R. Knight, D. McLaren, and S. Hu. "Planning Coverage Campaigns for Mission Design and Analysis: CLASP for the proposed DESDynI Mission." International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), 4-6 September, 2012. Turin, Italy.

PLANNING COVERAGE CAMPAIGNS FOR MISSION DESIGN AND ANALYSIS: CLASP FOR THE PROPOSED DESDYNI MISSION

Russell Knight [(1)], David McLaren [(1)], and Steven Hu [(1)]

[(1)] Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109
<first name>.<last name>@jpl.nasa.gov

ABSTRACT

Mission design and analysis present challenges in that almost all variables are in constant flux, yet the goal is to achieve an acceptable level of performance against a concept of operations, which might also be in flux. To increase responsiveness, our approach is to use automated planning tools that allow for the continual modification of spacecraft, ground system, staffing, and concept of operations while returning metrics that are important to mission evaluation, such as area covered, peak memory usage, and peak data throughput. We have applied this approach to DESDynI (Deformation, Ecosystem Structure, and Dynamics of Ice) mission design concept using the CLASP (Compressed Large-scale Activity Scheduler/Planner) planning system [7], but since this adaptation many techniques have changed under the hood for CLASP and the DESDynI mission concept has undergone drastic changes, including that it has been renamed the Earth Radar Mission. Over the past two years, we have run more than fifty simulations with the CLASP-DESDynI adaptation, simulating different mission scenarios with changing parameters including targets, swaths, instrument modes, and data and downlink rates. We describe the evolution of simulations through the DESDynI MCR (Mission Concept Review) and afterwards.

1. INTRODUCTION

As mission designs evolve, some form of mission evaluation is necessary to determine how much science could be done with a given mission design. Historically, such evaluation was approximated, in the end leading to either missions that were very simple that could have done more or missions that promised too much. Generating high-fidelity mission plans is costly, yet a complete mission plan is the only real artifact that can be defended both from a scientific point of view and an engineering point of view. Enter automated planning and scheduling. Previously, we have reported on using CLASP to generate high-fidelity mission plans that can be evaluated by both scientists and engineers. But, new changes have forced us to increase the capability of CLASP to handle new challenges in the DESDynI domain.

First, we describe our history with CLASP and the DESDynI concept. We characterize the types of analyses needed for mission design evaluation and some of our iterations providing these analyses.

Then, we describe in detail the specific adaptation of CLASP proposed for DESDynI, including duty cycle, memory, downlink rates and visibilities, target sets and coverage requirements, and throughput limits. We then detail the way that CLASP uses the adaptation and instance information to produce a schedule.

CLASP can compress problems when they get too large for memory at a cost in terms of execution time, but we have discovered that this capability is largely unnecessary in that proper parameterization of the problem can keep it from becoming too large, even for huge campaigns spanning months or years, but planning at single second resolution. We describe the techniques we use to avoid filling memory, including sequencing of target sets (certain schedules can be planned, with subsequent schedules being interleaved into the previously scheduled observation plans), and choice of spatial decomposition parameters.

There are different ways in which CLASP can decompose spatial representations, from decomposing the globe into pixels of fixed size to using polygon representations of arbitrary resolution aka shards. Each of these has advantages and disadvantages, and we discuss the results of using these techniques in the context of the DESDynI mission concept. Finally, the overarching search algorithm of CLASP is described in detail using the DESDynI mission design as our example domain. We describe the decomposition of the possible visibilities and requested target sets into pixels or shards, and how this representation gives us access to information at runtime at constant-time speed. We describe how squeaky-wheel optimization works, including our techniques of conformant synergistic scheduling, where targets are added to observations where we can prove that no priority inversion is taking place for the target.

2. The DESDynI Concept

The objectives of the proposed DESDynI mission concept are to "determine the likelihood of earthquakes, volcanic eruptions, and landslides, predict the response

of ice sheets to climate change and impact on the sea level, characterize the effects of changing climate and land use on species habitats and carbon budget, [and] monitor the migration of fluids associated with hydrocarbon production and groundwater resources." [4] This is accomplished using an L-band RADAR with multiple polarizations. Further details about the DESDynI concept can be found on either the mission's web-site [4] or from the various IEEE Geoscience and Remote Sensing Symposia [5][9].

