Area Coverage Planning with 3-axis Steerable, 2D Framing Sensors

Elly Shao and **Amos Byon** and **Chris Davies** and **Evan Davis** and
Russell Knight and **Garett Lewellen** and **Michael Trowbridge** and **Steve Chien**
Jet Propulsion Laboratory, California Institute of Technology
{elly.j.shao, amos.j.byon, christopher.j.davies, evan.w.davis,
russell.l.knight, michael.a.trowbridge, steve.a.chien}@jpl.nasa.gov

Abstract

Existing algorithms for Agile Earth Observing Satellites((Lemaître et al. 2002)) were largely created for 1D line sensors that acquire images in linear swaths. However, imaging satellites increasingly use 2D framing sensors (cameras) that capture discrete rectangular images. We describe tiling step-stare approaches that are more suited to rectangular image footprints than are 1D swath-based algorithms. Optimal area planning for these 2D framing instruments is an NP-complete problem and intractable for large areas, so we present four approximation algorithms. Strategies are compared against a prior 2D framing instrument algorithm (Knight 2014) in three computational experiments. The impact of observer agility on schedule makespan is examined. Makespans vary more as observer agility decreases toward a critical point, then vary less after the critical point, suggesting a possible problem phase transition.

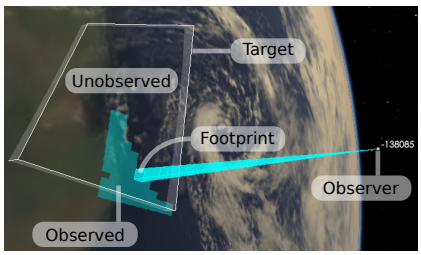

Figure 1: Framing sensor observations (teal) of a target area (white).

Introduction

Most existing area coverage algorithms for agile Earth-observing satellites are intended for 1D sensors and adapt poorly to 2D framing instruments with rectangular fields of view. These pushbroom algorithms continuously sweep the sensor across the target, which would smear the image captured by a framing sensor. An obvious alternative is to track a target point, which informs a step-stare strategy where the target is decomposed into a rectangular grid, and the sensor tracks each grid point for the duration of an image capture before moving on to the next(figure 2). The challenge

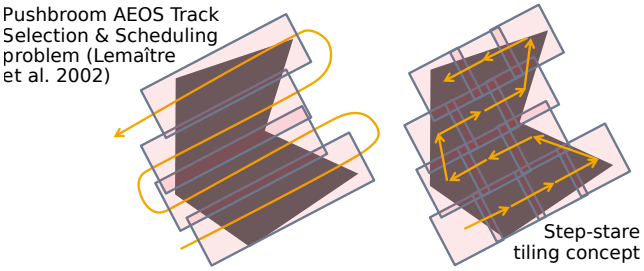

Figure 2: Comparison of pushbroom and step-stare.

in determining grid layout is that the the imager footprint (projection of instrument field of view onto body surface) change as the observer flies past (figure 3).

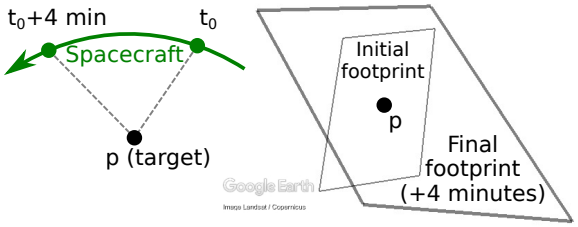

Figure 3: Imager footprint changes size and shape

We wish to image the entire target area while minimizing makespan (schedule duration). The area visible to the sensor is time dependent, and the cost to slew between target points is time varying and asymmetric (Lewellen et al. 2017a). This paper discusses the difficulty of this optimization problem, presents optimal solution approximation algorithms, then evaluates them in three computational experiments.

Contact author: Michael Trowbridge

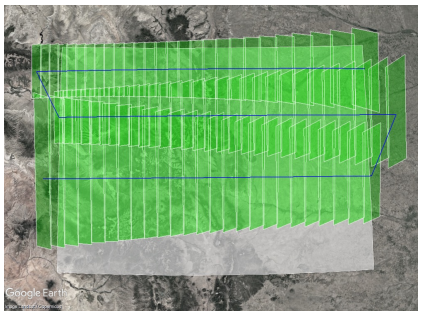

Figure 4: 1D (Lemaître et al. 2002) algorithm adapted to use a 2D framing sensor, with sweep lines (blue line) and image footprints (green). Only 81% of the target (white) was satisfied

Related Work

Lemaître et al. present a swath-based Boustrophedon (lawnmower) decomposition algorithm for area coverage with agile satellites (Lemaître et al. 2002). They argue that the track selection problem is NP-hard, so approximation algorithms are acceptable. The paper assumes 1D sensors and does not discuss 2D sensors.

Our adaptation of their algorithm for 2D framing instruments performed poorly when slew rate was constrained to the slower readout times of 2D framing instruments. On our easiest test case[1], the swath-based algorithm did not cover the entire target (figure 4).

