# Planning Satellite Swarm Measurements for Climate Models: Comparing Dynamic Constraint Processing and MILP methods

12/9/21

**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

**Abstract**

We present D-SHIELD, a challenging climate science application to plan coordinated measurements (observations) for a constellation of satellites, each containing two different sensors, each with 61 pointing angle options. The L-band and P-band radar sensors collect data fed into a soil moisture model which tracks and predicts soil moisture across 1.67 million Ground Positions (GP). Soil moisture is an important predictor of wildfires, and then a predictor of floods, landslides and debris flow after a fire.

Each measurement covers multiple GP due to the sensor footprint. Each GP has a "model error" which represents the uncertainty of the the soil moisture state prediction. Model error changes at different rates for each GP as the time since last observation increases and after significant events like rain. The planner's goal is to select measurements which maximize soil moisture model improvement (reduce model uncertainty).

This problem is combinatorically explosive, involving many degrees of freedom for planner choices. Good domain heuristics can find solutions within a reasonable time for our application needs but cannot be proven optimal. In this paper we compare two different 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. We demonstrate and discuss the trades between DCP flexibility and performance vs. MILP's promise of provable optimality.

## Climate Science Problem and Application

Soil moisture is important to manage fires both before and after they occur. Soil moisture predicts fire locations, then after they occur it predicts flooding, debris flow and landslides. The Planner must produce a coordinated plan for a constellation of satellites to observe ground positions (GP) based on predicted model and measurement errors. The planner's objective is to maximize soil moisture model improvement, where model improvement means reducing model uncertainty, which is the sum of model improvement for all ground positions (GP) observed.

**Important need for planning to solve problem**: Rapid responses to quick changing natural phenomena (like fire) requires fast replanning with fewer humans in the loop to direct agile spacecraft to optimal observation location with the optimal sensor selection. Offline engineering models of spacecraft, sensors and orbits do not include models of duty cycles of the expected plans to be executed, so cannot simulate/predict the system behavioral/dynamic requirements. Agile satellites that can change viewing angle provide new opportunities for opportunistic planning, compared to satellites where the sensor angle is fixed. Planning is also required to optimize for minimizing error compared to the relatively simple metric of maximizing the # of GP observed regardless of model error and measurement error.

**The D-SHIELD application** (Levinson et. al., 2021) uses a (proposed) constellation of satellites looking at Earth to reduce global soil moisture model uncertainty by making observations that target spatio-temporal points of rising prediction error, which accurate data can alleviate. Such uncertainty reduction benefits accurate models for floods, wildfires, vegetation drydowns, etc.

D-SHIELD is a challenging climate science application to plan coordinated measurements (observations) for a constellation of satellites, each containing two different sensors, each with 61 pointing angle options. The L-band and P-band radar sensors collect data fed into a soil moisture model which tracks and predicts soil moisture across 1.67 million Ground Positions (GP). Soil moisture is an important predictor of wildfires and of floods, landslides and debris flow after a fire.

Each measurement covers multiple GP due to the sensor footprint. Each GP has a "model error" which is the uncertainty of the prediction of the soil moisture state at a specified time (RMSE). Model error changes at different rates for each GP as the time since last observation increases and also after significant events like rain. The planner's goal is to select measurements which maximize soil moisture model improvement (reduce model uncertainty).

The overall goal is to demonstrate a system which continually updates a hydrologic land surface model with external forcing functions (e.g. precipitation) and dynamically schedules new observations to improve the model where the error is greatest, such as right after rain occurs, or places which have not been observed recently, using instrument parameters that minimize retrieval errors. The dynamic soil moisture model detects regional model quality degradation in near real-time and provides input to the planner about the highest priority ground positions (GP) to observe next from a science perspective. This paper focuses on the D-SHIELD planner more than the science model.