2.1. Simulation Background

The DESDynI simulations have been a long-running iterative process influencing the design of the proposed DESDynI mission, the development of CLASP, and further simulations that are run. Each simulation run with CLASP answers some questions about performance as it creates new ones. Additionally, some simulations generate requests for more output from CLASP or for new features in scheduling, and adding these to the CLASP core creates new possibilities for future simulations.

Most DESDynI simulations cover one quarter or "season" of a year (roughly 90 days in length). Each quarter is divided into equal-length cycles. There would be three major campaigns of interest in the DESDynI mission concept, with several subgroupings: ice (further divided into sea ice, land ice, and high-priority ice), vegetation (leaf on, leaf off, dry, wet), and land (global strain rate, anthropogenic deformation, and volcanic). Many of these campaigns would require one or more observations per cycle; some would require only one observation during an entire season. Some campaigns would require multiple looks (observations on both ascending and descending passes) to be satisfied. Finally, the target areas for campaigns would vary by quarter; sea ice targets change every quarter, and vegetation targets are most extensive in summer-time, when they cover most land in the northern hemisphere. For this reason summer quarters tend to be the most difficult to schedule while meeting science goals.

There are three major instrument modes used in the simulations: SP (single pol, required for most ice and land targets), LBSP (low-bandwidth single pol, suitable for sea ice and some land targets), and QP (quad- pol, usually required for vegetation targets). Quad-pol is the most data-intensive of these modes, often requiring more than 2 Gbps to take, with the narrowest swath, while LBSP has the lowest data requirements and the widest swath. Many simulations explore the use of additional modes (like QQP or "quasi-quad- pol") to meet science goals, and most define a hierarchy where higher data rate modes can satisfy lower modes. Simulations also define swaths of varying sizes. Early DESDynI simulations created QP observations on the narrowest swath, and LBSP/SP observations on wider swaths. In later simulations, the size differences between swaths became much smaller, and only the narrowest swath was used.

The goal of each simulation is to model the science goal satisfaction achievable by a particular DESDynI mission configuration, expressed as the percentage of targets satisfied both overall and within each campaign. Each simulation can vary a variety of parameters, including:

- Target sets and observation requirements (frequency and look requirements)
- Quarter and cycle length
- Swath sizes and temporal resolutions (60 second, 15 second, or 5 second segments). Observation resolution is the same as swath resolution.
- Instrument modes, data rates, and domination/satisfaction hierarchy
- Downlink rates and schedules
- Campaign priority
- Onboard recorder size and initial memory usage

2.2. Evolution of Simulations

Early CLASP simulations for the DESDynI mission concept used simple swaths and ample downlinks. They began with 90 and 180 km swaths, for QP and SP modes, divided into 60-second segments. 45 minutes of downlink time was scheduled per orbit at 1000 Mbps. In most simulations this level of downlink is a dream scenario, used to model an "unconstrained" case to test what the spacecraft could capture without memory constraints. Early simulations allowed the spacecraft to alternate between left and right rolled modes as needed to capture targets. Six 16-day cycles were simulated.

After these simple beginnings, more realistic swaths and stingier downlinks were introduced. Requirements for switching between left and right rolls were dropped, and simulations stuck to using right-rolled swaths only. We simulated scenarios to prepare for a meeting with DESDynI scientists in October 2010. These scenarios experimented with juggling priorities of campaigns in different cycles, in an attempt to force certain requirements to be met. The new swaths were between 226 (QP) – 248 (LBSP) kilometers wide. Three downlink scenarios were considered: a "minimum mission" consisting of a limited schedule of 130 Mbps downlinks using the Near Earth Network (NEN), a "basic" schedule providing 300 Mbps from the Tracking and Data Relay Satellite System (TDRSS) at 2 x 12 minutes per orbit, and an "unconstrained" schedule of 1000 Mbps at 45 minutes per orbit. The minimum mission scenario allowed only 17% total target coverage, the basic configuration allowed 68% coverage, and the unconstrained case captured 98% of all targets. A more detailed breakdown of target coverage by discipline is shown in Tab. 1. Recorder

usage for the unconstrained mission is shown in Fig. 1; memory usage remains low for most of the simulation, as data is drained off by downlinks faster than it is acquired.