Knight identified area coverage planning for 2D framing instruments as NP-hard by analogy to finding a Hamiltonian path through a grid-graph, then presented a concentric ring (Milling) tiling algorithm (Knight 2014). While provably optimal for even grid-graphs, a target polygon's grid decomposition could be odd, producing sub-optimal ring stitch points. The algorithm subdivides grid tiles into either two or four sub-tiles, which makes the graph even, but reduces area satisfied per image capture and increases makespan due to overlapping imager footprints. This paper presents algorithms that do not subdivide.

Formulation

Problem Statement

Choose observations (real-valued target points, rotations, observation times) s.t. the union of all images captured by those observations when projected onto the target body (modeled as a triaxial ellipsoid) cover the target great-circles polygon within a bounded temporal interval (single overflight), subject to constraints, with sufficient slew time between observations.

[1]GOLIAT/CICLOP CubeSat, using ON Semiconductor AR0331 subwindowed to 16:3 aspect ratio (Semiconductor Components Industries, LLC 2017)

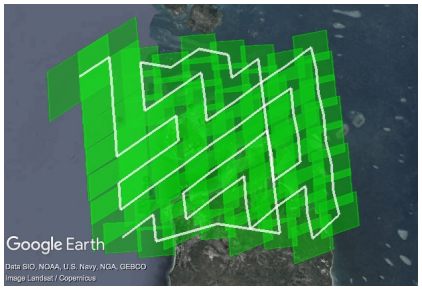

Figure 5: Milling (Knight 2014) algorithm grid points (connected by white line) and footprints (green). Subdivision causes excessive overlap and some skipped tiles that cover no target area (see edge crossing).

Nomenclature

p	A point on the target body
$(p_1, \ldots, p_n)$	A great circles polygon on the target body with n vertices
P	A set of target great circles polygons
$\mathbf{r}_{\text{obs}}$	Position vector of the observer at a given time t, determined by spacecraft orbit
$\mathbf{r}_{\text{tgt}}$	Position vector of the observer's imaging target (center of camera field of view)
θ	A rotation of the observer about its look vector $\mathbf{r}_{\text{look}}(\mathbf{r}_{\text{tgt}} - \mathbf{r}_{\text{obs}})$
t	Time. t_0 is the start time of the planning horizon, t_f is final time.
b	A single observation that captures an image: $(\mathbf{r}_{\text{obs}}, \mathbf{r}_{\text{tgt}}, \theta, t)$
B	The set of all possible valid observations b
A	The observation tour (schedule), an ordered list of scheduled valid observations a
m	Makespan of A, $m = [\min t \in A, \max t \in A]$
$Tile$	A image footprint polygon: intersection of the body surface and the observer's field of view
g	A *tile* corresponding to a scheduled observation
G	The set of scheduled image footprint polygons.

Formal Problem Statement

Given a set P of polygons on the target body,

$$P = \{(p_1, p_2, p_3)_i\} \tag{1}$$

the set B of all possible valid observations b that fall within the planning horizon $[t_0, t_f]$

$$\forall\, (b = \{\mathbf{r}_{\text{tgt}}, \theta, t\}) \in B, t_0 < t < t_f \tag{2}$$

a function to create a polygon g from an observation, representing the image footprint[2]

$$g \leftarrow \text{footprint}(\mathbf{r}_{\text{tgt}}, \theta, t) \tag{3}$$

[2]The image's angular field of view depends on instrument design

a Boolean valued function to check if a slew between observations b_i and b_j is valid

$$Boolean \leftarrow \text{slewOk}(b_i, b_j) \tag{4}$$

Some tour $A \subseteq B$ is valid iff

$$P \subseteq \text{union}\left(\{\text{footprint}\,(a_i)\,|i \in 1, 2, \ldots, |A|\}\right) \tag{5}$$

and

$$\bigwedge_{i=1}^{|A|-1} \text{slewOk}\,(a_i, a_{i+1}) \tag{6}$$

Constraints

The observation schedule A must have a makespan m within the visibility window $[t_{v0}, t_{vf}]$ where there is a line of sight from the observer to the target:

$$m \subseteq [t_{v0}, t_{vf}] \tag{7}$$

We enforce this by scoping B to the planning horizon ($[t_0, t_f] \subseteq [t_{v0}, t_{vf}]$). The geometric visibility window $[t_{v0}, t_{vf}]$ is a finite planning horizon.

Observations $b \in B$ have minimum duration

$$\Delta t_{\text{obs}} > 0 \tag{8}$$

Observations cannot be concurrents. Observation transition time is strictly positive and depends on spacecraft agility.

Geometric constraints are satisfied by restricting our search to the visibility interval $[t_{v0}, t_{vf}]$ determined with existing software[3].

Tractability of the Optimization Problem

OptFramePlan is the optimization formulation of the framing instrument area coverage scheduling problem. The goal is to find the shortest makespan schedule that satisfies conditions 5 and 6, subject to the previously listed constraints.

