem (C4) is feasible. Given the combinatorical complexity of the TTP, it is unlikely that one would randomly select a target sequence that would result in problem (C4) being feasible.

Thus motivated, we present in this section an algorithm that, starting from any sequence, seeks to return a feasible sequence. We note that this algorithm is a heuristic, and is not guaranteed to succeed. Nevertheless, our numerical results in Section C.4 indicate that this heuristic is generally very effective.

At a very high level, the algorithm we will propose resembles our local search procedure, Algorithm 1, in that it starts from a sequence σ and makes moves to neighboring sequences. The key difference is the objective function

that is used. Instead of using the function F, the feasibility algorithm first seeks to locally optimize a function G_1, followed by a function G_2. The function $G_1(\sigma)$ measures, for the sequence σ, the smallest violation of the slot window constraints (C4h) and (C4i) that can be attained when we choose the departure times and the slots to which each target is assigned to. This violation is a nonnegative quantity; a positive value implies that we are unable to satisfy all of the constraints, i.e., at least one constraint in constraint sets (C4h) and (C4i) is violated. A value of zero implies that all of the constraints in the two constraint sets are satisfied. The function $G_2(\sigma)$ similarly measures the smallest possible violation of the visibility constraints (C4j) and (C4k) when we choose the departure times and the slots. Again, a positive value implies that at least one constraint in the constraint sets (C4j) and (C4k) is violated, while a value of zero implies we can satisfy all of the constraints defined by (C4j) and (C4k).

G_1 is defined by the following MILP:

$$\text{minimize} \quad \sum_{z=0}^{N+1} \epsilon_z^{\texttt{slot,e}} + \sum_{z=0}^{N+1} \epsilon_z^{\texttt{slot},\ell} \tag{C5a}$$

$$\text{subject to} \quad Y_{0,0}^{\text{by}} = 1, \tag{C5b}$$

$$Y_{z,m}^{\text{by}} \le Y_{z+1,m}^{\text{by}}, \quad \forall\ z = 0,1,\ldots,N,\ m = 0,1,\ldots,M-1, \tag{C5c}$$

$$Y_{z,m+1}^{\text{by}} \le Y_{z,m}^{\text{by}}, \quad \forall\ z = 0,1,\ldots,N+1,\ m = 0,1,\ldots,M-2, \tag{C5d}$$

$$\tag{C5e}$$

$$Y_{z,m}^{\text{at}} = Y_{z,m}^{\text{by}} - Y_{z,m+1}^{\text{by}}, \quad \forall\ z = 0,1,\ldots,N+1,\ m = 0,1,\ldots,M-2, \tag{C5f}$$

$$Y_{z,M-1}^{\text{at}} = Y_{z,M-1}^{\text{by}}, \quad \forall\ z = 0,1,\ldots,N+1, \tag{C5g}$$

$$t_z \ge t_{z-1} + \sum_{m=0}^{M-1} \tau_{\sigma(z-1),\sigma(z),m}^{\text{slew}} \cdot Y_{z-1,m}^{\text{at}} + \tau_{\sigma(z)}^{\text{exp}}, \quad \forall z = 1,2,\ldots,N+1, \tag{C5h}$$

$$t_z \ge \sum_{m=0}^{M-1} w_m \cdot Y_{z,m}^{\text{at}} - \epsilon_z^{\texttt{slot,e}}, \quad \forall z = 0,1,\ldots,N+1, \tag{C5i}$$

$$t_z \le \sum_{m=0}^{M-1} w_{m+1} \cdot Y_{z,m}^{\text{at}} + \epsilon_z^{\texttt{slot},\ell}, \quad \forall z = 0,1,\ldots,N+1, \tag{C5j}$$

$$t_z \ge t_{\sigma(z)}^{e} - \epsilon_z^{\text{e}}, \quad \forall\ z = 0,1,\ldots,N+1, \tag{C5k}$$

$$t_z \le t_{\sigma(z)}^{\ell} + \epsilon_z^{\ell}, \quad \forall\ z = 0,1,\ldots,N+1, \tag{C5l}$$

$$Y_{z,m}^{\text{by}} \in \{0,1\}, \quad \forall\ z = 0,1,\ldots,N+1,\ m = 0,1,\ldots,M-1, \tag{C5m}$$

$$Y_{z,m}^{\text{at}} \in \{0,1\}, \quad \forall\ z = 0,1,\ldots,N+1,\ m = 0,1,\ldots,M-1, \tag{C5n}$$

$$\epsilon_z^{\text{e}}, \epsilon_z^{\ell}, \epsilon_z^{\texttt{slot},\ell}, \epsilon_z^{\texttt{slot,e}} \ge 0, \quad \forall z = 0,1,\ldots,N+1. \tag{C5o}$$