Campaign	Coverage, 130 Mbps	Coverage, 300 Mbps, 2 x 12 mins / orbit	Coverage, 1000 Mbps, 45 minutes / orbit
Deformation	17.9	70.1	99.2
Ice	11.8	63.2	94.1
Vegetation	27.4	71.6	100.0
Overall	16.9	68.2	97.8

Table 1. Target coverage for various downlink scenarios

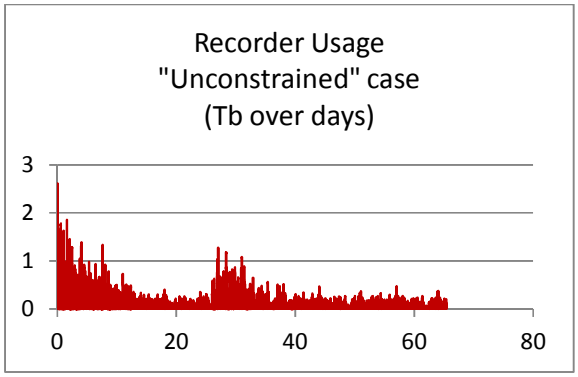

Figure 1. Recorder usage at 1 Gbps downlink, 45 min. per orbit

New simulations were requested to explore the possibility of meeting science goals with the 300 Mbps TDRSS option, without significantly increasing the amount of downlink available. An important element of the new simulations was the adoption of 15 second swath resolution, as opposed to 60 seconds in earlier simulations. Sixty second observations are long (covering more than 400 kilometers of arc on the surface, assuming a 7 km/s ground speed). Targets can be captured more efficiently with shorter observations, reducing the likelihood that observation time and memory are wasted on areas with no targets of interest. The downside to this approach is that scientists prefer to see contiguous observations, rather than seeing the instrument repeatedly turn on and off in short spurts. There is interest in making CLASP output more contiguous observations instead of frequently turning the instrument on and off, but this requirement is still ill-defined. By adopting fifteen-second swath segments, slightly increasing available downlink to 300Mbps at to 2 x 15 minutes per orbit, and dropping priority requirements on campaigns, target coverage improved to 97% overall, compared to the 68% coverage for the original basic scenario. Target coverage for this case is shown in Tab. 2. Memory usage is shown in Fig. 2. The usage is highest in the earlier cycles of the simulation, and drops off once targets that only need to be visited once have been scheduled.

Campaign	Coverage, Revised "basic" case
Deformation - Anthropogenic	100.0
Deformation – Global strain rate	99.5
Deformation - Landslides	100.0
Deformation – Volcano on land	99.8
Ice – High priority areas	99.8
Ice - Land	99.3
Ice - Sea	87.5
Vegetation	99.8
Overall	96.6

Table 2. Target coverage for basic case with 300 Mbps downlink rate.

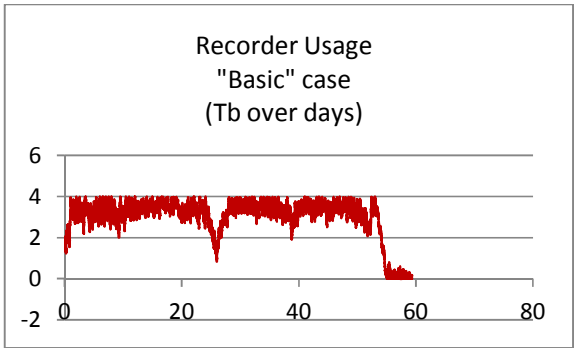

Figure 2. Recorder usage for case with 300 Mbps downlink rate.

The next set of simulations we ran incorporated new information from the Science team meeting to evaluate proposed DESDynI mission scenarios for the Mission Concept Review in January 2011. One of the realizations coming out of the meeting was that vegetation targets needed to be observed in both ascending and descending passes, sometimes more than once per season, to get adequate data. To meet these requirements, we began implementing an ability for CLASP to schedule targets that required multiple constraints (e.g. both ascending and descending passes) to be satisfied. Additionally, since swath widths were now very similar and differed by less than 25 km, we began running simulations using only the narrowest swath for all observations. This avoided the problem that CLASP does not upgrade an existing observation to a higher-data rate mode when doing so requires the upgraded observation to use a narrower swath (possibly uncovering some previously-satisfied targets).