Theorem 1. *Finding the makespan-optimal area coverage plan for a space-based 2D framing instrument is NP-complete.*

Lemma 1.2 shows that the problem belongs to NP because an arbitrary solution is polynomial-time verifiable. Lemma 1.3 shows that the problem is at least as hard as finding a Hamiltonian path in a grid graph, which is NP-complete (Itai, Papadimitriou, and Szwarcfiter 1982).

Lemma 1.1. *The size $|A|$ of a solution schedule $A \subseteq B$ is bounded from above by $\frac{t_f - t_0}{\Delta t_{obs}}$.*

Proof. By contradiction/pigeon-hole principle. Assume A contains $\frac{t_f - t_0}{\Delta t_{\text{obs}}} + 1$ observations. Neglecting transition costs between observations, the horizon $[t_0, t_f]$ can accommodate $\frac{t_f - t_0}{\Delta t_{\text{obs}}}$ non-overlapping observations. If $|A| = \frac{t_f - t_0}{\Delta t_{\text{obs}}} + 1$, then at least two observations overlap, contradicting the problem constraints. □

[3]Systems Toolkit (STK) (Analytical Graphics Inc. 2017), Satellite Orbit Analysis Platform (SOAP) (Stodden and Galasso 1995), SPICE (Acton et al. 2016)

Lemma 1.2 (OptFramePlan $\in NP$)**.** *An arbitrary area coverage plan for a space-based 2D framing instrument is verifiable in polynomial time.*

Proof. Lemma 1.1 proves that $|A| = |G|$ is bounded linearly by the planning horizon and minimum observation duration. A plan that contains $|A|$ observations has at most $|A| - 1$ slews to validate using the constant-time slewOk function. Validating constraint compliance of each observation is also linear in $|A|$ observations. Unioning G (a set of sets) has no worse than $O(|G|^2)$ time complexity (Cormen et al. 2009). □

Lemma 1.3 (HamiltonianPath $\leq_P$ OptFramePlan)**.** *Finding the makespan-optimal area coverage plan for a space-based 2D framing instrument is polynomial-time mappable onto an instance of finding a Hamiltonian path in a grid graph.*

Proof. Relax the makespan-optimal area coverage planning problem by discretizing the target polygons into a uniform, target-fixed grid of points V. Omit rotation about the look vector θ and set the planning horizon short enough that skew may be neglected. Assume that the observer can slew equally well in any direction, making slew distance metric within the grid graph.

Define general grid graph $\Gamma = \{V, E\}$, where time cost $c_{ij}(t)$ between $v_i \in V$ to $v_j \in V$ via edge $e_{ij} \in E$ is a function of arrival time t, the sum of all previous costs added to the tour start time. An edge e_{ij} exists only if arrival time $t \leq t_{end} \leq t_f$, where t_{end} is the end time of the shortest known schedule. The makespan-optimal area coverage plan for this target discretization will visit each point in V once (i.e. be a Hamiltonian path). □

OptFramePlan is NP-complete, so makespan-optimal framing instrument scheduling of large areas is expected to be intractable. Thus, we limit our discussion to approximation algorithms.

Approximation Planning Algorithms

This section presents four deterministic step-stare tiling algorithms that approximate a makespan-optimal solution to OptFramePlan. The algorithms differ in when they commit the plan to target points $\mathbf{r}_{\text{tgt}}$, how far ahead they plan and how they maintain the plan.

Sidewinder: Target-fixed Boustrophedon

Sidewinder is an adaptation of the Boustrophedon (lawnmower) algorithm (Choset and Pignon 1998). Construct a grid of ground points R with a 2D flood-fill algorithm (Lee, Pan, and Chu 1987) and walk the grid in alternating rows (algorithm 1). Each $\mathbf{r}_{\text{tgt}} \in R$ is fixed at plan start time t_0 - but the other time-varying elements of the observation tuple a are fixed at schedule time t (algorithm 2).

Algorithm 1 Sidewinder planTour

```
function planSidewinderTour(P, t)
    Tour ← ∅                                   ▷ Planned tour
    bBox ← computeBoundingBox(P)
    closestSide ← findClosestSide(bBox, t)
    Tiles ← discretize(bBox)
    bearing ← true                  ▷ Alternates row direction
    (x_min, x_max, y_min, y_max) ← gridExtrema(Tiles)
    if closestSide ∈ {NORTH, SOUTH} then
        for i ← y_min to y_max do
            y ← y_max − i + y_min
            if closestSide = NORTH then
                y ← i
            end if
            for j ← x_min to x_max do
                x ← x_max − j + x_min
                if bearing then
                    x ← j
                end if
                Tour.add(x, y)
            end for
            bearing ← ¬bearing
        end for
    else                          ▷ East or West side is closest
        for i ← x_min to x_max do
            x ← x_max − i + x_min
            if closestSide = EAST then
                x ← i
            end if
            for j ← y_min to y_max do
                y ← y_max − j + y_min
                if bearing then
                    y = j
                end if
                Tour.add(x, y)
            end for
            bearing ← ¬bearing
        end for
    end if
    return Tour
end function
```