Observe that this integer program is similar to the minimum slew problem (C4), except that we now allow for violations of the constraints (C4h), (C4i), (C4j) and (C4k). The main modifications are as follows. First, observe that in addition to the decision variables of problem (C4), problem (C5) includes the decision variables $\epsilon_z^{\text{e}}, \epsilon_z^{\ell}, \epsilon_z^{\texttt{slot},\ell}, \epsilon_z^{\texttt{slot,e}}$, which measure how much the rise, set, slot upper bound and slot lower bound constraints can be violated in time.

Second, observe that constraints (C5i) - (C5l) resemble constraints (C4h) - (C4k), except that the new $\epsilon_z^{\text{e}}, \epsilon_z^{\ell}, \epsilon_z^{\texttt{slot},\ell}, \epsilon_z^{\texttt{slot,e}}$ appear. These new decision variables are bounded from below by zero and unbounded from above, so they effectively allow the optimizer to choose to not satisfy these constraints. For example, in the constraint $t_z \le t_{\sigma(z)}^{\ell} + \epsilon_z^{\ell}$, for whatever value of t_z we can always make the constraint satisfied by setting ϵ_z^{ℓ} to be equal to any value greater than $\max\{0, t_z - t_{\sigma(z)}^{\ell}\}$; as a concrete example, if $t_z = 120$ and $t_{\sigma(z)}^{\ell} = 80$, then any $\epsilon_z^{\ell} \ge \max\{120 - 80, 0\} = 40$ will satisfy the constraint.

Lastly, observe that the objective function is equal to the sum of the $\epsilon_z^{\texttt{slot},\ell}$ and $\epsilon_z^{\texttt{slot,e}}$ variables. Thus, the optimizer seeks to find the assignments of sequence positions to slots and the departure times so as to minimize how much the slot lower and upper bound constraints from problem (C4) are violated. Observe that if the optimal objective value of problem (C5) is zero, then we have found a partially feasible solution to problem (C4) that satisfies all of the constraints, and in particular constraints (C4h) and (C4i), with the possible exception of the rise and set time

constraints (C4j) and (C4k). Importantly, note that no matter what sequence σ one chooses, problem (C5) is always feasible.

We now define the function G_2. The function G_2 is defined as the objective value of the following integer program, which is

$$\begin{aligned}
\text{minimize} \quad & \sum_{z=0}^{N+1} \epsilon_z^{\mathrm{e}} + \sum_{z=0}^{N+1} \epsilon_z^{\ell} && \text{(C6a)}\\
\text{subject to} \quad & \text{constraints (C5b) - (C5o)}, && \text{(C6b)}\\
& \epsilon_z^{\texttt{slot,e}} = 0, \quad \forall z = \{0, 1, \ldots, N+1\}, && \text{(C6c)}\\
& \epsilon_z^{\texttt{slot},\ell} = 0, \quad \forall z = \{0, 1, \ldots, N+1\}. && \text{(C6d)}
\end{aligned}$$

Problem (C6) has the same structure as problem (C5), except that we force the violation variables $\epsilon_z^{\texttt{slot,e}}$ and $\epsilon_z^{\texttt{slot},\ell}$ to zero; thus, we no longer allow for violations of the slot bound constraints (C4h) and (C4i). We do still allow for violations of the rise and set time constraints (C4j) and (C4k). The objective function measures how much the rise and set time constraints are violated. Observe that if the objective value of problem (C6) is zero, then we have exactly verified that the minimum slew time MILP (C4) is feasible.

