Planning Satellite Swarm Measurements for Earth Science Models: Comparing Constraint Processing and MILP Methods

Rich Levinson,[1,2] **Samantha Niemoeller,**[1] **Sreeja Nag,**[1,3] **Vinay Ravindra**[1,3]
[1]NASA Ames Research Center, Moffett Field, CA 94035
[2]KBR Wyle Services, LLC, Moffett Field, CA 94035
[3]Bay Area Environmental Research Institute, Moffett Field, CA 94035
{rich.levinson, samantha.c.niemoeller, sreeja.nag}@nasa.gov
vravindra@baeri.org

Abstract

We compare two planner solutions for a challenging Earth science application to plan coordinated measurements (observations) for a constellation of satellites. This problem is combinatorically explosive, involving many degrees of freedom for planner choices. Each satellite carries two different sensors and is maneuverable to 61 pointing angle options. The sensors collect data to update the predictions made by a high-fidelity global soil moisture prediction model. Soil moisture is an important geophysical variable whose knowledge is used in applications such as crop health monitoring and predictions of floods, droughts, and fires.

The global soil-moisture model produces soil-moisture predictions with associated prediction errors over the globe represented by a grid of 1.67 million Ground Positions (GPs). The prediction error varies over space and time and can change drastically with events like rain/fire. The planner's goal is to select measurements which reduce prediction errors to improve future predictions. This is done by targeting high-quality observations at locations of high prediction-error. Observations can be made in multiple ways, such as by using one or more instruments or different pointing angles; the planner seeks to select the way with the least measurement-error (higher observation quality).

In this paper we compare two planning approaches to this problem: Dynamic Constraint Processing (DCP) and Mixed Integer Linear Programming (MILP). We match inputs and metrics for both DCP and MILP algorithms to enable a direct apples-to-apples comparison. DCP uses domain heuristics to find solutions within a reasonable time for our application but cannot be proven optimal, while the MILP produces provably optimal solutions. We demonstrate and discuss the trades between DCP flexibility and performance vs. MILP's promise of provable optimality.

Science Problem and Application

Soil moisture is an important geophysical variable that can forewarn of impending drought or flood conditions before other more standard indicators are triggered (NIDIS, 2021). Other soil moisture applications include wildfire and landslide forecasting, climate change, agriculture, water supply monitoring, and other resource management.

Our goal is to improve soil-moisture predictions. We hypothesize predictions will be improved by observing ground positions with larger prediction-errors, using observations of higher quality (i.e., lower measurement errors), then feeding this data back to the predictor. Thus, our planner has the objective to maximize the reduction of prediction error for the Ground Positions (GP) which are planned to be observed.

Important need for planning to solve problem: The need for optimal usage of limited satellite resources for time, energy and computing provides strong motivation for planning. Rapid responses to quick changing natural phenomena like fire requires fast replanning to direct agile spacecraft to optimal observations with the optimal sensor selection. Agile satellites that can change viewing angle provide opportunities for opportunistic planning, compared to satellites where the sensor angle is fixed. Planning is also required to optimize for complex metrics combining prediction and measurement-errors, compared to the relatively simple metric of maximizing the # of GP observed.

Remote sensing missions in the past have re-pointed single instruments given ground-commanded waypoints (CHRIS on Proba (Barnsley et al., 2004)), 3-DOF imaging for Planet's Skybox spacecraft (Augustein et al., 2016), and EO-1 re-tasking for monitoring of floods (Chien et al., 2019), volcanoes (Chien et al., 2020), and wildfires (Chien et al., 2011). Missions without physical agility have shown to benefit from reactive planning to prioritize hyperspectral data collection, such as IPEX which served as the HyspIRI pathfinder (Chien et al., 2016) and the future EnMAP (Worle et al., 2014, Fruth et al., 2019), and to inform operational parameters like electronic beam steering to optimize radar looks, such as TerraSAR-X (Werninghaus and Buckreuss, 2009) and TanDEM-X (Krieger et al, 2007). Power and bandwidth restrictions on small spacecraft has spurred literature on scheduling data download (Jian and Cheng, 2008) and use of crosslinks to propagate planning information via space nodes (Linnabary et al., 2019). However, these tools are optimized only for data downlink without a science-driven observation planner in the loop.

The D-SHIELD application (Levinson, et al, 2021, Nag et al., 2020; 2019) uses a (proposed) constellation of satellites looking at Earth to reduce errors in global soil moisture prediction by making observations that target spatio-temporal points of rising prediction error.

The constellation consists of 3 maneuverable satellites. Each satellite carries two different instruments: L-band, and P-band Synthetic Aperture Radars (SARs). Each of the satellites is maneuverable to 61 different orientations, also referred to as pointing-options or viewing angles. On each satellite, both instruments share the same pointing option. The global land-area is represented by a grid with resolution of 9kmx9km, which results in 1.67 million GP. A soil moisture model predicts soil moisture and associated prediction-errors across all GP. The prediction model uses machine learning to dynamically produce soil-moisture predictions based on prior observations.

The overall goal is a system which dynamically plans new observations to improve predictions of a science model (e.g., weather forecast, flood simulations). In the case of D-SHIELD's soil-moisture application, this is done by targeting locations where the prediction error is greatest, using instrument parameters that minimize measurement errors. The planner produces observation plans for each satellite. It decides what to look at, when to look at it, and how to look at it. The planner schedules coordinated <instrument, viewAngle> measurement pairs for each satellite.

Figure 1 shows the D-SHIELD system. The solid lines show the closed-loop control flow while the dashed lines show static planner inputs and external data sources used to update the prediction model.