Algorithm 2 Sidewinder

```
while P ≠ ∅ do
    Tour ← planSidewinderTour(P, γ, t)
    while Tour ≠ ∅ do
        a_i ← pop(Tour, t)
        append a_i to A
        g ← footprint(a_i)
        P ← P − g
        t ← t + Δt_obs + slewDur(t, a_{i−1}, a_i)
    end while
end while
```

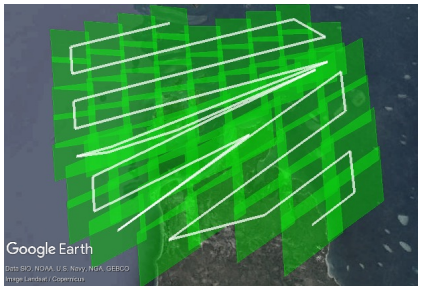

Figure 6: Sidewinder: rotation and skew invalidate the initial and second plans, prompting restarts by the outer loop (while $P \neq \emptyset$).

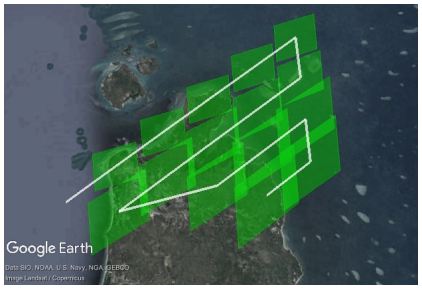

Figure 7: Fixing the grid to the target at t_0 causes gaps.

Replanning Sidewinder

Sidewinder commits grid points to target-fixed points too early. Image footprints change, so the plan develops gaps over time (figure 7). Replanning Sidewinder replans the tour after each move (algorithm 3) and only commits a planning grid point to a target point $\mathbf{r}_{\text{tgt}}$ at schedule time t.

Algorithm 3 Replanning Sidewinder

```
while P ≠ ∅ do
    γ_{i−1} ← pop(Tour) or center(P) if Tour = ∅
    γ ← optimizeGridOrigin(γ_{(i−1)})
    Tour ← planSidewinderTour(P, γ, t)
    a_i ← pop(T, t)
    append a_i to A
    g ← footprint(a_i)
    P ← P − g
    t ← t + Δt_obs + slewDur(t, a_{i−1}, a_i)
end while
```

Five pieces of plan state persist between replans: the most recently scheduled point $\mathbf{r}_{\text{tgt},i}$, next target point $\mathbf{r}_{\text{tgt},i+1}$, $\tau_{\text{row}} \in \{+,-\}$, $\tau_{\text{col}} \in \{+,-\}$ and row alignment axis $\alpha \in \{w,h\}$. The next grid origin point γ_{i+1} is $\mathbf{r}_{\text{tgt},i+1}$ and the algorithm usually[4] scans row 0. When $\mathbf{r}_{\text{tgt},i}$ discretizes to a row that is in the τ_{row} direction from $\mathbf{r}_{\text{tgt},i+1}$, τ_{col} is negated.

[4] If the next tour point is the final point in a row, and is in the tour at γ_{i-1} but out of the tour at γ_i, then the next

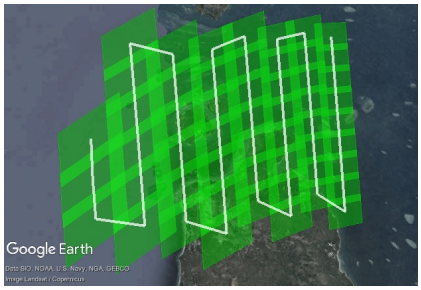

Figure 8: Replanning Sidewinder: adaptive row width

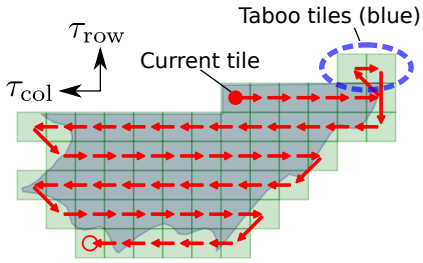

Figure 9: Two taboo tiles (circled in blue).

Changing geometry can move target area into a previously visited grid cell (the blue circle of figure 9), requiring a revisit of either a prior row (in the τ_{row} direction), or a different column in the current row (in the τ_{col} direction). Tiles requiring backwards moves are *taboo*. The tour can cycle between two taboo tiles until the opportunity interval is exhausted, so we remove taboo tiles by shifting the row with 1d constrained local optimization (algorithm 4). Perturb the origin by δ until the grid with origin $\gamma_{i+1} + \delta$ has no taboo tiles. Minimize δ in the τ_{row} direction (shifting rows backward).

Online Frontier Repair

This strategy updates a global plan after each action. The target is discretized into a grid using an 8-neighbor flood-fill based on the $O(n \log n)$ stack flood-fill algorithm (Lee, Pan, and Chu 1987). The initial plan (figure 10) is a rectangular Boustrophedon decomposition (Choset and Pignon 1998). Tiles are converted from observer planning space to ground coordinates at schedule time t (algorithm 5).

point in the tour will belong to row 1, not 0.