Each satellite in the constellation includes at least 2 different radar instruments (L-band and P-band) to take images of ground positions (GP). Each image typically covers multiple GP. Each satellite has a set of access times when it can view various GP, based on its orbit. These access times are called timepoints (TP). The planner creates an "observation plan". It must decide which satellites look at which GPs, at what times, with which instruments using which available viewing angles. For 3 satellites over a 6-hour period, the constellation sees a total of 1,062,777 GP distributed over 29,700 TP. Each satellite can see an average of 354,259 GP, distributed over an average of 9,900 TP. The planner produces satellite plans. For every satellite, the planner must choose <instrument, angle> measurement pairs (or choose no measurement) for every TP when it can see any GP.

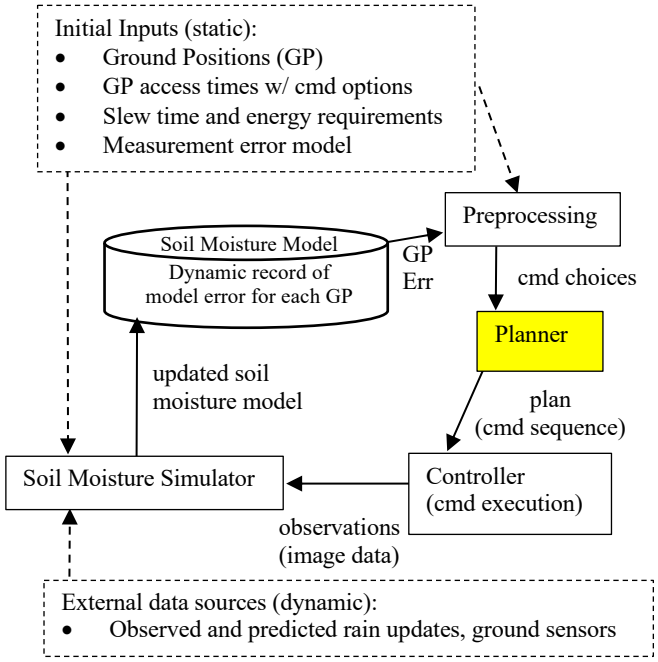


**Figure 1: D-SHIELD architecture**

Figure 1 shows the D-SHIELD system. The solid lines show the closed-loop control flow while the dashed lines show external data sources used to update the model. The process is initialized with inputs based on the satellite orbits and specifications for the spacecraft and instruments, which are used to determine available observation times, along with slew time and energy requirements. Those raw inputs go through *pre-processing* to generate the planner input files. The planner searches through the space of all available observation times and decides what to look at, when to look at it, and how to look at it. The planner produces a sequence of commands for the Controller to execute (command the instruments to take the images).

The (simulated) spacecraft executes the plan, collects the observation data and passes that to the Soil Moisture Simulator, to update the model based on the new observations. The Soil Moisture Model is a dynamic database which maintains a record of the current model error for each GP.

**The planner's job** is to create a coordinated multi-satellite observation plan which improves the model quality by observing the GP with the highest model error using measurements with the least error. Each plan step is an observation command of the form <time, instrument(s), viewing angle>, which specifies the time when one or both instruments will take images at the given viewing angle.

**Model and Measurement Error Tables:** Planner input includes a model of predicted soil moisture for the next 24 hours (the "model 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). Measurement error is the predicted RMSE for using the given (combination of) instruments at a given angle. D-SHIELD uses high-fidelity models of spacecraft and soil moisture dynamics, combined with a constraint satisfaction planner to generate new observation plans for all satellites in the constellation with the objective of improving model quality. Model quality is inversely proportional to the model error associated with each GP. Model error for each GP is a combination of a given prior model error (predictive uncertainty) and measurement error (soil moisture estimate retrieval error) from new observations. Model error increases over time without new observations and increases when rain occurs. Measurement error is a function of which instrument (L-band or P-band) is used, the viewing angle, the type of ground cover (e.g., barren, shrubs, forest, croplands), and other ancillary parameters.

**Constraints:** The planner must enforce the following constraints:
Deconfliction constraints (each satellite can do only one thing at a time). The image lock and slew time constraint constraints are enforced per satellite.