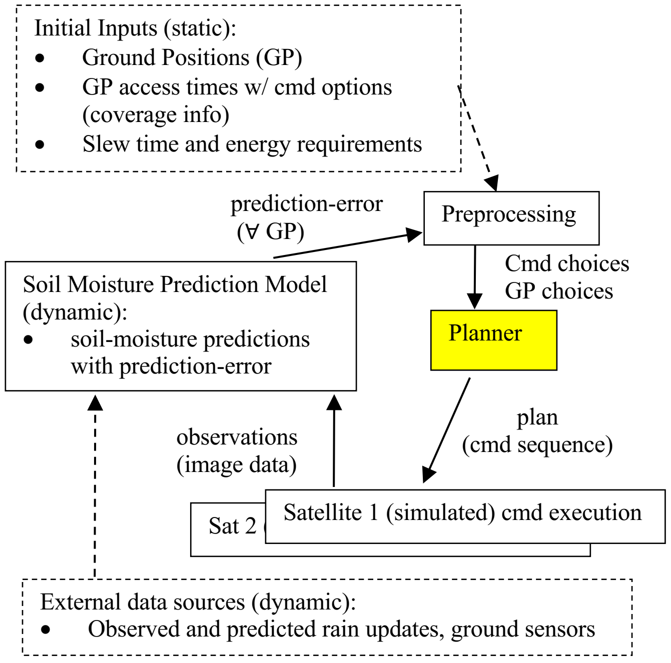

Initial Inputs (static):
• Ground Positions (GP)
• GP access times w/ cmd options (coverage info)
• Slew time and energy requirements
Preprocessing
prediction-error (∀ GP)
Cmd choices
GP choices
Soil Moisture Prediction Model (dynamic):
• soil-moisture predictions with prediction-error
Planner
plan (cmd sequence)
observations (image data)
Sat 2
Satellite 1 (simulated) cmd execution
External data sources (dynamic):
• Observed and predicted rain updates, ground sensors

Figure 1: D-SHIELD architecture

Raw inputs are preprocessed into planner input files. The (simulated) satellites execute their plans, collect observation data and pass that back to the Soil Moisture Model, to update the model based on the new observations.

This paper compares two versions the planner: Dynamic Constraint Processing (DCP) and MILP. The DCP planner was introduced in (Levinson et. al., 2021). The MILP formulation and comparison is new.

Planning Problem

The planner's job is to create a coordinated multi-satellite observation plan which improves prediction quality by observing the GP with the highest prediction error using measurements with the least error. It produces a command sequence for each satellite to execute to take observations.

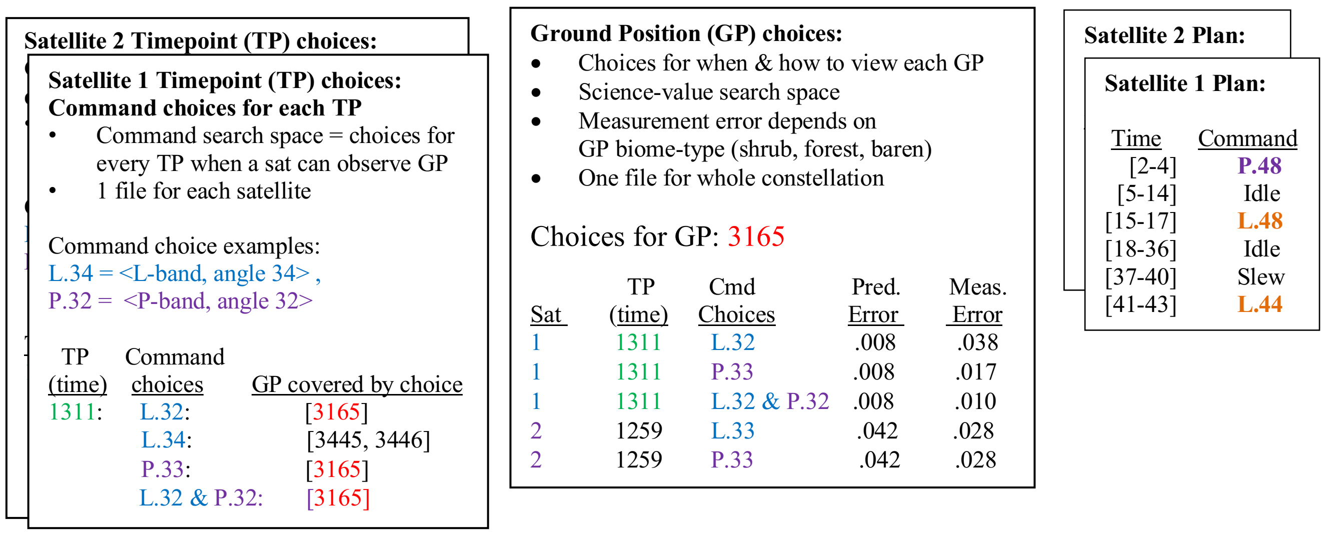
Each satellite's plan specifies the time when one or both instruments will take images at a given pointing option (viewing angle). Each observation by the L or P band instruments covers multiple and variable number of GPs.

The satellite coverage information maps each GP to time-points (TP) when it can be accessed (observed), and is produced by the EO-Sim software (Ravindra et al., 2021). GP coverage information specifies the observation opportunities for different satellites, instruments and pointing options This raw coverage information is converted to a format which defines the planner's search space, specifying *which* satellites can look at *which* GPs, at *which* times, with *which* instruments, using *which* viewing angles. The planner also takes in a *Slew Table* specifying the time and energy required to maneuver between any two pointing options (Sin et. al., 2021).

Prediction and Measurement Error Tables: Planner inputs include global predicted soil moisture error for the next 24 hours (the "prediction error"), and a measurement error table which defines the expected measurement error for any combination of instruments and viewing angles and biome type (e.g., forest, marshland, urban). Prediction quality is inversely proportional to the prediction error associated with each GP. The prediction error generally increases over time and jumps after events like rain or fire.

Measurement error is a function of which instrument is used, the viewing angle, the type of ground cover (e.g., barren, shrubs, forest, croplands), and other ancillary parameters. Each GP is associated with a type of ground cover (*"biome type"*).

Constraints: The planner enforces constraints within a single satellite and swarm-wide. For each satellite, deconfliction constraints enforce that each satellite can do only one thing at a time. Each command must hold for 3 seconds during which no other command can be scheduled. Additionally, no commands can be scheduled while the satellite slews to a new viewing angle. Image lock and slew time constraints are enforced per satellite.