Algorithm 4 Optimize Grid Origin $\gamma_{(0,0)}$

T ▷ set of taboo tiles in the grid
w_{tile} ▷ width of a tile
h_{tile} ▷ height of a tile
α ▷ grid alignment direction (w or h)
function OPTIMIZEGRIDORIGIN($\gamma, \tau_{\text{row}}, \tau_{\text{col}}$)
  $D \leftarrow \begin{cases} \left[0, \frac{w_{\text{tile}}}{2}\right), & \text{if } \alpha = w \\ \left[0, \frac{h_{\text{tile}}}{2}\right), & \text{otherwise} \end{cases}$ ▷ search domain
  $\delta_\alpha \leftarrow \text{MINIMIZE}(|\delta|, \delta \in D \text{ s.t. } T|_{\gamma+\delta} = \emptyset)$
  return $\gamma + \delta_\alpha$
end function

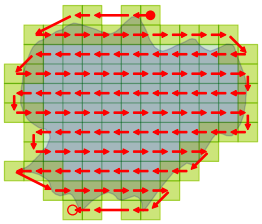

Figure 10: Initial plan (red arrows) from Online Frontier Repair algorithm. Frontier tiles are yellowed.

Algorithm 5 Online Frontier Repair

Plan $Tour$
while $P \neq \emptyset$ **do**
  UPDATEGRID($Tour, F, N, X$)
  REMOVE($Tour, x \in X$) ▷ tiles we no longer need
  INSERTCHEAPEST($Tour, n \in N$) ▷ New tiles
  $a_i \leftarrow \text{POP}(Tour, t)$
  append a_i to A
  $g \leftarrow \text{FOOTPRINT}(a_i)$
  $P \leftarrow P - g$
  $t \leftarrow t + \Delta t_{\text{obs}} + \text{SLEWDUR}(t, a_{i-1}, a_i)$
end while

Update the grid by seeding the flood-fill algorithm with the prior iteration's flood-fill result (algorithm 6). In the best case, only the outer edge changes. In the worst case, every tile is on the frontier, so updating is as slow as creating a new discretization – $O(|T| \log |T|)$ per update, where T is the set of grid points in the tour. In practice, interior points are infrequently re-evaluated.

We use the cheapest insertion heuristic (Rosenkrantz, Stearns, and Lewis 1977) with a Manhattan distance cost function when adding new tiles. Future work should examine more sophisticated heuristics.

Figure 11: Online Frontier Repair. Note suboptimal repairs on the right side (final leg).

Algorithm 6 Frontier Flood-fill Update

function UPDATEGRID($Tour, F, N, X$)
  $C_{in} \leftarrow \emptyset, C_{out} \leftarrow \emptyset$ ▷ closed lists
  $O \leftarrow F$ ▷ Open list O starts as frontier set F
  if $O = \emptyset$ **then**
    $O \leftarrow$ DISCRETIZE(unsatisfied target vertices)
  end if
  $S \leftarrow O$ ▷ Seeds: initial open list
  while $O \neq \emptyset$ **do**
    $o \leftarrow O$.pop()
    if COVERS(o, targets) **then**
      $C_{in} \leftarrow o$
      $membershipChanged \leftarrow (o \ni Tour)$
    else
      $C_{out} \leftarrow o$
      $membershipChanged \leftarrow (o \in Tour)$
    end if
    if $membershipChanged \vee (o \in S)$ **then**
      for all $n \in$ neighbors (o) **do**
        if $(n \ni C_{in}) \wedge (n \ni C_{out})$ **then**
          O.push (n)
        end if
      end for
    end if
  end while
  $N \leftarrow C_{in} - Tour$ ▷ identify new tiles
  $X \leftarrow Tour \cap C_{out}$ ▷ tiles to remove from tour
  $F \leftarrow \emptyset$ ▷ rebuild frontier list
  for all $tile \in Tour$ **do**
    if $|$NEIGHBORS$(tile) \cap Tour| < 8$ **then**
      F.push($tile$)
    end if
  end for
end function

Algorithm 7 Grid Nibbler checkNeighbors

function CHECKNEIGHBORS($\mathbf{r}, t$)
  $C \leftarrow$ GETCARDINALNEIGHBORS($\mathbf{r}, t$) ▷ ↑↓←→
  $D \leftarrow$ GETDIAGONALNEIGHBORS($\mathbf{r}, t$) ▷ ↙↖↗↘
  $best \leftarrow$ ARGMAX(score(c, t),$c \in C$)
  if $best$ did not finish a polygon **then**
    x ▷ Bias against diagonal neighbors
    $bestScore \leftarrow$ SCORE$(best, t)$ $\times x$
    for d in D **do**
      if d finishes a polygon **then**
        $best \leftarrow d$
      else if SCORE$(d, t) > bestScore$ **then**
        $best \leftarrow d$
        $bestScore \leftarrow$ SCORE(d, t)
      end if
    end for
  end if
  return $best$
end function

Local Grid Planning

This approach uses AI-inspired local planning with globally-informed heuristics such as radial distance of

Figure 12: Grid Nibbler: Radial distance heuristic.