Image Lock - Each observation requires the instrument to hold its viewing angle for 3 seconds so that a stripmap image can be created. This blocks out slewing to another viewing angle during that 3-second image lock period.

Slew time constraints - The satellite must slew to change viewing angles. There are constraints on how quickly it can change viewing angles, depending on the slew magnitude. Changing viewing angle takes a different amount of time depending on the combination of initial angle and target angle. This is called the slew time constraint. The planner must ensure there is enough time to slew between each observation.

Duplicate observations (enforced swarm-wide) - We do not want the collective swarm to look at the same position twice in the same 24-hour period to cover more unobserved locations. There are exceptions for serendipitous cases when we aim at one GP but capture others in the same image, and for intentional cases when we plan a follow-up observation. These are the only swarm-wide constraints.

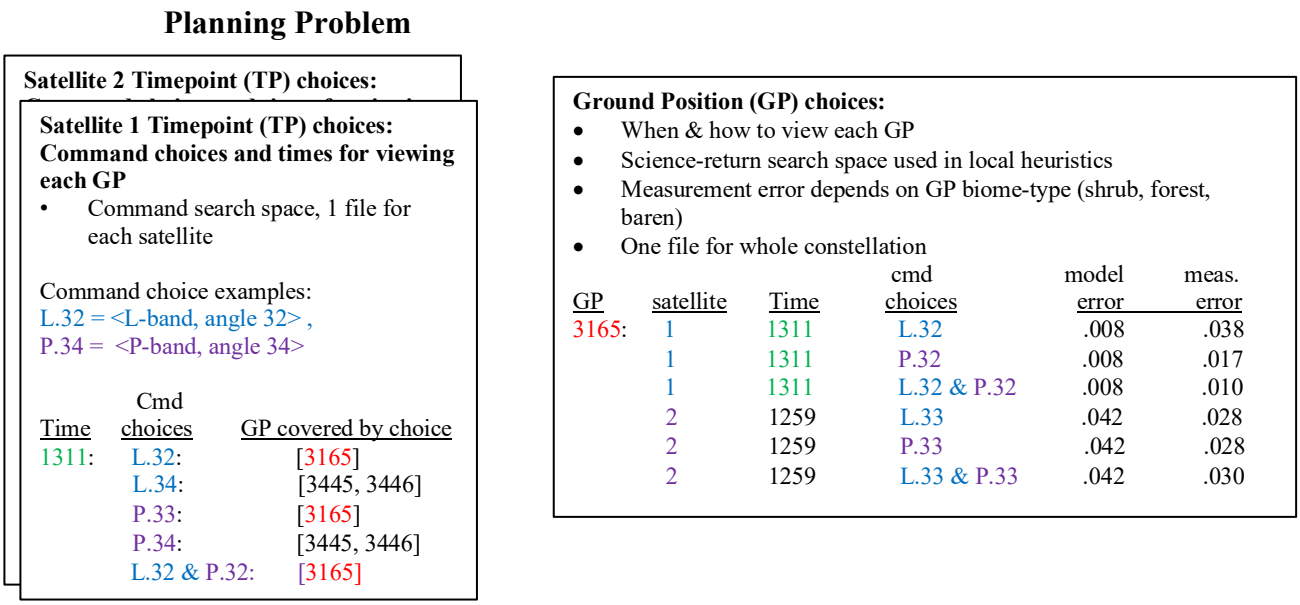


**Figure 2:** Preprocessing produces TP choice files for each satellite (left) and a single GP choice file (right) for the swarm.

Preprocessing involves assimilating a wide range of heterogeneous input data, to produce two 2 search spaces as planner inputs. The TP choices (left) specifies the space of commands available at each TP. The GP choices (right) maps each GP to choices for observing it with associated model and measurement errors.

The TP space is the satellite command space describes commands: <sat, time, inst, angle> for all sats, times, instrument, angles. This is the primary search space which corresponds to the required output: an observation plan for each satellite. This is the physical search space.