Satellite 2 Timepoint (TP) choices:

Satellite 1 Timepoint (TP) choices:
Command choices for each TP

- Command search space = choices for every TP when a sat can observe GP
- 1 file for each satellite

Command choice examples:
L.34 = <L-band, angle 34> ,
P.32 = <P-band, angle 32>

TP (time)	Command choices	GP covered by choice
1311:	L.32:	[3165]
	L.34:	[3445, 3446]
	P.33:	[3165]
	L.32 & P.32:	[3165]

Ground Position (GP) choices:

- Choices for when & how to view each GP
- Science-value search space
- Measurement error depends on GP biome-type (shrub, forest, baren)
- One file for whole constellation

Choices for GP: 3165

Sat	TP (time)	Cmd Choices	Pred. Error	Meas. Error
1	1311	L.32	.008	.038
1	1311	P.33	.008	.017
1	1311	L.32 & P.32	.008	.010
2	1259	L.33	.042	.028
2	1259	P.33	.042	.028

Satellite 2 Plan:

Satellite 1 Plan:

Time	Command
[2-4]	**P.48**
[5-14]	Idle
[15-17]	**L.48**
[18-36]	Idle
[37-40]	Slew
[41-43]	**L.44**

Figure 2: Preprocessing produces TP choices files for each satellite (*left*) and a single GP choice file (*middle*) for the swarm. Planner Output (*right*) is a plan for each satellite containing scheduled observations and slew actions.

Image Lock Constraints - Each observation requires the instrument to hold viewing angle for 3 seconds, blocking out slewing to another viewing angle during that period.

Slew time Constraints - The satellite must slew to change viewing angles. Agility constraints on how quickly a satellite can change viewing angles depend on slew magnitude. The planner ensures enough time to slew between each observation. The maximum slew time is 22 secs.

Duplicate observation Constraints (enforced swarm-wide) - To cover more GP, we don't want the collective swarm to look at the same GP twice in the same 24-hour period. These are the only swarm-wide constraints.

Figure 2 shows the planner inputs and outputs. Preprocessing involves assimilating a wide range of heterogeneous input data, to produce two the 2 search spaces as planner inputs, shown in figure 2. The TP choices (left) specify the search space of commands available at each TP. This is the satellite command space describing available observation choices of the form <TP, instrument(s), angle, list of GP covered by choice> for all satellites. It's the primary search space, corresponding to the required output: a command sequence for each satellite.

The GP choices (figure 2, middle) map each GP to choices for observing it with associated model and measurement errors. This is the scientific value search space, where each GP is mapped to its observation choices as <satellite, time, command, measurement error> tuples.

The planner produces a plan for each satellite (Figure 2, right), which assigns commands for every timepoint (TP) when it can observe any ground position (GP). Each TP has 1 second duration. The objective is to reduce the aggregate prediction error by observing as many high-error GP as possible using the measurements with least error.

Planner Challenges Beyond the State of Art: This problem is combinatorically explosive. For 3 satellites over a 6-hour period, the constellation has access to a total of 764,197 unique GP distributed over 8,701 timepoints (TP). Each satellite has access to an average of 254,732 unique GP, distributed over an average of 2,900 TP. On average, each satellite has access to 87 GP per TP, with 55 command choices to view different subsets of those 87 GP. At any given second (TP) the swarm may have a choice of over 150 different commands, many of which are mutually exclusive, or cover duplicate GP, and each command is associated with a different measurement error.

Our goal for near-real time response to rapidly changing phenomena like wildfire and floods, combined with the combinatoric complexity of the problem, produces many planner challenges. We can solve this problem for a swarm of 3 satellites quickly using Dynamic Constraint Processing (DCP), but it's unknown how close to optimal the solution is.

Preprocessing filters out infeasible and undesirable choices before building the planning model, thus reducing the search space significantly. Even with these measures, the problem is very large (90 million MILP constraints for 3 satellites with 6-hour plan horizon). There are many degrees of freedom (choices of satellites, observation targets, times, and command choices).

Our goal of near real-time response seems out of reach for exact solutions, while fast heuristic solutions are available. We are interested in using MILP's exact solutions offline to evaluate and improve the online DCP approximate solutions which may meet our performance needs.

Planning Model and Methods

We now introduce formal terms which are shared by both the DCP and MILP planners. Given the following inputs:

N = the # of satellites in constellation

s_i = satellite i, $1 \leq i \leq N$

G_i = the set of all GP visible by s_i in plan horizon

T_i = Set of all times in plan horizon when satellite s_i can see any $g_j \in G_i$

$C_{i,t}$ = the set of command choices for s_i at time t, each command choice = <instrument(s), angle>, $\forall t \in T_i$

$G_{i,c,t}$ = set of Ground Positions (GP) covered by sat *i* executing command $c \in C_{i,t}$, at time t.

$err_{g,t}$ = the *predicted* error for GP g at time *t,* including weather forecast but prior to new observation. $\forall g \in G$

$err^{m}_{c,b}$ = *measurement* error for command *c* in biome type *b*

$$r_{g,c,t} = err_{g,t} - err^{m}_{c,b} \quad (1)$$
= *gpReward* = GP *g*'s error improvement

Note that $G_{i,c,t}$ is filtered to remove GP in cases when $r_{g,c,t} < 0$ (when GP prediction err at time *t* cannot be improved by any commands available at *t*).

$slew^{d}_{c_1,c_2}$ = slew time duration between angles for c_1 and c_2

Dynamic Constraint Processing (DCP): A Constraint Processing System (Dechter, 2003) is defined by a set of variables, a set of variable domains for each variable, and a set of constraints on valid variable combinations.

In our application, there are too many GP to observe them all. This is an oversubscription planning problem so it's inherently a Constraint Optimization Problem rather than pure constraint satisfaction/feasibility problem. Our DCP planner is written in Python 3.8.

Decision Variables: We define a set of decision variables $x_{i,t}$, each representing the command choice for sat s_i at time t. $\forall\, t \in T_i$, T_i= {All TP for sat s_i}. We don't model every second in the horizon. We only define $x_{i,t}$for times *t* when s_i has access to a GP. Each satellite has only about 9000 TP (seconds) out of a 6-hour plan horizon when it can see any GP, but on each TP it may have access to more than 50 GP.