Vegetation targets covered large areas and required the highest data-rate modes to observe, so either the amount of downlink available had to increase, or new schemes had to be tested for capturing vegetation targets. Thus a

new QQP ("quasi-quad pol") mode was introduced for imaging vegetation targets. This mode used a lower data rate than quad pol, but required three pairs of ascending and descending observations per quarter to meet science objectives. For the MCR, we simulated a configuration where the TDRSS downlink time was upped to 45 minutes per orbit at 300 Mbps. Even with the increased downlink time, our simulation showed that only 89% of the vegetation targets could be satisfied during the July-September quarter, when these targets were most extensive in land area. Tab. 3 shows the target coverage for this case. Fig. 3 is a profile showing onboard recorder usage during the course of the simulation. Memory usage peaks at the maximum of 4 Tb and tends to fall back towards zero between each 13-day cycle. This pattern appears because many targets need to be scheduled within a specific cycle.

Discipline	**Coverage (MCR)**
Deformation - Anthropogenic	100.0
Deformation – Global strain rate	100.0
Deformation – Landslides	99.8
Deformation – Volcano on land	100.0
Ice – High priority areas	100.0
Ice - Land	100.0
Ice - Sea	89.0
Vegetation	88.9
Overall	96.1

Table 3. MCR Simulation target coverage

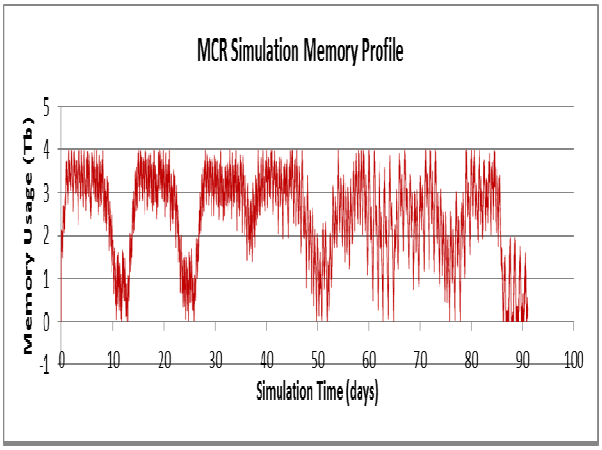

Figure 3. Memory profile for MCR simulation

After a successful MCR, DESDynI's budget was cut in Feburary 2011 and most work was put on hold. More recent CLASP simulations have explored cooperation with third parties, and tradeoffs between radar performance and cost. These simulations have run a variety of scenarios, including the use of two spacecraft to capture observations, and the use of a SpaceX Dragon capsule to serve as a platform for carrying the radar instrument, and collaborations with the Indian Space Research Organization and Canadian Space Agency.

One of the outcomes of these simulations is that they suggest new improvements to make for CLASP. As noted earlier, it is desirable for the CLASP-DESDynI adaptation to prefer creating contiguous observations instead of frequently starting and stopping instruments. Simulations for mission scenarios using multiple spacecraft also revealed a desire for spacecraft flying in tandem to observe a target using single-pass interferometry. We are also working continuously on improving CLASP performance, with the goal of initializing a simulation and performing multiple optimization iterations within 30 minutes.

3. CLASP

CLASP decomposes the geometric and temporal coverage problem into a set coverage problem, and then employs priority-based optimization to improve the schedule iteratively. First, we cover the encoding of the problem domain and describe the inputs to CLASP. Then, we describe how these inputs are transformed into a set scheduling problem (with a great deal of side constraints). We cover the choice between the pixel representation and the polygon or shard representation. Finally, we describe how we iterate over the creation of various schedules, changing the weights given to each target component (pixel or shard) and resorting and rescheduling, until we either cannot improve the schedule or we reach the end of our resources.

3.1. Inputs to CLASP

Fundamentally, CLASP is a coverage scheduler. That is, its role is to produce a list of time-stamped configurations for a spacecraft or aircraft (or collection of spacecraft or aircraft) that results in covering as much of the given target set as possible.

Inputs to CLASP describe 1) **resources**: the capacity, initial value, and constraints on resources available (memory, power, duty cycle), 2) **instruments**: what modes are available, what modes "dominate" or can substitute for other modes, the orbital characterization of the spacecraft or aircraft, and limits on slewing and footprint of the instrument, and what resource effects (data rate, power rates, etc) are associated with each mode, and 3) **targets**: what areas are we to collect data on, how many observations are necessary, what angles and "node" (ascending or descending) are required, and what are the temporal requirements.