GP command space is the scientific search space, where GP are sorted in decreasing order of scientific value. In our case this means the GP with the highest model error which can benefit most from high quality measurements. The GP space is used as a local heuristic to sort the choices in the GP space. See (Levinson et. al 2021) for more details about how the local heuristics work in the DCSP.

The planner's job is to assign commands for every satellite for every timepoint (TP) when it can observe any ground position (GP). The TP when a satellite can observe a GP is called an *access time*. There are too many GP to observe them all so this is inherently an optimization problem. The objective is to reduce the average model error by observing as many high-error GP as possible.

## Planner challenges beyond the State of Art

The desire for near-real time response to rapidly changing climate phenomena, combined with the complexity of the problem produces many planner challenges. Our concept of operations involves at least 6 satellites and a 3 to 6 hour plan horizon. The larger problems can be solved by DCP in a time reasonable for our needs, but we don't know how close to optimal the solution is. The MILP cannot be proven optimal on those large problems within 40 hours running on Gurobi, so we don't yet know the optimal solution for those cases. For example, on a problem a tiny fraction of the size of our target, the MILP solver can take 13 hours to find an optimal solution and another 25 hours to prove it was optimal.

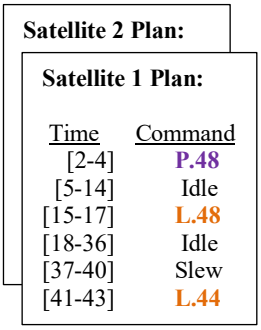


**Figure 3:** Planner Output is a plan for each satellite containing scheduled observations and slew actions

Our target scenario involves generating plans for a swarm of 6+ satellites for a 6 hour plan horizon, possibly concatenated into a 24-hour plan horizon (21,600). Our actions are modeled on the scale of 3 to 25 seconds. At any given second the swarm may have a choice of 50 different commands, many of which are mutually exclusive.

Problem size is huge (90 million MILP constraints for 1 satellite with 6 hour plan horizon). There are many degrees of freedom (branching factor) for available command choices and observation targets. Preprocessing is required to prune infeasible choices. Problem difficulty increases with increasing number of command choices/TP, and when many TP are clustered together within 25 seconds. Close

temporal proximity between TP requires more deconfliction constraints. The closer the TP are the more constraints are required to deconflict them, up to the maximum slew time of 25 seconds.

Near real-time response to rapidly changing phenomena such as wildfires and mudslides. This requirement seems out of reach for exact solutions while heuristic solutions are available. We are interested in using MILP's exact solutions in non-real-time to evaluate the DCP approximate solutions which may meet our performance needs. Planners running onboard satellites have limited resources which increases the challenge.

## Planning Model and Methods

We now introduce formal terms which are shared by both the DCP and MILP solutions. 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, s.t.: $s_i$ 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 = <instruments, params>, $\forall t \in T_i$

$G_{i,c,t}$ = set of Ground Positions (GP) covered by sat *i* executing command $c \in C_{i,t}$. *Note: $G_{i,c,t}$ is filtered to remove gp in cases where $r_{g,c,t} < 0$ (model err at time t cannot be improved by any commands available at t.)*

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

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

$$r_{g,c,t} = max(err_{g,t} - err^m_{c,b}, 0) \quad (1)$$

= *GP reward* = GP *g*'s error improvement

If $r_{g,c,t} < 0$ then taking an image with command c would increase model error for GP $g$. Observations which increase error are discarded, so $r_{g,c,t}$has a lower bound of 0. Equation (1) is the basis for the plan score metric (objective value) for our tests with both the DCP and MILP methods.

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

**Constraint Processing**

A Constraint Processing System (Dechter 2003) is defined generically as:

- Set X of variables $\{x_i, .. x_n\}$
- Set D of variable domains $\{d_i, .. d_n\}$ for each variable
- Set C of constraints on legal variable combinations
- Satisfiability requirement: Find a consistent set of variable assignments for all variables for hard constraints

There are too many GP to observe them all. This is an oversubscription planning problem so it's inherently a Constraint Optimization Problem (COP) rather than pure constraint satisfaction/feasibility problem.