Algorithm 8 Nibbler

while $P \neq \emptyset$ **do**
  $best \leftarrow$ CHECKNEIGHBORS($prev$)
  if AREA$(best, t) < \epsilon$ **then**
    $newStart \leftarrow closesttargetcorner$
    $best \leftarrow$ CHECKNEIGHBORS$(newStart, t)$
    if SCORE$(newStart, t) >$ SCORE$(best, t)$ **then**
      $best \leftarrow newStart$
    end if
  end if
  $a \leftarrow$ MAKEOBSERVATION$(best, t)$
  append a_i to A
  $g \leftarrow$ FOOTPRINT(a_i)
  $P \leftarrow P - g$
  $t \leftarrow t + \Delta t_{\text{obs}} +$ SLEWDUR(t, a_{i-1}, a_i)
end while

$\mathbf{r}_{tgt}$ from center of target polygons P, area of P that footprint g satisfies and number of polygons g eliminated from P.

Consider a 3×3 grid of tiles centered on the target corner closest to the previous pointing. Score the eight neighbors with some global heuristic, and add the highest-scoring neighbor to the tour (algorithm 7). Nibble (subtract) the imager footprint from the target. Repeat, centering the grid on the previous tour point, until no target remains (algorithm 8). To prevent gridlock, the previous direction is taboo.

Experiment Methodology

Each algorithm is used to schedule a target polygon in five experiments, with these metrics:

- Completeness: fraction of target satisfied
- Schedule efficiency: shortest makespan (duration)
- Computational efficiency: lowest CPU runtime
- Memory consumption (minus fixed overhead)

In one experiment, we fix the orbit, imaging payload and target, then vary observer agility to characterize the planning problem. The other experiment tests algorithm performance over both target difficulty and observer capability. Target difficulty is based on size (226381 km^2 vs 8181 km^2, respectively). Both the hard

and easy targets have almost direct overflights during the planning horizon.

Table 1: Hard and easy observer configurations

	Easy	**Hard**
Agility	GOLIAT	Commercial
Imager	CICLOP	THEIA
Orbit Altitude (km)	309 × 1441 [5]	615 [6]

Two key traits influence observer suitability for minimum-makespan scheduling: agility and field of view (FOV). The actual GOLIAT/CICLOP CubeSat (Balan and Piso 2008; Dumitru 2006) is our capable (easy) observer with a wide FOV and high agility (180° slew in 30 seconds). A hypothetical[7] observer with typical commercial imagery satellite orbit and agility (180° slew in 120 seconds), but a smaller 1° FOV THEIA framing imagery payload (Ellison et al. 2013) is our less capable (hard) observer. Table 2 shows the instrument models used in this experiment. Agility is modeled as two-point linear interpolation of eigenaxis slew angle[8] between targets, with fixed settle time.

Table 2: Imaging Instruments

	CICLOP [9]	**THEIA** [10]
Shape	Rectangular	Rectangular
Horizontal FOV	5.73°	1°
Vertical FOV	4.26°	1°
Image duration	0.17s	1.0s

The experiments run on a 2015 Macbook Pro (2.6 GHz Intel Core i7, 16 GB RAM).

Results

Agility and Problem Difficulty

Figure 13 demonstrates agility ranging from that of GOLIAT (upper limit) to ALL-STAR (lower limit), with a band covering some typical commercial imagery satellite agilities (Hutin 2009; Satellite Imaging Corporation 2017; MDA DigitalGlobe 2017). A commercial imagery satellite bus should be capable of satisfying the target area in a single overflight.

The problem is easier (more constrained) for less agile spacecraft because they slew too slowly to cover the entire target. For a more agile spacecraft, it is easy to find a valid solution, but harder to find the optimal solution. Future work should examine the critical point at the slow end of the commercial agility band (figure 13) as a possible problem phase transition.

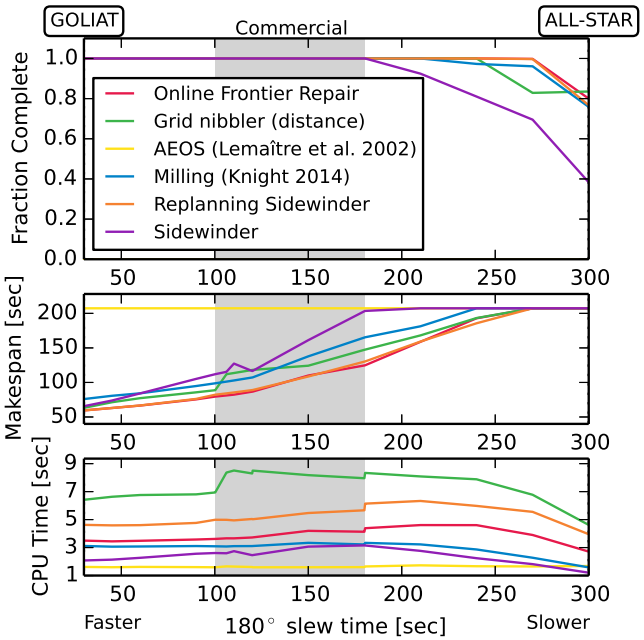

Figure 13: Performance under varying observer agilities

Observer Capability vs. Target Difficulty

When the observer is very agile and has a large FOV, path quality is less important. The easy/easy case is degenerate because CICLOP can satisfy the target with one image. In the easy observer, hard target case, Grid Nibbler generates the shortest schedule by requiring fewer images. The hard observer, easy target case shows the opposite - optimizing global planners generate shorter schedules with efficient pathing, despite requiring more images. We infer that lower agility requires more efficient paths because slews are more costly. With high agility, image number dominates path quality.