With problems (C5) and (C6) defined, we can now define the feasibility algorithm, which we provide in Algorithm 2. This algorithm works by first performing local search using the function G_1. If the local optimum is such that the value of G_1 is positive, then the algorithm terminates and returns that the problem is infeasible. Otherwise, if the value of G_1 is zero, then we continue to the next phase, in which we perform local search using the function G_2. If the local optimum of G_2 is positive, then the algorithm again terminates and returns that the problem is infeasible. Otherwise, if the value of G_2 is zero, then we have identified a feasible sequence. Note that Algorithm 2 is a heuristic and does not provably verify that problem (C4) is infeasible. If it returns "Problem is infeasible", it may not be the case that the minimum slew problem is actually infeasible.

C.4. *Computational results for local search heuristic*

We now present our results on our heuristic approach described above. We tested our approach on the same collection of 360 experiments described in Section 3.2, and compute the same result metrics. We tested two variants of our local search procedure:

- TTP-LS-1: Here, we execute our overall algorithm from a single random starting point, which we obtain by drawing a sequence σ uniformly at random from all possible $N!$ sequences.
- TTP-LS-10: In the second variant, we execute our overall algorithm from ten randomly generating starting points, each of which is a uniformly randomly generated sequence, and retain the best solution obtained over the ten repetitions.

With both TTP-LS-1 and TTP-LS-10, we impose a time limit of 600 seconds on the total run time. In the most extreme case, TTP-LS-1 will require 600 seconds, while TTP-LS-10 will require $600 \times 10 = 6000$ seconds. In both variants, the functions G_1 and G_2 (see Appendix C.3) and F (see Appendix C.2) are computed by solving the corresponding MILPs using Gurobi with a single thread. We again implement our procedure in Python and run our experiments on the same Amazon EC2 instance described in Section 3.2.

Table 3 summarizes the results for TTP-LS-1 and Figure 5 shows run time and slew efficiency for different problem sizes. TTP-LS-1 exhibits favorable performance in terms of computation time; in most cases, TTP-LS-1 terminates with a locally optimal solution within 600 seconds. The only exception is the $(D, N, M) = (\text{Full}, 100, 10)$ set of instances. Note that TTP-LS-1 resulted in a feasible schedule in all but seven of the 360 instances; importantly, TTP-LS-1 finds a feasible schedule in all of the instances for parameter combinations for which the TTP-Global MILP fails (for example, for $(\text{Full}, 100, 10)$, TTP-LS-1 produces a feasible schedule in all ten instances in 600 seconds, whereas TTP-Global fails to find a feasible schedule in all ten instances with 1800 seconds of computation. Of those seven instances in which TTP-LS-1 did not find a feasible schedule, six are the same instances which were determined to be infeasible by TTP-Global, and in one instance, the feasibility procedure (Algorithm 2) failed to identify a feasible solution, despite the fact that the instance does admit a feasible solution based on running TTP-Global. Lastly, in terms of solution quality, the total slew time, as measured by $\texttt{SlewTime}_\tau$ and $\texttt{SlewTime}_{\texttt{real}}$, compares favorably to the worst-case bound of $2N$.

Algorithm 2 Pseudocode of feasibility procedure.

Require: Initial sequence $\sigma \in \Sigma$.
{Phase 1: Minimization of G_1 (violation of slot lower and upper bound constraints)}
Set $\mathcal{U} \leftarrow \{1, \ldots, N\}$.
while $|\mathcal{U}| > 0$ **do**
  Select $z \in \mathcal{U}$; set $\mathcal{U} \leftarrow \mathcal{U} \setminus \{z\}$.
  Calculate $\sigma^* \leftarrow \arg\min_{\sigma' \in \mathcal{N}_z(\sigma)} G_1(\sigma')$.
  Calculate $G_1' \leftarrow \min_{\sigma' \in \mathcal{N}_z(\sigma)} G_1(\sigma')$.
  if $G_1' < G_1(\sigma)$ **then**
    Set $\sigma \leftarrow \sigma^*$.
    Set $\mathcal{U} \leftarrow \{1, \ldots, z-1, z+1, \ldots, N\}$.
  end if
end while
if $G_1(\sigma) > 0$ **then**
  return Problem is infeasible.
else
  {Phase 2: Minimization of G_2 (violation of rise and set time constraints)}
  Set $\mathcal{U} \leftarrow \{1, \ldots, N\}$.
  while $|\mathcal{U}| > 0$ **do**
    Select $z \in \mathcal{U}$; set $\mathcal{U} \leftarrow \mathcal{U} \setminus \{z\}$.
    Calculate $\sigma^* \leftarrow \arg\min_{\sigma' \in \mathcal{N}_z(\sigma)} G_2(\sigma')$.
    Calculate $G_2' \leftarrow \min_{\sigma' \in \mathcal{N}_z(\sigma)} G_2(\sigma')$.
    if $G_2' < G_2(\sigma)$ **then**
      Set $\sigma \leftarrow \sigma^*$.
      Set $\mathcal{U} \leftarrow \{1, \ldots, z-1, z+1, \ldots, N\}$.
    end if
  end while
  if $G_2(\sigma) > 0$ **then**
    return Problem is infeasible.
  else
    return Feasible sequence σ.
  end if
end if