Complexity: # of $x_{i,t}$ vars = O[$|T_i|$] = the # of TP.

Variable Domains: $x_{i,t} \in \{d_{i,t}\}$. The variable domain $d_{i,t}$ is the set of command choices for each $x_{i,t}$. The domain of choices for $x_{i,t}$ is the set of all command options for sat s_i at time t. $d^{s}_{t} \in$ {<instrument(s), viewAngle>}

The search space is a node tree. Each node represents a plan consisting of a set of variable assignments (command choices). Each branch/edge in the tree represents a variable assignment.

Root Node variables:
$x_{1,0}, x_{1,1}, x_{1,2}, x_{2,2}, x_{1,3}, x_{2,3}, x_{2,4}, x_{2,5}, x_{1,6}, \ldots$

Figure 3: Decision variables for the root node

Figure 3 shows and example of the decision variables for the root node. $x_{i,t}$ = the command for sat s_i at time *t.* The root node is initialized with variables for every TP for every satellite. There is 1 variable per TP per satellite. Root node variables are sorted chronologically, so all variables for time N precede all variables for times > N. This example shows that we don't model every second in the horizon. Figure 3 shows there are variables for only sat 1 at times 0, 1, and 6. There are variables for both satellites at times 2 and 3, and variables for only sat 2 at times 4 and 5. This is because those are the only times when the given sat has TP choices. We may choose to solve the variables in any order, but our default is to solve them in chronological order.

Objective: *maximize reduction of error in soil moisture predictions.* This means maximizing the sum of *gpRewards* (eq 1).

$$\textbf{maximize} \sum_{i \leq N, c \in P, t \in T_i, g \in v_{i,c,t}} r_{g,c,t} \quad (2)$$

where P = a list of commands c in a plan, and $v_{i,c,t}$= the set of all GP which are visible to s_i using command c at time t.

Equation 2 maximizes the sum of all *gpRewards* for all GP covered by all commands in the plan, for all satellites. The $r_{g,c,t}$ in (eq 2) is defined by equation (1). Equation (2) is the called the *plan score*, associated with each node in the planner's search space.

*Search Control***:** On each loop of the search process: The planner chooses a beam width of nodes to expand based on each node's *plan score* (eq. 2). The planner then chooses an an unassigned $x_{i,t}$ variable in each beam node to expand, then heuristically chooses a value (command) for the chosen variable in each of the beam nodes. Search terminates when a valid plan is found (when all variables have been assigned values without constraint violations).

Choice propagation: We use a form of Dynamic Constraint Processing (Mittal and Falkenhainer, 1990), where mutually exclusive variables are removed after each plan choice. Constraints are enforced through *choice propagation* which uses *forward checking* (Russell and Norvig, 2021) after each choice, to remove any future variable assignments which are inconsistent with that choice. For example, the 3-sec image lock is implemented as follows: when a choice is made to take an image at TP 10, then all choices for timepoints 11 and 12 are removed so nothing else will be scheduled during that 3-second hold. Choice propagation also enforces constraints for slew time and duplicate observations. After each command is selected by the planner, choice propagation dynamically removes GP's, commands, and TPs. This means the search space size is reduced significantly after each choice. Constraints are enforced on-demand, by calling software constraint handlers after each variable assignment is made. Constraints are only 'realized' for assignments selected for the plan.

The following example shows how choice propagation removes duplicate GP observations from future variables.

(a) [$x_{1,25}$: {~~L.32: [123]~~,
L.33: [436349, 436350, 436351],
P.32: [436350, 436351, 436352]]

(b) ~~[$x_{1,36}$: {P.42: [123]}]~~

Figure 4: Choice propagation examples

Figure 4 shows two examples of choice propagation to enforce the no duplicates constraint. Assume GP 123 is observed before TP 25. Case (a) shows the command choices for $x_{1,25}$, satellite 1 at TP 25. After GP 123 is observed, it is removed from the list of GP covered for every choice for all future variables. In this example, that was the only GP covered by command choice L.32, so the command L.32 is removed from the domain of choices for variable $x_{1,25}$. Case (b) shows that when the last choice is removed from a variable's domain (producing an empty domain), then the variable is removed from the node (because there are no commands available for that TP). In this case, after GP 123 is observed, it's removed from the GP list for command P.42, leaving an empty GP list. The command P.42 is removed from the domain for variable $x_{1,36}$, leaving an empty variable domain, so $x_{1,36}$ is removed from the list of open variables. Choice propagation removes variables which have no valid assignments based on the path dependencies of the current plan (node).

Heuristics: We tested a wide range of local heuristics, which sort command choices at each TP. See (Levinson et. al., 2021) for more details about DCP heuristics.

Mixed Integer Linear Programming (MILP) is an extension of Linear Programming where decision variables may be integers, often binary integers (Williams 2013).

Decision Variables:
We define 2 binary variables: $x_{i,t,c}, y_{g,i,c,t} \in \{0,1\}$

$x_{i,t,c} = 1 \leftrightarrow s_i$ executes $command\ c$ at time t (binary)
$\forall i \le N, \forall\, t \in T_i, \forall c \in C_{i,t}$

Complexity: This requires the same # of variables as $x_{i,t}$ in the DCP model ($|T_i|$), multiplied by $\sum_{i,t} |C_{i,t}|$, the sum of the number of commands per satellite $\forall\ t \in T_i$.

- O [$|T_i|\ \sum_{i\le N, t\in T_i} |C_{i,t}|$] = the total # of command choices *c* for all satellites s_i at all times *t*.