No algorithm completely satisfied the hard/hard case, even with multiple overflights. All algorithms consumed 10× more CPU and memory. Both Grid Nibbler variants covered more target area than the other algorithms.

Overall, Sidewinder used the least CPU time in all cases and the least memory in 3 of the 4. Grid Nibbler generally consumed more CPU time and memory than the other algorithms, with Online Frontier Repair and Replanning Sidewinder somewhere in between.

Overall Experiment Evaluation

All algorithms can produce admissible solutions, but no single algorithm is universally best. We recommend a

[5] Elliptical. Obtained from GOLIAT tracking TLE (Romanian Space Agency (ROSA) 2012).

[6] Circular. Reasonable when compared to commercial imagery satellite data sheets (Satellite Imaging Corporation 2017; MDA DigitalGlobe 2017).

[7] ALL-STAR is not agile enough and commercial imagery satellites (Hutin 2009; Satellite Imaging Corporation 2017; MDA DigitalGlobe 2017) do not use framing instruments

[8] The angle about an Euler rotation axis

[9] Data derived from (Dumitru 2006)

[10] Data obtained from (Ellison et al. 2013). No image duration value published for THEIA, assuming 1s.

Table 3: Comparison of algorithm performance for a cross-product of observer capability and target difficulty. Best in test values are green, - denotes failure to produce a schedule.

		Easy Observer (GOLIAT)					Hard Observer (Hybrid)				
	Algorithm	**CPU**	**RAM**	$\|m\|$	$\|A\|$	**%**	**CPU**	**RAM**	$\|m\|$	$\|A\|$	**%**
Easy target	Online Frontier Repair	2s	0.04MB	1s	1	100	4s	3.14MB	87s	54	100
	Replanning Sidewinder	2s	0.08MB	2s	2	100	5s	2.11MB	89s	54	100
	Milling (Knight 2014)	11s	0.04MB	11s	8	100	3s	0.37MB	107s	64	100
	Sidewinder	2s	0.05MB	2s	2	100	2s	0.30MB	117s	63	100
	Grid Nibbler (distance)	4s	0.05MB	1s	1	100	8s	4.13MB	118s	72	100
	Grid Nibbler (area)	3s	0.05MB	1s	1	100	14s	3.71MB	109s	52	100
Hard target	Online Frontier Repair	7s	3.21MB	87s	48	100	80s	22.80MB	39429s	387	32
	Replanning Sidewinder	9s	2.19MB	81s	41	100	-	-	-	-	-
	Milling (Knight 2014)	6s	0.42MB	118s	68	100	24s	3.80MB	39430s	343	30
	Sidewinder	3s	0.22MB	74s	43	100	19s	2.30MB	39430s	389	18
	Grid Nibbler (distance)	19s	4.52MB	96s	52	100	56s	23.20MB	39428s	391	34
	Grid Nibbler (area)	20s	3.73MB	70s	39	100	146s	23.30MB	39429s	392	41

portfolio approach, where a higher level scheduler considers a possible start time, then chooses the best algorithm for each circumstance, by either executing each algorithm and comparing makespans, or by evaluating a heuristic estimate model of each algorithm's makespan (Lewellen et al. 2017b).

Discussion

Overall, fewer tour points $|A|$ means shorter makespans $|m|$. However, tour quality has a greater impact on the less-agile observer: an efficient plan with more points can outperform a bad plan with fewer points (compare the hard observer/easy target instance of Online Frontier Repair vs. Grid Nibbler (area) in table 3).

All algorithms have linear complexity in number of tiles except for Online Frontier and Replanning Sidewinder, which are quadratic. Algorithmic complexity had negligible impact on schedule makespan and CPU runtime in our tests because $|A|$ was small.

These algorithms are only feasible on the upper end of current CubeSat processor modules. The algorithms themselves consume between 0.3 and 6.15 MB RAM, but our hasty implementation adds an extra 470 MB of non-algorithm memory overhead. This is too much RAM for low end PIC-based CubeSat modules, but reasonable for the 800 MHz CPU, 512 MB RAM Raspberry Pi compute module in the AAReST MirrorSat CubeSat (Underwood and Bridges 2015). Linearly scaling to an 800 MHz flight processor, our algorithms would require an estimated 10-65 seconds runtime to produce 70-118 seconds of schedule, faster than real time.

Future Work

We constrained our experiments to targets that are entirely within the field of regard. Larger targets could be decomposed into neighborhoods associated with visibility windows to accommodate multiple overflights.