Before going into detail about resources, instruments, and targets, a fundamental decision must be made by the user of CLASP as to which underlying representation we wish to use: pixels or shards. Pixels allow for fast computation at the cost of fidelity, shards allow for arbitrarily high fidelity at a cost of speed (in most

cases).

Pixels: when we *pixelate* the target body (Earth, Mars, etc.), we produce a cartographic graph over the surface of the body. The graph consists of cells along latitude and longitude boundaries (with the exception of the poles, where the graph cell is singleton and circular). The size of the shortest edge of the graph is a constant value for any latitude. The value is computed by dividing the latitude by the baseline cell size and rounding, thus the cells are guaranteed to be at least the size of a square where each side is of the baseline cell size. This cell size determines how many pixels cover the planet. Fig. 4 shows a comparison of two different pixel sizes.

Each pixel is then used as an individual component of area for both sensor-swaths and targets.

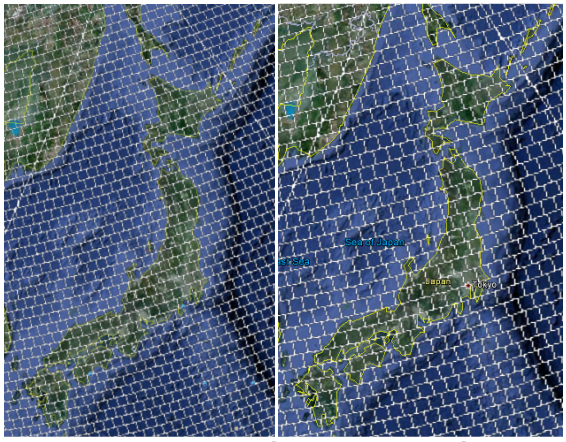

Figure 4. Japan with varying size pixels.

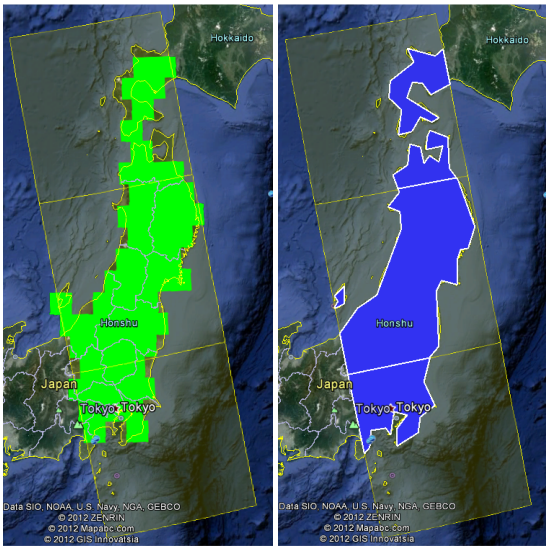

Figure 5. Japan coverage solution with pixels and shards.

Shards: shards are the smallest polygons that are induced by all of the lines representing sensor-swaths and targets. Each shard consists of a unique membership of overlapping swath-segments and targets. Shards have arbitrary resolution (almost, they are implemented using c++ doubles, so the resolution on Earth in radians is limited to less than a foot when values near π [at the anti-meridian], which is well within epsilon for any of our purposes). Fig. 5 shows a comparison of coverage solutions using pixels and shards.

Both shards and pixels represent areas that are covered by a set of sensors and a set of targets. The trade-off between these different representations has to do with the scale of the task at hand, the temporal resolution of the sensor-swaths, and the size and complexity of the target set. Small target sets with narrow swaths (e.g., aircraft flight plans) run much faster using shard representations than using high-density pixel representations. On the other hand, large target sets with wide sensor-swaths often run much faster using pixel representations. There is no hard and fast rule, so we leave such choices up to the user.

Henceforth, we will not refer to shards or pixels, but simply **cells** to represent the collection of areas that can be sensed and are requested to be collected.