$y_{g,i,c,t} = 1 \leftrightarrow$ GP g is observed by sat i using command c at time t (binary variable).
$\forall i \le N, \forall g \in G_i, \forall\, t \in T_i, \forall c \in C_{i,t}$

- Complexity: # of y vars = (# of x vars) * (# of GP)
- O [$|G|$ x $\sum_{i\le N, t\in T_i} |C_{i,t}|$], where $G = \cup_{i\le N} \{G_i\}$ = set of all GP visible to all satellites.
- Equals complexity for $x_{i,t,c}$ times $|G|$
- This *y* variable is not part of the DCP model (the # of vars in the DCP = # TP).

duplicateGpLimitConstraints:
Ensure no more than d_{max} duplicate obs of any GP *g*

$$\sum_{i,c,t} y_{g,i,c,t} \le d_{max} \qquad \forall_g \in G_{i,c,t} \qquad (3)$$

where d_{max} = max # of duplicate observations allowed. (default d_{max} = 1)

- Complexity: O [|G| x (avg # commands/GP)]

mutexConstraints: For each satellite s_i, any pair of command choices which occur within 2 seconds + slew duration are mutually exclusive. Mutual exclusion constraints (equation 4) are created to enforce:

- One command at a time
- Image lock (each command c must be held for 3 seconds)
- Slew duration (no commands allowed while slewing)

$$x_{i,c_1,t_1} + x_{i,c_2,t_2} \le 1 \qquad (4)$$

$\forall i \le N,\ \forall t_1, t_2 \in T_i \quad : t_1 \le t_2 \le t_1 + 2 + slew^d_{c_1,c_2}$
$\forall c_1 \in C_{i,t_1}, \forall c_2 \in C_{i,t_2} \quad : c_1 \ne c_2$

- Complexity: O [$\sum_{i\le N, t_1,t_2 \in T_i:\ t2-t1\le slew^d_{c_1,c_2}} |C_{i,t}|$]

gpCoverageConstraints (constrain $y_{g,i,c,t}$ by relating each GP *g* to commands *c* which cover it)

$$y_{g,i,c,t} \le x_{i,c,t} \qquad (5)$$

$i \le N, \quad \forall\, g \in G_{i,c,t}, \quad \forall t \in T_i, \quad \forall c \in C_{i,t}$

- Complexity: O [|G| x (avg # cmds/GP)], where avg # cmds/GP = $\sum_{0\le i<N, t\in T_i} |C_{i,t}|/|G|$

Objective:
maximize $\sum_{\forall g\in G} r_{g,c,t}\ y_{g,i,c,t}$ (6)
where $r_{g,c,t}$ is defined by equation (1)

- Complexity: same as complexity for $y_{g,i,c,t}$ vars = O [$|G|$ x $\sum_{i<N, t\in T_i} |C_{i,t}|$]

The solver terminates when optimality is proven or when a specified time limit has been reached before proving optimality. Optimality is proven when the gap between MILP's the "primal" objective (6) and the "dual" objective reaches zero (within some tolerance).

Evaluation and Comparison

The GP reward (equation 1) is the basis for the plan score metric (objective) for our tests comparing DCP and MILP methods. Note the similarity between MILP objective (6) and DCP objective (2) and the GP Reward (1). This shared metric, along with identical input data, enable the apples-to-apples comparison between the two methods.

Case #	# sats	Plan Horizon (secs)	# TP	# GP visible	#cmd /TP	# GP obs	# Vars	#Constraints	Time to best sol (* = optimal)	Time to prove optimal	Makespan (# commands in plan)
1	3	21,600	1,705	968	12.55						
MILP						1,032	26,959	914 K	*** 156 s**	156 s	403
DCP						948	1,705		7 s		449
2	1	1,000	900	666	70.98						
MILP						620	89,118	14.4 M	*** 5 h**	16 h	195
DCP						666	900		5 s		254
3	1	1,800	1,468	1,181	55.83						
MILP						1,212	122,675	16.8 M	*** 13 h**	38 h	295
DCP						1,181	1,468		8 s		377
4 (15%)	1	21,600	7,463	55,779	53.85						
MILP						4,010	244,207	20.8 M	*** 10.7 h**	10.8 h	1,468
DCP						3,113	7,464		1.5 m		1,656
5 (15%)	3	7,200	7,527	47,929	52.41						
MILP						4,200	244,363	20 M	*** 24 h**	45.2 h	1,473
DCP						3,140	7,527		2.5 m		1,636
6	3	21,600	8,701	764,197	55.46						
MILP						***DNF***	1,856,154	90 M	***DNF***	***DNF***	***DNF***
DCP						16,732	8,701		28 m		6,104

Table 1: Comparison of scenario search space complexity and branching factors

All experiments were run on a 2020 MacBook Pro 13-inch, 2.3 GHz Quad-Core Intel Core i7 processor, 32 GB RAM. The MILP solutions were generated using Gurobi 9.5 (Gurobi, 2022). All data sets and code will be released as open source.

Table 1 shows details of the search complexity and solver performance for each scenario. The table columns are as follows: Case # is followed by *# of satellites* in the constellation, the *plan horizon,* and the *# of TP in the horizon*, then *#GP visible* (number of GP visible) during horizon. The next column, *#cmd/TP* is the average number of command choices at each TP (the average of all $|C_{i,t}|$). *# GP obs* is the # of GP observed by the commands in the plan. *# Vars* is the # of decision variables created. *# Constraints* is the # of constraints created. Constraints are pre-enumerated only for the MILP solver so there are no entries for DCP. *Time to best solution* is the solver time required to find the best solution (which is optimal for MILP, but suboptimal for DCP). *Time to prove opt* is the time it takes the MILP solver to prove the best solution it has found is optimal. Note how long it takes to prove optimality after the optimal solution is found. The last column, *makespan*, is the # of commands in the plan.

We compare the DCP and MILP methods using six test cases (Table 1). Note there are far fewer rainy GP than GP where it hasn't rained, so the search space for case 1, the only case with rainy GP, is far smaller than all other cases.

Case 1: 3 satellites, rainy GP, plan horizon = 6 hours
Case 2: 1 satellite, no rain, horizon = 1000 sec (~17 mins)
Case 3: 1 satellite, no rain, horizon = 1800 secs (30 mins)
Case 4: 1 satellite, no rain, horizon = 6 hours, top 15% most needy GP only. This case includes only the top 15 % of GP with largest model error (sorted in decreasing model error). This percentage was selected empirically such that the scenario produced around 20 million constraints (for acceptable solver times < 50 hours).