Decision Variables: We define a set of decision variables $x_{i,t}$ , each representing the command choice for sat $s_i$ at time t. $\forall \boldsymbol{t} \in T_i$, $T_i$= {All access times (TP) for sat $s_i$}. This model is "sparsely time sliced", meaning we only define $x_{i,t}$for times when satellite S has access to view a GP. Each satellite has only about 9000 seconds out of a 21,600 second plan horizon (6-hours) when it can see any GP, but on each of those 9000 seconds it can see many GP.
Complexity: # of $x_{i,t}$ vars = O[ $|T_i|$ ]

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 at time t. $d^s_t \in$ {(<instrument1, viewAngle1>, <instrument2, viewAngle2>)}. The variable domain is symbolic compared to the MILP's requirement for quantitative domains. Here, a variable $x_{i,t}$ may be assigned a value from the domain shown as 'cmd choices' in figure 2, such as L.32 & P.32.

The search space is a node tree. Each node is a plan consisting of a (possibly partial) set of variable assignments (cmd choices). Each branch/edge in the tree represents a variable assignment. A node with all valid assignments for all of its variables is a solution node. Each node contains:

- 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 GP access times (TP) for sat $s_i$}
- Set of variable domains $d_{i,t}$ representing command choices for each $x_{i,t} \forall \boldsymbol{t} \in$ {access times}. The domain of choices for $x_{i,t}$, is the set of all command options for sat $s_i$ at time t.

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

**Figure 4**: Decision variables for the root node

Figure 4 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. The

root node variables are sorted chronologically, so all variables for time N precede all variables for times greater than N. There are variables only for the times when the satellite as access to a GP. This example 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**

The objective is to: *maximize reduction of model error.* This means maximizing the sum of *gpRewards* (1) for all GP covered by all satellites' commands in the plan. This is the aggregate reward for including command $c$ in the plan (the sum of rewards for all GP covered by c, for all $s_i$).

Our objective is to maximize the sum of all *gpRewards* $r_{g,c,t}$, for all commands in the plan:

$$\textbf{maximize } \Sigma_{i \le \mathrm{N}, c \in \mathrm{P}, t \in T_i, g \in v_{i,c,t}} \; r_{g,c,t} \qquad (2)$$

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. The $r_{g,c,t}$ in (2) is defined by equation (1). Equation (2) is 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 the plan score (2) described above. Then the planner chooses a variable in each node to expand, then finally selects a value for each variable in the beam nodes. Domain-specific implementations of sortNodes(tree), sortVariables(node), sortValues(variable) provide flexible methods to control the order of node, variable, and value selection. The search terminates when a valid plan is found. A valid plan is when all variables in a node have been assigned values which violate no constraints.

**Dynamic Constraint Processing (Choice propagation)**

We use a form of Dynamic Constraint Processing (Mittal and Falkenhainer 1990) because mutually exclusive variables are removed after each plan choice, so the solution does not require assignments for all variables created initially. Additionally, we do not need to pre-enumerate all possible constraints. Enforcing a constraint involves calling a constraint handler method after each plan choice, so we never consider constraints which are not required for the plan.

Constraint handlers: Constraints are enforced through *choice propagation* which uses *forward checking* (Russell and Norvig 2021) after each choice to remove any future variables which are inconsistent with it. For example, the 3-second image lock is implemented as follows: when a decision is made to start taking an image at tick 10, then all choices for timepoints 11 and 12 are removed so nothing else will be scheduled during that 3-second hold. Choice propagation is also used to enforce constraints for slew time and duplicate observations. After each command is selected by the planner, choice propagation dynamically removes GP's, commands, and TP. This means the search space size is significantly reduced after each choice.

This is form of dynamic constraint problem [Mittal and Falkenhainer, 1990]. Instead of pre-enumerating all constraints between all variables, our constraints are enforced by calling software constraint handlers after each variable assignment is made. The constraints are only 'realized' for variable assignments selected for the plan. This is in contrast with MILP where 10's of millions of constraints must be preconstructed and the solver must consider them, even though they involve variables which are mutually exclusive with choices which are not in the final plan.