Resources: in general, CLASP has the ability to characterize resources that fall into the integrating-value-over-time model. All of CLASP's resource reasoning is based on this model. The resource is modelled as a value over time (usually a c++ double, but other types are available). Constraints are levied on the resource in the form of rates over time. The resource has both rate and integrated value limits. Clamps can also be assigned to the upper or lower value of a resource rate or integrated value. This is useful for modelling resources such as memory, where a downlink that has emptied memory halfway through (the duration of the downlink) does not bring memory into the negative-fill but instead simply stops at empty (and also does not violate the limits of the resource). Time to propagate these timelines is linear in the number of individual time segments that it represents, and short-circuiting does occur, e.g., when a value has reached its clamped limit.

Hooks are available to provide CLASP resources of arbitrary semantics, but these must be implemented by the adaptor.

Instruments: one can think of instruments in CLASP as the marriage between resources and targets. Instrument definition starts with the concept of the sensor. Each sensor provides CLASP with an area over time that is visible from the spacecraft or aircraft. These can be

discretely described (in other words, each sensor can represent a slew angle of the spacecraft) or can be described using ranges (in other words, a sensor can indicate a left or right look angle from the spacecraft, with a slew angle and field of view prescribed). Sensors can be pre-processed and simply indicate time-stamped areas or can be described using SPICE [1] kernels and simple instrument models. Sensors also equate to *swaths* (areas on the surface to be imaged), and each discrete interval that a sensor covers is a *swath segment*. Thus, a swath-segment is a sensor field of view at a specific time for a specific duration. Each swath segment is mapped to the cells, which are in turn mapped back to the swath segment.

For example, imagine a spacecraft that can roll left or right, and images in a push-broom manner. We might provide CLASP a sensor definition in the form of a file containing a series of left-looking inner and outer coordinates with timestamps for the left swath and then label that sensor *1*. We would provide a similar file for the right swath and label that sensor *2*. The node (whether or not a sensor area at a certain is during the ascending node or descending node of the orbit of the spacecraft) is automatically computed.

Once a sensor is described and assigned its unique id, this id can be mapped to a mode. This allows us to indicate that various instrument modes can image only specific areas. Extending our previous example, sensor *1* could be assigned *high_resolution_left* and *low_resolution_left* and sensor *2* could be assigned to both *high_resolution_right* and *low_resolution_right* modes. Note that we only schedule modes that are associated directly with a sensor.

Modes need not be directly assigned to sensors. Modes can be abstract and only exist to be dominated. The domination hierarchy of modes indicates which modes can be used in substitution of other modes. A mode dominates another when it can substitute it. This allows us to model cases such as high fidelity modes that satisfy low fidelity target requests, but low fidelity modes not satisfying high fidelity target requests. Extending our previous example, we would indicate that both *high_resolution_left* and *high_resolution_right* dominates *high_resolution* and we would indicate that *low_resolution_left* and *low_resolution_right* dominate *low_resolution*, but not vice versa. We would also indicate that *high_resolution* dominates *low_resolution*. To speed dominance checking, the transitive closure is computed on the dominance graph, the result of which is used to determine dominance during scheduling. This results in a $O(logn)$ time to compute dominance where n is the number of dominance relationships.

Modes are also assigned resource usage. For example, one mode might produce more data than another. Any number of resource effects can be assigned to a mode. Extending our example, we would indicate that *high_resolution* mode uses memory at a rate of 1Gbps, but *low_resolution* mode uses memory only at 1Mbps.

Minimum transition duration requirements can also be assigned to modes. This allows us to model the time it takes to transition from one mode to the next. For example, if it takes no time to switch between high and low resolution, but it takes 1 minute to slew the spacecraft, we would indicate that the transition between *high_resolution_left* and *low_resolution_left* is 0 seconds, but that the transition between *high_resolution_left* and *high_resolution_right* is 60 seconds. Transition constraints are only defined for sensor-level modes (not abstract modes).

Targets: targets are described using maps of areas (primarily using GoogleEarth KML files). Targets may be polygons or points. Each target includes a campaign statement that describes what is required to image the target. The campaign statement includes the mode (sensor-level or abstract) priority, how many times the target must be collected, the constraints on look angle (left and right) and node (ascending or descending). Temporal constraints can be described for observations, including disjunctive intervals.