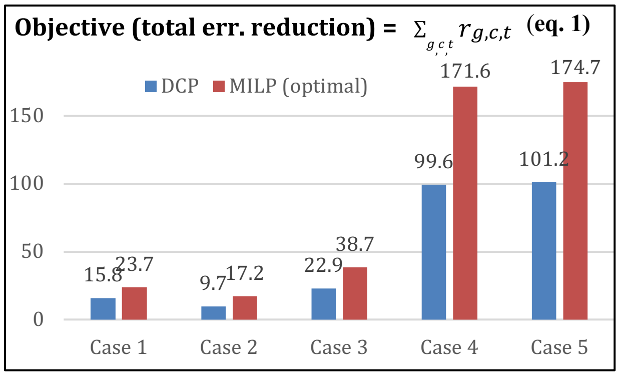
Case 5: 3 satellites, no rain, horizon = 2 hours, Top 15 % most needy GP only (like case 4).

Cases 4 and 5 are 'triage' scenarios where the top 15 percent of GP which need the most help (have the largest model error) are prioritized over less 'needy' GP. This helps to separate the wheat from the chaff so the solver spends less time trying to optimize the chaff which we may have limited effect on the objective.

Case 6: 3 satellites, no rain, horizon = 6 hours. This case can be solved by DCP but we haven't been able to solve it with the MILP model because it includes 90 M constraints and resulting the file is too large to write out, which is required for us to pass it to Gurobi. MILP entries for this case are marked DNF to indicate that test did not finish.

The cases are roughly in order of increasing complexity (measured in # of constraints). The difference between rainy and non-rainy is seen comparing the *# GP visible* column for case 1 (rainy case) vs. case 6 (non-rainy). Both cases have 3 satellites and a 6-hour planning horizon. Note that case 1 has only 968 visible GP vs. 764,197 for case 6. This shows the larger number of observation opportunities for non-rainy GP.

Increasing complexity can also be seen by noting how the *time to best sol* and *time to prove optimal columns* scale between cases 2 and 3. The only difference between cases 2 and 3 is the horizon increases from 17 minutes to 30, but time to prove optimal increases from 16 hours to 38-hours, while DCP case 3 time barely increases at all and remains less than 10 seconds. This shows superior performance and scaling of DCP vs. MILP on larger problems.



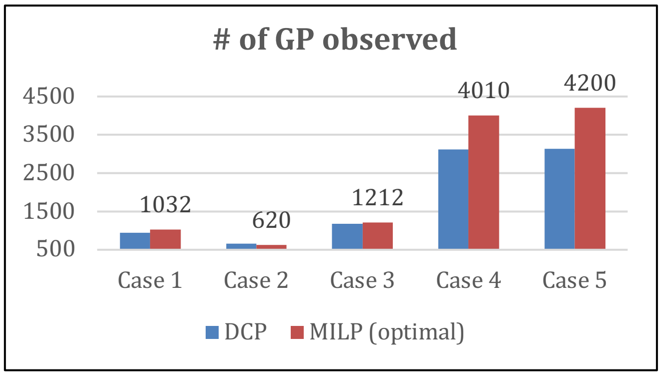
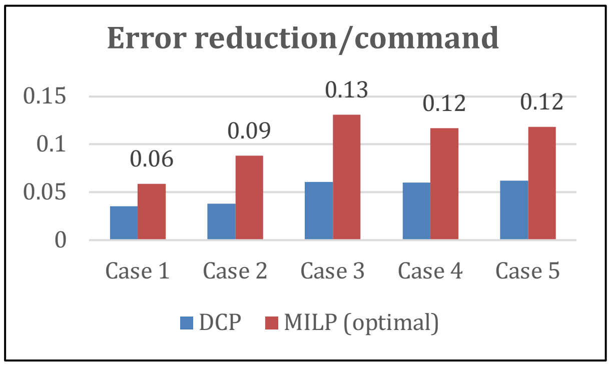


Figure 5: Comparison of DCP and MILP objective scores (left) and # of GP observed (right)



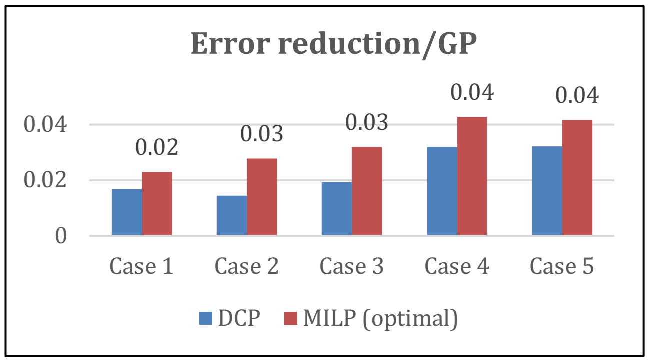


Figure 6: Efficiency metrics show average error reduction per command (left), and # per GP (right)

Figure 5 (left) compares the objective values achieved by both planners on the 5 cases which we could solve with MILP. All MILP objective values are optimal. DCP achieves 67% optimal for case 1, 56% for case 2, 59% for case 3, and 58% for case 4, and 57% for case 5. The objective value for DCP on case 6 is 286.94, so much larger than all other cases, it's not included in figure 5 for readability.

Figure 5 (right) shows MILP solutions observe more GP for all cases, except case 2. Table 1 shows in case 6, DCP observes 16,732 GP in 6 hours, much more than any other case, so it was omitted from figure 5 for readability.

Figure 6 shows the MILP solution is always more efficient than DCP, reaping higher objective rewards per command. Figure 6 (left) shows the average error reduction (objective contribution) from each command. Figure 6 (right) shows the average error reduction per observed GP. Table 1 shows MILP plans always have fewer commands (shorter makespan), while still achieving superior results.