Deconfliction constraints are applied only to variables for the same satellite, while the duplicate observation constraint applies across all satellites. This means the image lock constraint for satellite 1, requires that only satellite 1 hold its position. On the other hand, if satellite 1 observes GP 10, then satellite 2 is constrained to ***not*** observe GP 10. The following choice propagation example shows how it is used to remove 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: [253]}~~]

**Figure 5: Choice propagation examples**

Figure 5 shows two examples of choice propagation to enforce the no duplicates constraint. 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. In this case, after GP 253 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. This is different from a pure CSP system where all variables must receive a valid assignment, and a variable with an empty domain

indicates infeasibility. In our case, choice propagation removes variables which have no valid assignments based on the path dependencies of the current plan (node). Choice propagation also updates the energy model and the plan score for each node.

**Heuristics**: We have developed and tested a wide range of local heuristics, which sort command choices at each TP. These heuristics include maxErrReduction, gpRankedChoice, maxGpCount.

**Explainability**: GP which are not included in the plan are annotated with explanations about why they were excluded (a constraint or preference involved in the decision). A GP may be excluded due to rain, poor viewing angle, redundancy, or when the satellite is busy doing something else such as slewing to a new viewing angle.

**MILP**

Given:

$G = \cup_{i \le N} \{G_i\}$ = The set of all (unique) GP visible to all sats

$T = \cup_{i \le N} \{T_i\}$ = The set of all TP (available measurement times) for all sats

**Decision Vars:**

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

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

- Complexity: O [ $\sum_{i \le N, t \in T_i} |C_{i,t}|$ ] = the total # of cmd choices *c* for all sats *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).

$\forall g \in G, \forall\, t \in T_i, \forall c \in C_{i,t}$

- Complexity: O [ $|G|$ x $\sum_{i \le N, t \in T_i} |C_{i,t}|$ ]
- Equals complexity for $x_{i,t,c}$ times $|G|$
- This *y* variable is not part of the CO model.

**Constraints:**

Deconfliction constraints create constraints between mutually exclusive actions in overlapping time windows. For each satellite, each command must hold for 3 seconds during which no other command can be scheduled, and when pointing angle change, no commands can be scheduled while slewing between angles. Slew time ranges from 0 seconds (no angle change) to 22 seconds depending on slew magnitude. The means for each satellite, any pair of command choices which occur within slew time + 2 seconds are mutually exclusive. For each satellite $s_i$, mutual exclusion constraints (1) are created to enforce**:**

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

$$x_{i,c_1,t_1} + x_{i,c_2,t_2} \le 1 \qquad (3)$$

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

- 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 GP each *g* to commands *c* which cover it)

$$y_{g,i,c,t} \le x_{i,c,t} \qquad (4)$$
$i \le N, \quad \forall\, g \in G_{i,c,t}, \quad \forall t \in T_i, \quad \forall c \in C_{i,t}$

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

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

$$\sum_{i,c,t} y_{g,i,c,t} \le d_{max} \qquad \forall_g \in G_{i,c,t} \qquad (5)$$

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

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

**Objective:**

maximize $\sum_{\forall g \in G} r_{g,c,t}\ y_{g,i,c,t}$ (6)

- 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 $r_{g,c,t}$ in (6) is defined by equation (1). Notice the similarity between the MILP objective (6) and the DCP objective (2) and the GP Reward (1). This shared metric, along with identical input data, is the basis for our apples-to-apples comparison between the two methods.

The solver terminates when optimality is proven or when a specified time limit has been reached without proving optimality. Optimality is proven when the gap between the "primal" objective (6) and the MILP's "dual" objective reaches zero (within some tolerance). Equation (6) is the primal objective. Since this is a maximization problem, the "dual" objective is a minimization problem. It has been proven that the minimum value of the dual is equivalent to the optimal maximal value of the primal. MILP solvers exploit this property and flip back and forth between trying to maximize the primal and minimize the dual until no gap remains, at which time we know the primal objective is optimal.