In the most significant case, with $N = 100$ targets, TTP-LS-1 obtains schedules with total slew times that achieve a reduction of approximately 80% relative to the $2N$ bound.

Table 4 and Figure 5 presents analogous results for TTP-LS-10.

We found that the TTP-LS-10 schedules were signficantly more efficient than the TTP-LS-1 schedules. For example, for $(\text{Full}, 100, 1)$, $\texttt{SlewTime}_\tau$ is 32.6 min for TTP-LS-10, compared to 43.7 min for TTP-LS-1.

In cases where the TTP-Global returned an optimal schedule, this schedule was often much more efficient than TTP-LS-10 and TTP-LS-1. For example, for $(\text{Full}, 50, 1)$, the TTP-Global MILP was solved to full optimality in seven out of ten instances, and the average $\texttt{SlewTime}_\tau$ was 13.2 min, compared to 27.6 min for TTP-LS-1 and 19.9 min for TTP-LS-10. TTP-LS-10 and TTP-LS-1 do not guarantee a globally optimal solution, but gap between local and global optima suggests additional work on heuristic solutions could prove fruitful.

On the other hand, in cases where the TTP-Global MILP does not terminate with an optimal solution, it is possible for the local search solution to perform better. For example, for the $(D, N, M) = (\text{Half}, 25, 10)$ experiments, the TTP-LS-10 solution has an average $\texttt{SlewTime}_\tau$ of 10.8 min compared to 12.7 min for TTP-Global. Lastly, the computation time for TTP-LS-10 is roughly ten times that of TTP-LS-1, as one would expect. However, we note that the ten repetitions are independent, and could be carried out in parallel. This could be attractive from an implementation standpoint, as both TTP-LS-1 and TTP-LS-10 were executed with a single-thread, so one could easily execute the local search procedure from multiple starting points in parallel within a multi-threaded computing environment.

There are several key takeaways from Figure 5 when $D = \text{Full}$. Adopting static target-to-target slew overheads ($M = 1$), we find TTP-Global solves the schedule for N up to 100 in most runs. TTP-Global produces the highest $\texttt{RelRed}_{\texttt{real}}$ improvement of above 80% for the $N = 100$ case. For smaller cases of N, it sees some benefit from higher values of M, but cannot find feasible solutions in the time limit for large N. The local solvers TTP-LS-1 and TTP-LS-10 scale exponentially with N, and find local optima for all N in their expected time limits. $\texttt{RelRed}_{\texttt{real}}$ benefits from