Analysis: MILP performance is dominated by the exponential scaling in the number of constraints for some scenarios. The number of mutex constraints are dominated by deconfliction within 25-second windows. This means the number of mutex constraints is a function of how many TP and choices occur within any given 25 sec time window. MILP optimality ensures no local minima, but scaling creates so many more variables and constraints that solve time becomes impractical (> 3 days) for desired plan horizons of even 6 hours.

Table 1's cases illustrate the edge of problems we can solve with MILP. If we increase the horizons for any of the non-rain cases, the model becomes so large due to so many constraints, that our Mac laptop runs out of memory and/or takes longer than our time limit of 50 hours, so we don't yet know the optimal solution for those cases.

Table 2 summarizes the pros and cons for each method. DCP benefits include constraint and heuristic expressiveness and flexibility, and solver time. MILP offers the important "certificate of optimality" at the cost of impractically long solver times. DCP is prone to local minima but has a far smaller search space (fewer vars and constraints). The DCP variable domain is symbolic vs. the MILP's requirement for quantitative domains. For example, in DCP, a variable $x_{i,t}$ may be assigned a value such as "L.32", from the domain shown as "command choices" in figure 2 (left). Another difference is the DCP planner only considers constraints which are required for a given plan. This is in contrast with MILP where 10's of millions of constraints are preconstructed and the solver must consider them, although they involve variable assignments which are mutually exclusive with choices selected for any given plan.

DCP
Pro
• Search control over of node, var, and val choices
• Flexible for model prototyping. Can model any constraint and use symbolic (vs. numeric) vars.
• Leverages domain heuristics. Easy to swap between different heuristics.
• Fast, amenable to running onboard
• Explanations for why a GP is not in plan.
• Dynamic constraints improve performance
Con
• Suboptimal solutions
• Subject to local minima and path dependencies
MILP
Pro
• Provably optimal solutions
• Relies on 3rd party solver (has benefit of robust heavily tested tool)
Con
• Slow
• Limited modeling flexibility
• Difficult to include domain-specific heuristics
• Limited to no explainability
• solvers (many orders of magnitude different).
• Requires predefining all constraints in advance (10's of millions must be created but most are not required for any specific solution).
• Requires all variable domains be quantitative.

Table 2: Trades between DCP and MILP methods

Related Work

(Küçük and Yıldız, 2019) present a constraint programming approach to observation scheduling for a single agile satellite. However, their formulation is extremely MILP-like, with two binary decision variables ∈ {0,1}, compared to our DCP approach where the decision variables have symbolic values like "P.34". They present results using the commercial product CP Optimizer (CP Optimizer, 2022) for a small problem with a maximum of 55 GP targets.

MILP has previously been used for scheduling observations for agile satellite constellations. See (Wang et al. 2021) for a good survey of the field.

(Chen et al., 2019) present a MILP for multiple satellites with multiple heterogeneous sensors and slew constraints. Their MILP formulation differs from ours in several ways. For example, instead of modelling each satellite explicitly as we do, they model a pool of resources (instruments) which is the set union of all instruments on all satellites. The problem scale is significantly smaller than ours. The total # of visible GP in their tests range from 100 to 1000 GP in a 24-hour horizon, while ours range from 968 to 764,197 GP in a 6-hour horizon (see Table 1).

(Kim et al., 2020) present a MILP solution for scheduling multiple agile satellites, with slew time constraints and multiple heterogeneous instruments. The objective reward (profit) for each GP is a static input rather than a function of time, instrument, and angle as with D-SHIELD. They present results for a small problem with maximum of 100 visible GP. They also present a heuristic preprocessing method to prune undesirable choices before creating the MILP. This is similar to our 'triage' cases 4 and 5 where we sort GP by decreasing error and prune out the bottom 85% (keep the top 15%) of GP with the highest error, then we create the MILP for only that top 15%.

Several differences distinguish our work from the above. First, D-SHIELD uses very high-fidelity engineering models of satellite physics to calculate slew time (Sin et al. 2021). Instrument physics models are used to calculate instrument measurement errors, which is used in our objective. A second difference is we have many more target GP. For 3 satellites with 24-hour plan horizon, we have 764,197 potential targets (visible GP) and 16,732 observed GP. These are orders of magnitude larger numbers than we've seen in other work. A third difference is an extra degree of freedom between observations and targets which adds complexity. While other systems have a 1:1 mapping between each observation and target GP, each D-SHIELD observation covers ~2.7 unique GP.

Future Work and Conclusion

We will work on improving DCP towards the MILP-proven optimal solutions and explore alternative MILP formulations. There are several constraints which are yet to be developed. DCP includes an energy model to track energy consumed by instruments and slewing and energy produced by solar panels, to ensure no satellite dips below a minimum energy. The MILP formulation for that extension is still being developed. We also plan to integrate new sensors and multiple, independently developed planners for downlinking data and for intersatellite communications.

We have described an important climate change monitoring application which requires online planning capabilities beyond the state of art. We have presented two methods to solve the problem, DCP and MILP, and described the practical and challenging trades between them. Our conclusion is it makes sense to use them together because they serve complementary purposes. DCP offers practical solve times for suboptimal solutions, while MILP provides "ground truth" for provably optimal solutions at the cost of solve times which are impractical for our near real-time requirements. Additionally, the process of comparing the two methods head-to-head on identical inputs with identical metrics, helped to identify implementation asymmetries which led to improvements on both sides.

References

Augenstein, S., Estanislao, A., Guere, E., and Blaes, S. 2016. “Optimal scheduling of a constellation of earth-imaging satellites, for maximal data throughput and efficient human management". Proceedings. of ICAPS 2015, 26(1). pages 345-352.

Barnsley, M., Settle, J., Cutter, M., Lobb, D., and Teston, F. 2004. “The PROBA/CHRIS mission: A low-cost smallsat for hyperspectral multiangle observations of the earth surface and atmosphere,” Geosci. Remote Sens. IEEE Trans. vol. 42, no. 7.

Chen, X., Reinelt, G., Dai., G, Spitz, A. 2019. A mixed integer linear programming model for multi-satellite scheduling. In European Journal of Operational Research. Volume 275, Issue 2, 1 June 2019, pages 694-707.