## Evaluation

We compare the DCP and MILP methods using four test cases. It is important to note that there are far fewer rainy

| Case | # sats | Horizon (secs) | # TP | TP/ Hrz | #cmd/ TP | #GP/ TP (change to # GP) | Obs GP | # vars | # con-straints | time to best sol | time to prove opt | makespan (# cmds) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 21,600 | 1,705 | 0.08 | 12.55 | 6.2 | | | | | | |
| MILP | | | | | | | 1,032 | 26,959 | 914 K | 156 s | 156 s | |
| DCP | | | | | | | 948 | 1,705 | | 7 s | | |
| 2 | 1 | 1,000 | 900 | 0.93 | 70.98 | 64.0 | | | | | | |
| MILP | | | | | | | 620 | 89,118 | 14.4 M | 5 h | 16 h | |
| DCP | | | | | | | 666 | 900 | | 5 s | | |
| 3 | 1 | 1,800 | 1,468 | 0.83 | 55.83 | 53.9 | | | | | | |
| MILP | | | | | | | 1,212 | 122,675 | 16.8 M | 13 h | 38 h | |
| DCP | | | | | | | 1,181 | 1,468 | | 8 s | | |
| 4 | 1 | 21,600 | 7,463 | 0.35 | 53.85 | 47.2 | | | | | | |
| MILP | | | | | | | 4,010 | 244,207 | 20.8 M | 10.7 h | 10.8 h | 1,468 |
| DCP | | | | | | | 3,113 | 7,464 | | 1.5 m | | |
| 5 | 3 | 7,200 | 7,527 | 1.0 | 52.41 | | | | | | | |
| MILP | | | | | | | | 244,363 | 20 M | | | |
| DCP | | | | | | | 3,140 | 7,527 | | 2.5 m | | 1,636 |
| 6 | 3 | 2100 | | | | | | | | | | |
| MILP | | | | | | | | | | | | |
| DCP | | | | | | | | | | | | |

**Table 1: Comparison of scenario search space complexity (branching factor)**

GP than GP where it hasn't rained recently, so case 1 with rainy GP has far smaller search space.

Case 1: 3 satellites, Rainy GP only, horizon = 6 hours

Case 2: 1 satellite, no rain, 1000 ticks (16.67 minutes)

Case 3: 1 satellite, no rain, for 1800 ticks (30 minutes)

Case 4: 1 satellite, no rain. horizon = 6 hours, top 15 % of GP with largest model error. Select top 15 % of GP (sorted in decreasing model error). The 15 percent was selected empirically such that scenario produced around 20 million constraints (for acceptable solver times < 50 hours).

Case 5: 3 satellites, horizon = 2 hrs, top 15 % GP. Select top 15 % of GP (sorted in decreasing model error).

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

Table 1 shows details about the search complexity and solver performance for each scenario. The table columns are as follows: Case ID, # of satellites in the constellation, plan horizon followed by the number of TP in the horizon. The column TP/Horz = # TP/ Horizon. This represents the "TP density" which is a rough measure of problem complexity (particularly for MILP). The next column, #cmd/TP is the average number of command choices at each TP (the average of all $|C_{i,t}|$). #GP/TP = average # of GP visible per TP. Obs GP is the # of GP observed by the commands in the plan. # vars is the # of decision variables created. # cons is the # of constraints created. Constraints are pre-enumerated only for the MILP solver. Time to best solution is the solver time required to find the best solution (which is optimal for MILP, but suboptimal for CSP).

Time to prove opt is the time it takes the MILP solver to prove that the best solution it has found is actually optimal.

Note how much longer it takes to prove optimality compared to the time it takes to find an optimal solution for cases 2, 3, and 4. The last column, makespan, is the # of commands in the plan.