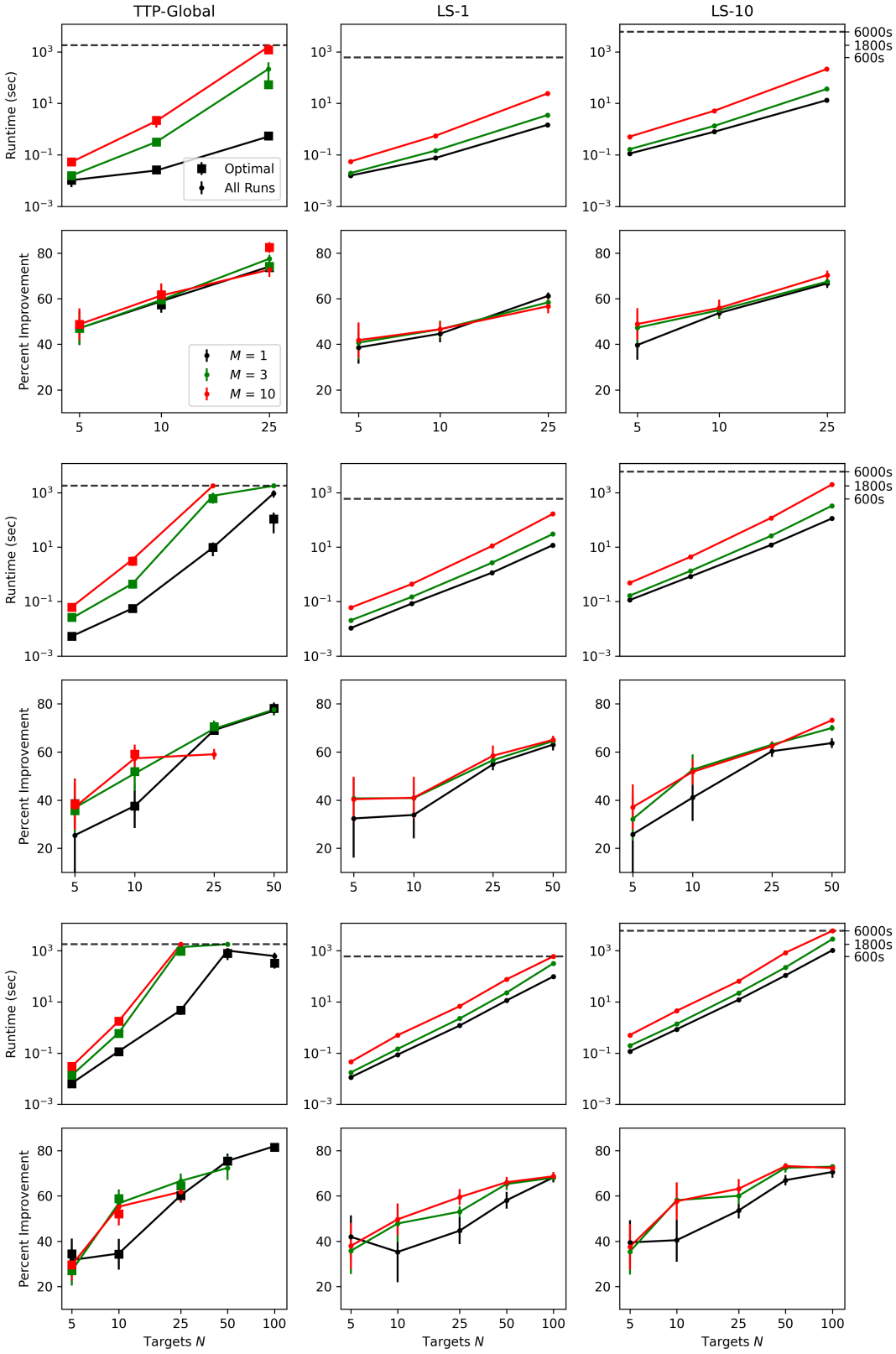

Figure 5. `Runtime` and $\texttt{RelRed}_{\texttt{real}}$ for each duration type D, as found in Tables 1, 3, and 4. The first two rows summarize these statistics for the D = Quarter simulations, the next two for D = Half, and the bottom two for D = Full. The top row in each pair shows the average `Runtime` (and optimal subset, for TTP-Global) across all runs for D with varying M as a function of N. The bottom row in each pair shows the average $\texttt{RelRed}_{real}$ for the same values of D and M.