Chien, S.; Doubleday, J.; Mclaren, D.; Davies, A.; Tran, D.; Tanpipat, V.; Akaakara, S.; Ratanasuwan, A.; and Mandl, D. 2011. Space-based Sensorweb Monitoring of Wildfires in Thailand. In International Geoscience and Remote Sensing Symposium (IGARSS 2011), Vancouver, BC, July 2011.

Chien, S.; Doubleday, J.; Thompson, D. R.; Wagstaff, K.; Bellardo, J.; Francis, C.; Baumgarten, E.; Williams, A.; Yee, E.; Stanton, E.; and Piug-Suari, J. 2016. Onboard Autonomy on the Intelligent Payload EXperiment (IPEX) CubeSat Mission. Journal of Aerospace Information Systems (JAIS). April 2016.

Chien, S, Mclaren, D., Doubleday, J., Tran, D., Tanpipat, V., and Chitradon, R. 2019. Using Taskable Remote Sensing in a Sensor Web for Thailand Flood Monitoring. Journal of Aerospace Information Systems (JAIS), 16(3): 107-119. 2019.

Chien, S. A.; Davies, A. G.; Doubleday, J.; Tran, D. Q.; Mclaren, D.; Chi, W.; and Maillard, A. 2020. Automated Volcano Monitoring Using Multiple Space and Ground Sensors. Journal of Aerospace Information Systems (JAIS), 17:4: 214-228.

CP Optimizer. 2022. https://www.ibm.com/analytics/cplex-cp-optimizer. Accessed: March 23, 2022.

Dechter. R, 2003. Constraint Processing. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pages 25-26.

Fruth, Thomas, Christoph Lenzen, Elke Gross, and Falk Mrowka. 2019. "The EnMAP Mission Planning System." In Space Operations: Inspiring Humankind's Future, Springer, Cham, 2019

Gurobi. 2022. https://www.gurobi.com Accessed: March 3, 2022.

Jian, L., and Cheng, W. 2008. “Resource planning and scheduling of payload for satellite with genetic particles swarm optimization,” in Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). pp. 199–20

Kim, J., Ahn, J., Choi, H. 2019. Task Scheduling of Agile Satellites of Transition Time and Stereoscopic Imaging Constraints. Journal of Aerospace Information Systems. Vol. 17, No. 6. 2020.

Krieger, G., Alberto, M.,, Hauke F., Hajnsek, I., Werner, M., Younis, M., Zink, M. 2007. "TanDEM-X: A satellite formation for high-resolution SAR interferometry." IEEE Transactions on Geoscience and Remote Sensing 45, no. 11 (2007): 3317-3341

Küçük, M., Yıldız, S. 2019. A Constraint Programming Approach for Agile Earth Observation Satellite Scheduling Problem. 9th Int'l Conference on Recent Advances in Space Technologies (RAST), pp. 613-617

Levinson, R., Nag, S., and Ravindra, V. 2021. Agile Satellite Planning for Multi-payload Observations for Earth Science, Proceedings of the International Workshop on Planning and Scheduling for Space (IWPSS). 2021.

Linnabary, R., O’Brien, A., Smith, G., Ball, C., and Johnson, T. 2019. “Open Source Software For Simulating Collaborative Networks Of Autonomous Adaptive Sensors,” in IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium, pp. 5301–5304.

Mittal, S., Falkenainer, B., 1990. Dynamic Constraint Satisfaction Problems". Proceedings of AAAI-90.

Nag S., et al., 2019. “Autonomous Scheduling of Agile Spacecraft Constellations with Delay Tolerant Networking for Reactive Imaging,” presented at the International Conference on Planning and Scheduling (ICAPS) SPARK Applications Workshop, Berkeley, California, U.S.A., 2019

Nag, S., Moghaddam, M., Selva, D., Frank, J, Ravindra, V., Levinson, R., Azemati, A., Aguilar, A., Li, A., Akbar, R., 2020. D-Shield: Distributed Spacecraft with Heuristic Intelligence to Enable Logistical Decisions, Proc. of the 2020 IEEE International Geoscience and Remote Sensing Symposium.

Nag, S., Moghaddam, M., Selva, D., Frank, J., Ravindra, V., Levinson, R., Azemati, A., Gorr, B., Li, A., Akbar, R. 2021. "Soil Moisture Monitoring using Autonomous and Distributed Spacecraft (D-SHIELD)", IEEE International Geoscience and Remote Sensing Symposium, Virtual, July 2021

NIDIS. *Soil Moisture.* 2021. https://www.drought.gov/topics/soil-moisture. Accessed: Dec 12, 2021.

Ravindra, V., Ketzner, R., and Nag, S., "Earth Observation Simulator (EO-Sim): An Open-Source Software for Observation Systems Design," 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, 2021, pp. 7682-7685.

Russell, S., Norvig, P. 2020. Artificial Intelligence: A Modern Approach. Fourth Edition. Pearson. page 194.

Sin, E., Arcak, M., Nag, S., Ravindra, V., Li, A., & Levinson, R. (2021). Attitude Trajectory Optimization for Agile Satellites in Autonomous Remote Sensing Constellations. In AIAA Scitech 2021 Forum (p. 1470).

Wang, X., Wu, G., Xing, L., Pedrycz, W. 2021 "Agile Earth observing satellite scheduling over 20 years: formulations, methods, and future directions". IEEE Systems Journal. 2021. Vol. 15, pp 3881-3892.

Werninghaus, Rolf, and Stefan Buckreuss. 2009. "The TerraSAR-X mission and system design." IEEE Transactions on Geoscience and Remote Sensing 48.2 (2009): 606-614.

Williams, H. P., 2013. Model Building in Mathematical Programming, Fifth Edition. Wiley and Sons Publications. page 155.

Wörle, Maria Theresia, Christoph Lenzen, Tobias Göttfert, Andreas Spörl, Boris Grishechkin, Falk Mrowka, and Martin Wickler. 2014. "The Incremental Planning System GSOC's Next Generation Mission Planning Framework." In SpaceOps 2014 Conference, p. 1785. 2014.