Grid Nibbler was comparable to Online Frontier Repair, but was susceptible to dead ends due to its greedy approach. If Grid Nibbler looked ahead, it could retain the advantages of late commitment while avoiding dead ends. Future work should examine local optimization and relationships between next nibble heuristics.

Crude slew models were used for these experiments because detailed performance models of imagery satellites are typically not available to the public. Future efforts should constrain maximum angular rates and accelerations for slews, considering different agility about different spacecraft-fixed axes.

Conclusion

The three axis steerable 2D framing instrument area coverage planning problem was proven to be NP-complete. Four approximation algorithms for an optimal framing instrument path were outlined and compared in a computational experiment.

Naive approaches, such as applying pushbroom algorithms to a framing instrument or choosing a fixed target decomposition at start time, performed poorly. Locally scoped or replanning algorithms produced the shortest makespan schedules. Replanning Sidewinder and Online Frontier performed the best across the widest range of observer agilities. An algorithm from this paper outperformed the existing 1D (Lemaître et al. 2002) and Milling/Subdividable framing instrument (Knight 2014) algorithms in all experiments.

If the spacecraft is extremely agile, or if the target area is small relative to the imager footprint, the more sophisticated algorithms offer few advantages over a naive plan. The choice of algorithm matters most when the observer has only marginally sufficient agility to attempt a target.

Acknowledgments

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

Acton, C.; Bachman, N.; Semenov, B.; and Wright, E. 2016. Spice tools supporting planetary remote sensing. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 357–359.

Analytical Graphics Inc. 2017. Stk help/training part 4: Compute access. Web.

Balan, M., and Piso, M. 2008. GOLIAT project overview. In *5th Annual CubeSat Developers' Summer Workshop at the 22nd Annual AIAA/USU Conference on Small Satellites, Utah State University, Logan, Utah*.

Choset, H., and Pignon, P. 1998. Coverage path planning: the boustrophedon cellular decomposition. In *Field and Service Robotics*, 203–209. Springer.

Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. 2009. *Introduction to Algorithms*. MIT Press.

Dumitru, C. 2006. GOLIAT. In *Proceedings of the 2006 Spring CubeSat Developers' Workshop*. Cal Poly San Luis Obispo.

Ellison, J.; Massone, G.; Ela, N.; Goh, A.; Smith, L.; Sobtzak, J.; Muralidharan, V.; Hayden, I.; Spetzler, B.; Vente, G.; Lopez-Dayer, A.; Montoya, R.; McGehan, Q.; Jeffries, T.; Cook, C.; and Campuzano, B. 2013. ALL-STAR system integration review. Web.

Hutin, C. 2009. Pleaides meeting with ffg. Web.

Itai, A.; Papadimitriou, C. H.; and Szwarcfiter, J. L. 1982. Hamilton paths in grid graphs. *SIAM Journal on Computing* 11(4):676–11. Copyright - Copyright] 1982 Society for Industrial and Applied Mathematics; Last updated - 2012-02-06.

Knight, R. 2014. Area coverage planning for subdividable framing instruments. In *Proceedings of the 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014)*.

Lee, E. T.; Pan, Y.; and Chu, P. 1987. An algorithm for region filling using two-dimensional grammars. *International journal of intelligent systems* 2(3):255–263.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6:367–381.

Lewellen, G.; Davies, C.; Byon, A.; Knight, R.; Shao, E.; Tran, D.; and Trowbridge, M. 2017a. A Hybrid Traveling Salesman Problem - Squeaky Wheel Optimization Planner for Earth Observational Scheduling. In Chien, S., and Augenstein, S., eds., *Proceedings of the 10th International Workshop on Planning and Scheduling for Space (IWPSS)*, 62–72.

Lewellen, G.; Trowbridge, M.; Shao, E.; Davies, C.; and Knight, R. 2017b. Estimating Time to Image Areas with Steerable 2D Framing Sensors. In Chien, S., and Augenstein, S., eds., *Proceedings of the 10th International Workshop on Planning and Scheduling for Space (IWPSS)*, 73–83.

MDA DigitalGlobe. 2017. WorldView-4 Data Sheet. Web.

Romanian Space Agency (ROSA). 2012. GOLIAT TLE. Web. http://www.goliat.ro/.

Rosenkrantz, D. J.; Stearns, R. E.; and Lewis, II, P. M. 1977. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing* 6(3):563–581.

Satellite Imaging Corporation. 2017. IKONO Satellite Sensor. Web.

Semiconductor Components Industries, LLC. 2017. *AR0331 1/3Inch 3.1 Mp/Full HD Digital Image Sensor*, rev. 14 edition.

Stodden, D. Y., and Galasso, G. D. 1995. Space system visualization and analysis using the satellite orbit analysis program (soap). In *1995 IEEE Aerospace Applications Conference. Proceedings*, number 0, 369–387 vol.1.

Underwood, C., and Bridges, C. 2015. AAReST Spacecraft Update: Spacecraft Bus, Propulsion, ADCS, SSTL-50 CoreSat, RDV/Docking, OBDH and Comms. Web.