D	N	M	NumFeas	Runtime (s)	$\texttt{SlewTime}_{\tau}$ (min)	$\texttt{RelRed}_{\tau}$ (%)	$\texttt{SlewTime}_{\texttt{real}}$ (min)	$\texttt{RelRed}_{\texttt{real}}$ (%)
Quarter	5	1	10	0.0	4.3	57.1	6.1	38.6
Quarter	5	3	10	0.0	4.4	56.3	5.9	40.7
Quarter	5	10	10	0.1	4.1	58.7	5.8	41.8
Quarter	10	1	10	0.1	7.7	61.3	11.1	44.6
Quarter	10	3	10	0.1	7.3	63.6	10.7	46.5
Quarter	10	10	10	0.6	7.1	64.7	10.7	46.6
Quarter	25	1	10	1.5	14.3	71.5	19.4	61.2
Quarter	25	3	10	3.5	13.7	72.5	20.8	58.4
Quarter	25	10	10	24.3	13.9	72.2	21.7	56.7
Half	5	1	10	0.0	5.4	45.9	6.8	32.4
Half	5	3	10	0.0	4.8	51.8	5.9	40.7
Half	5	10	10	0.1	4.6	53.6	6.0	40.4
Half	10	1	10	0.1	7.9	60.6	13.2	33.8
Half	10	3	10	0.1	7.7	61.5	11.8	40.8
Half	10	10	10	0.4	7.2	64.0	11.8	41.0
Half	25	1	10	1.1	15.9	68.3	22.6	54.9
Half	25	3	10	2.7	13.1	73.8	21.7	56.6
Half	25	10	10	11.1	13.4	73.2	20.8	58.4
Half	50	1	10	11.8	23.1	76.9	36.9	63.1
Half	50	3	10	30.4	20.3	79.7	35.5	64.5
Half	50	10	10	167.4	22.5	77.5	35.0	65.0
Full	5	1	9	0.0	6.2	37.6	6.6	34.4
Full	5	3	9	0.0	5.7	43.1	7.2	27.5
Full	5	10	9	0.0	5.5	44.6	7.0	30.0
Full	10	1	8	0.1	10.0	49.9	16.4	17.9
Full	10	3	9	0.1	9.5	52.6	11.7	41.5
Full	10	10	9	0.5	8.5	57.4	11.3	43.5
Full	25	1	10	1.2	17.5	65.0	27.7	44.7
Full	25	3	10	2.2	17.0	66.0	23.5	53.0
Full	25	10	10	6.8	14.4	71.2	20.3	59.5
Full	50	1	10	11.3	27.6	72.4	41.9	58.1
Full	50	3	10	23.1	23.2	76.8	34.7	65.3
Full	50	10	10	76.2	23.3	76.7	33.9	66.1
Full	100	1	10	98.7	43.7	78.2	63.6	68.2
Full	100	3	10	321.6	37.0	81.5	63.9	68.0
Full	100	10	10	602.0	38.7	80.6	62.6	68.7

Table 3. Computational results for TTP-LS-1 procedure.

higher M for moderate values of N, but also has diminishing returns for $N = 100$, and a lower ceiling compared to the global solution. Local search is equipped to find feasible solutions to larger models, but struggles to find solutions near the global optimum at high N, regardless of the value of M.