3.2. Transforming the Targets

Targets and swath segments are all eventually assigned to a set of cells (pixels or shards). The individual cells are the smallest component of area CLASP reasons about. Targets are broken down into a set of cell-targets. These cell-targets are what are scheduled by the scheduling engine. Each cell-target consists of a set of constraints that must be met (priority, look angle, node, mode, etc.) and a scheduling priority (not to be confused with target priority).

3.3. Iterative scheduling

The task at hand is to assign modes over time that maximize our coverage objective. Our approach is to use squeaky-wheel optimization to schedule the cell-targets. Squeaky-wheel optimization (SWO) [5] has two pre-requisites: 1) each task to be scheduled has a scheduler priority that is modified during optimization and 2) a greedy scheduler exists that can schedule the tasks according to this priority. SWO is used primarily when the goal is to schedule as many tasks as possible with the assumption that we have more tasks than capacity. SWO terminates when all tasks have been scheduled or when a time bound is exceeded. During our outline of the scheduling algorithm we will see that we have modified SWO in several different ways.

The first step then is to initialize the scheduler priority for each cell-target. Each is assigned a scheduler-

priority of 0.

Then, at each iteration, we order the cell-targets by priority (in ascending order), scheduler priority (in descending order), then by the earliest visibility, and then randomly. Here we see that CLASP is a strict priority order scheduler, i.e., no number of lower priority tasks can pre-empt a higher priority task. Thus, the first time through the loop, we are scheduling in temporal order.

So, for each cell-target, we find the first possible scheduling that violates no constraints, where *scheduling* means assignment to a swath segment and sensor-level mode. It should be noted that swath segments are considered mutually exclusive. Some targets are expressed as constraint pairs, so in that case we must find a scheduling that satisfies both sets of constraints. Once a time is found, the cell-target is assigned to that time. Determining visibility is very fast (constant time) as the cell-target maintains a reference to the cell, which has a reference to the sensors that cover that cell. In the case that a sensor is continuously adjustable, we simply check to see if the width of the current range of included cells plus the cell-target is covered by the field of view of the instrument. This is a least-commitment strategy that allows us to avoid painting ourselves into a corner with respect to choosing pointing angles for the sensor.

Note that determining whether or not a cell-target can be assigned at a time is not a simple matter of checking the visibility. If a mode is already active over the interval, then we need to see if that mode is compatible with or could be upgraded to a mode that dominates the requested cell-target mode and the current active mode. If a mode is not already active over the interval, then we need to find the "closest" mode that can be accommodated by the resources, dominates or is equivalent to the cell-target mode, and violates no transition duration constraints. The most costly of these checks are checks against resources.

When we set a new mode over an interval, we are assigning a mode to a swath segment. At this time, we perform *synergistic* scheduling: scheduling all other cell-targets that can be scheduled during this sensor. First, we find the swath segment associated with this cell-target. This requires no time as it is maintained as a by-product of determining that this cell-target can be scheduled in the first place, although the original cost of finding this is still constant time through references to the cell. Once we have a sensor, we can then loop through all cells associated with that sensor and loop through all cell-targets associated with each cell. We check for compatibility. We disallow changing modes as this leads to priority inversion. We also disallow expanding the width of the current field of view for continuous sensors for the same reason. We schedule (assign to this swath segment) all cell-targets that are compatible. We disallow scheduling of pair-observation targets as empirically we have determined that a decrease in performance is realized as previous results of checking one half of the pair are not cached for future evaluation.

After generating each schedule, we evaluate how well we did based on the area covered. In the case of pixel representations, we simply use the count. In the case of shard representations, we use the sum of the area. We keep a log of our performance and short-circuit when a long enough cycle of stagnant performance is detected.

Cell-targets that are not scheduled during this round have their scheduler priority increased, i.e., the squeaky wheel gets the grease. For previous versions, we kept track of the highest quality schedule that each cell-target participated in. This allowed us to find poison targets that, when introduced into a schedule, forced the overall quality of the schedule to be exceedingly low. We then could remove the targets and stop them from dragging down the overall quality of our search (basically resulting in filling the valleys), i.e., the squeaky wheel gets greased. But, for large problem instances, we discovered that no individual cell-target was poisonous, thus this optimization actually reduces the overall performance due to the slight increase in resources required to track it.

3.4. Pseudo code

Here we present some pseudo code to help make more concrete our algorithms.