The difference in complexity between rainy and non-rainy is illustrated comparing case 1 (rainy case) with case 3 (non-rainy). Case 1 includes 3 satellites and a 6 hour planning horizon, with a total of 1,705 TP, while Case 3 includes 1 satellite and a 30 minute plan horizon for a total of 1,468 TP. This difference is caused by the larger number of observation opportunities for non-rainy GP.

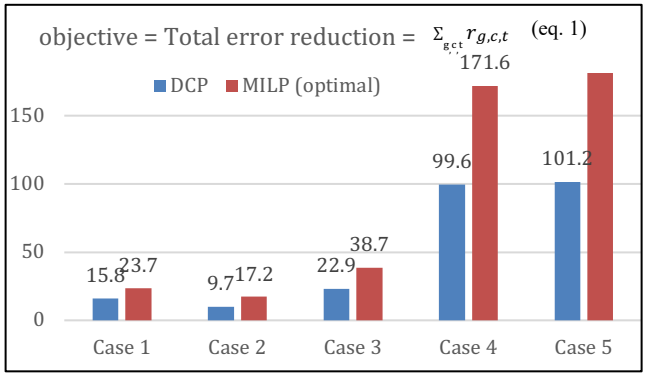


**Figure 6:** Comparison of MILP and DCP objective scores

Figure 6 shows a summary comparison of the objective values achieved by both planners on each scenario. 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. MILP performance is dominated by the

| Algorithm | Pro | Con |
|---|---|---|
| DCP | • Search control over order of node, var, and val choices Flexible (can model any constraint) or problem-solving procedure<br>• Leverages domain heuristics. Easy to swap between different heuristics.<br>• Fast, amenable to running onboard<br>• Explainable<br>• Constraint code is executed on demand rather than predefining<br>• Symbolic reasoning (decision variable domains are symbols like 'L.34' for more flexibility vs. #s) | • Suboptimal solutions<br>• Local minima<br>• Path dependency |
| MILP | • Provably optimal solutions<br>• Slow<br>• Relies on 3rd party solver (benefit of robust heavily tested tool) | • Limited modeling flexibility<br>• No domain heuristics<br>• Limited explainability<br>• Relies on 3rd party solver (disadvantage of it being a black box with limited insight or control of search process details).<br>• Requires predefining all constraints in advance (10's of millions of constraints must be created but most are not required for any specific solution).<br>• Requires all variable domains be quantitative. |

**Table 2:** Summary of trades between DCP and MILP

exponential scaling in the number of constraints for some scenarios. The number of mutex constraints are dominated by deconfliction within 25-second windows. The maximum slew time is 25 seconds, so mutex constraints are created only to deconflict command choices which occur within 25 seconds of each other. The number of mutex constraints is a function of how many TP and choices occur within any given 25 tick time window.

**Trade Analysis.** Table 2 shows a summary of 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 and MILP fall prey to different scaling challenges. DCP is prone to local minima but far smaller search space (fewer vars and constraints). MILP optimality ensures no local minima but scaling creates so many more vars and constraints that solver time becomes impractical (> 1 day) for desired plan horizons of 24 hours (or even 6 hours).

The cases in table 1 illustrate the boundary of scenarios which we can solve with MILP. For example if we increase the horizons for any of the non-rain cases, then the model file becomes to large to write out, or it takes longer than 48 hours to solve.

## Future Work

Several constraints have not yet been developed. These include:

- "Multi-observations", where we may take multiple [illegible]s of the same GP within a 2 hour window which are "merged" together as a "single" observation with reduced measurement error.
- Energy model to track energy consumed by instruments and slewing and produced by solar panels, to ensure no satellite dips below a minimum energy.
- Contiguous identical commands (contiguous GP observations > 2 seconds).
- We also plan to integrate with new sensors and with multiple, independently developed planners for: downlinking data, intersatellite communications, and UAV flight planning.

## Conclusion

We have described an important climate change monitoring application which requires online planning capabilities which are beyond the state of art. We have presented alternative methods to solve the problem, Dynamic Constraint Processing (DCP) and Mixed Integer Linear Programming (MILP), and described the many 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 application's 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 differences which led to improvements on both sides.

## References

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.

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

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