D	N	M	NumFeas	Runtime (s)	$\texttt{SlewTime}_\tau$ (min)	$\texttt{RelRed}_\tau$ (%)	$\texttt{SlewTime}_{\texttt{real}}$ (min)	$\texttt{RelRed}_{\texttt{real}}$ (%)
Quarter	5	1	10	0.1	4.1	59.1	6.0	39.6
Quarter	5	3	10	0.2	3.7	63.3	5.3	47.2
Quarter	5	10	10	0.5	3.5	64.5	5.1	48.9
Quarter	10	1	10	0.8	5.8	70.9	9.3	53.7
Quarter	10	3	10	1.3	5.8	71.2	9.0	55.0
Quarter	10	10	10	5.2	5.5	72.6	8.8	55.9
Quarter	25	1	10	13.2	10.2	79.7	16.7	66.7
Quarter	25	3	10	36.1	9.9	80.2	16.3	67.5
Quarter	25	10	10	214.4	9.6	80.8	14.8	70.3
Half	5	1	10	0.1	4.9	50.8	7.4	25.8
Half	5	3	10	0.2	4.7	53.2	6.8	32.1
Half	5	10	10	0.5	4.4	55.5	6.3	37.1
Half	10	1	10	0.8	6.6	67.2	11.8	41.1
Half	10	3	10	1.3	6.0	70.1	9.5	52.6
Half	10	10	10	4.4	5.8	71.0	9.7	51.7
Half	25	1	10	12.1	11.2	77.7	19.8	60.3
Half	25	3	10	26.0	10.8	78.4	18.5	63.0
Half	25	10	10	118.6	10.8	78.5	18.8	62.4
Half	50	1	10	113.1	19.1	80.9	36.3	63.7
Half	50	3	10	330.3	16.7	83.3	30.0	70.0
Half	50	10	10	2003.9	17.0	83.0	26.8	73.2
Full	5	1	9	0.1	6.0	39.7	6.8	31.6
Full	5	3	9	0.2	5.3	47.0	7.3	27.1
Full	5	10	9	0.5	5.3	47.3	7.0	29.5
Full	10	1	9	0.9	8.8	56.2	13.3	33.3
Full	10	3	9	1.4	7.7	61.7	9.4	52.9
Full	10	10	9	4.5	6.5	67.5	9.5	52.4
Full	25	1	10	12.2	12.2	75.5	23.2	53.6
Full	25	3	10	22.3	12.1	75.9	20.0	60.1
Full	25	10	10	65.4	11.4	77.1	18.4	63.2
Full	50	1	10	109.2	19.9	80.1	33.0	67.0
Full	50	3	10	224.9	19.5	80.5	27.5	72.5
Full	50	10	10	842.0	17.9	82.1	26.8	73.2
Full	100	1	10	1052.1	32.6	83.7	58.8	70.6
Full	100	3	10	2852.0	29.8	85.1	54.1	72.9
Full	100	10	10	6025.6	32.3	83.8	55.3	72.4

Table 4. Computational results for TTP-LS-10 procedure.

REFERENCES

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33, doi: 10.1051/0004-6361/201322068

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123, doi: 10.3847/1538-3881/aabc4f

Bellm, E. C., Kulkarni, S. R., Graham, M. J., et al. 2019a, PASP, 131, 018002, doi: 10.1088/1538-3873/aaecbe

Bellm, E. C., Kulkarni, S. R., Barlow, T., et al. 2019b, PASP, 131, 068003, doi: 10.1088/1538-3873/ab0c2a

Bertsimas, D., Chang, A., Mišić, V. V., & Mundru, N. 2019, Transportation Science, 53, 773, doi: 10.1287/trsc.2018.0847

Bertsimas, D., Lulli, G., & Odoni, A. 2011, Operations research, 59, 211, doi: 10.1287/opre.1100.0899

687 Croes, G. A. 1958, Operations research, 6, 791
688 Giuliano, M., & Johnston, M. 2008, 107–115
689 Gurobi Optimization, LLC. 2023, Gurobi Optimizer
690 Reference Manual. https://www.gurobi.com
691 Harris, C. R., Millman, K. J., van der Walt, S. J., et al.
692 2020, Nature, 585, 357, doi: 10.1038/s41586-020-2649-2
693 Howard, A. W., Johnson, J. A., Marcy, G. W., et al. 2010,
694 ApJ, 721, 1467, doi: 10.1088/0004-637X/721/2/1467
695 Hunter, J. D. 2007, Computing in Science & Engineering, 9,
696 90, doi: 10.1109/MCSE.2007.55
697 Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, ApJ, 873,
698 111, doi: 10.3847/1538-4357/ab042c
699 Johnston, M. D., & Miller, G. E. 1994
700 Mišić, V. V. 2020, Operations Research, 68, 1605,
701 doi: 10.1287/opre.2019.1928
702 pandas development team, T. 2020, pandas-dev/pandas:
703 Pandas, latest, Zenodo, doi: 10.5281/zenodo.3509134
704 Rosenthal, L. J., Fulton, B. J., Hirsch, L. A., et al. 2021,
705 ApJS, 255, 8, doi: 10.3847/1538-4365/abe23c
706 Sun, P., Veelenturf, L., Hewitt, M., & Van Woensel, T.
707 2018, Transportation Research Part B: Methodological,
708 116, 1, doi: 10.1016/j.trb.2018.07.002
709 Vielma, J. P. 2015, Siam Review, 57, 3,
710 doi: 10.1137/130915303