Collect Shards

(1) let S be the set of swath segments
(2) let T be the set of targets
(3) let E be a set of edges
(4) let C be the resultant set of polygons (the shards)
(5) for each $s \in S$, for each $e \in s$.edges
(6) add e to E
(7) end for, end for
(8) for each $t \in T$, for each $e \in t$.edges
(9) add e to E
(10) end for, end for
(11) while $\exists\ e \in E, e' \in E \mid e \neq e'$ and e intersects e' and the intersection is not at either end of e
(12) remove e from E
(13) split e into two edges at the intersection point and reinsert both into E
(14) end while
(15) while $|E| > 0$
(16) $e = E$.pop
(17) if e is not vertical
(18) let p be a new polygon where e is a

```
        southern edge
(19)        let e′ = e
(20)        do
(21)            add e′ to p.edges
(22)            let e′ be the next edge adjacent to e
        following a clockwise motion
(23)            if e′ is a southern edge of p
(24)                remove e′ from E
(25)            end if
(26)        until e′ = e
(27)        add p to C
(28)    end if
(29) end while
```

Schedule

```
(1)  let C be the set of cell-targets
(2)  let S be a set of (mode, swath-segment)
(3)  let F be the set of failed cell-targets
(4)  do
(5)     let F = {}
(6)     sort C according to scheduler priority
(7)     for each c ∈ C
(8)        if we find a mode m and swath segment s |
        (m, s) satisfies c
(9)           add (m, s) to S
(10)          for each c′ in S.cell-targets
(11)             if (m, s) satisfies c′
(12)                remove c′ from C
(13)             end if
(14)          end for
(15)       else
(16)          add c to F
(17)       end if
(18)    end for
(19)    for each c ∈ F
(20)       increment c.scheduler-priority
(21)    end for
(22) until |F| = 0 or we have no more time
```

3.5. Other Adaptations of CLASP

Other adaptations/applications of CLASP to spacecraft scheduling include the THEMIS prototype [8] and scheduling for the proposed HyspIRI IPM [3]. It has also been baselined as the IPE CubeSat ground system [2].

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

4. REFERENCES

[1] Acton, C., Backman, N., Elson, L., Semenov, B., Turner, F., Wright, E., “Extending NASA’s SPICE ancillary information system to meet future mission needs,” SpaceOps 2002, Houston, Texas, October 2002.

[2] S. Chien, J. Doubleday, K. Ortega, T. Flatley, G. Crum, A. Geist, M. Lin, A. Williams, J. Bellardo, J. Puig-Suari, E. Stanton, E. Yee, “Onboard Processing and Autonomous Operations on the IPEX Cubesat Mission,” Government Cubesat Symposium, Moffett Field, CA, April 2012.

[3] S. Chien, D. Silverman, G. Rabideau, D. Mandl, J. Hengemihle, “A Direct Broadcast Operations Concept for the HyspIRI Mission,” Space Operations 2010, Huntsville, AL, April 2010.

[4] “DESDynI”, http://desdyni.jpl.nasa.gov

[5] Donnellan, A., Rosen, P., Ranson, K.J., Zebker, H., "Deformation, Ecosystem Structure, and Dynamics of Ice (DESDynI)," IGARSS 2008, Boston, Massachusetts, July 2008.

[6] Joslin, D. E., and Clements, D. P., “Squeaky wheel optimization,” *J. Artif. Intell. Res*., vol. 10, pp. 353–373, 1999

[7] Knight, R., Hu, S., “Compressed Large-scale Activity Scheduling and Planning (CLASP) applied to DESDynI,” Proceedings of the Sixth International Workshop in Planning and Scheduling for Space, Pasadena, CA, July 2009.

[8] G. Rabideau, S. Chien, D. Mclaren, R. Knight, S. Anwar, G. Mehall, “A Tool for Scheduling THEMIS Observations, International Symposium on Space Artificial Intelligence, Robotics, and Automation for Space, Sapporo, Japan, August 2010.

[9] Rosen, P., Eisen, H., Shen, Y., Hensley, S. Shaffer, S., Veilleux, L, Ranson, J., Dress, A., Blair, B., Luthcke, S., Dubayah, R., Hager, B. H., "Making the most of DESDynI: An Overview of the Proposed Mission," IGARSS 2011, Vancouver, Canada, July 2011.