ORIE 6140 & 4140

Official name: Modeling operational systems

Goal: Optimization for fun and profit

Lecture 21 – Satellite Image Scheduling

Wednesday, November 6 2019

Brenda Dietrich, Ph.D

Writing code to build models

- Several of you had difficulty with the computational assignment
- Some issue were related to reading data from the files
- But most were due to either modeling errors, or mistakes in translating the math into either AMPL or Python code.
- The data set you were given was small, relative to real world problems
- But it was intentionally large enough to make hand entry of the data painful.
- And the model was complex enough that it was impossible to recognize a “correct” solution.
- So, how do you know if you’ve correctly implemented a model?

Test cases

- Unit test means testing a single feature
- System test means testing how all the features work together
- Unit test cases should be small enough that you can recognize correct and incorrect answers, but large enough that to test the feature you are trying to test.
- Example: To test the fact that the model handles late delivery of demand,
  - How many time periods do you need?
  - How many resources do you need?
  - How many products do you need?
  - Do you want the test problem to be able to meet all the demand on time?
  - Do you want it to be able to meet all the demand, in total?
  - What form should your objective function coefficients take so that you can tell whether demand is being met late?
- Can you design a single simple test case, or should you have multiple similar test cases (for example, with different RHS or OBJ coefficients)?

Minimal case to test the model for handling late demand as desired

One material resource is enough.

3 time periods to test allowing shipment to be one period late, but not 2 periods late

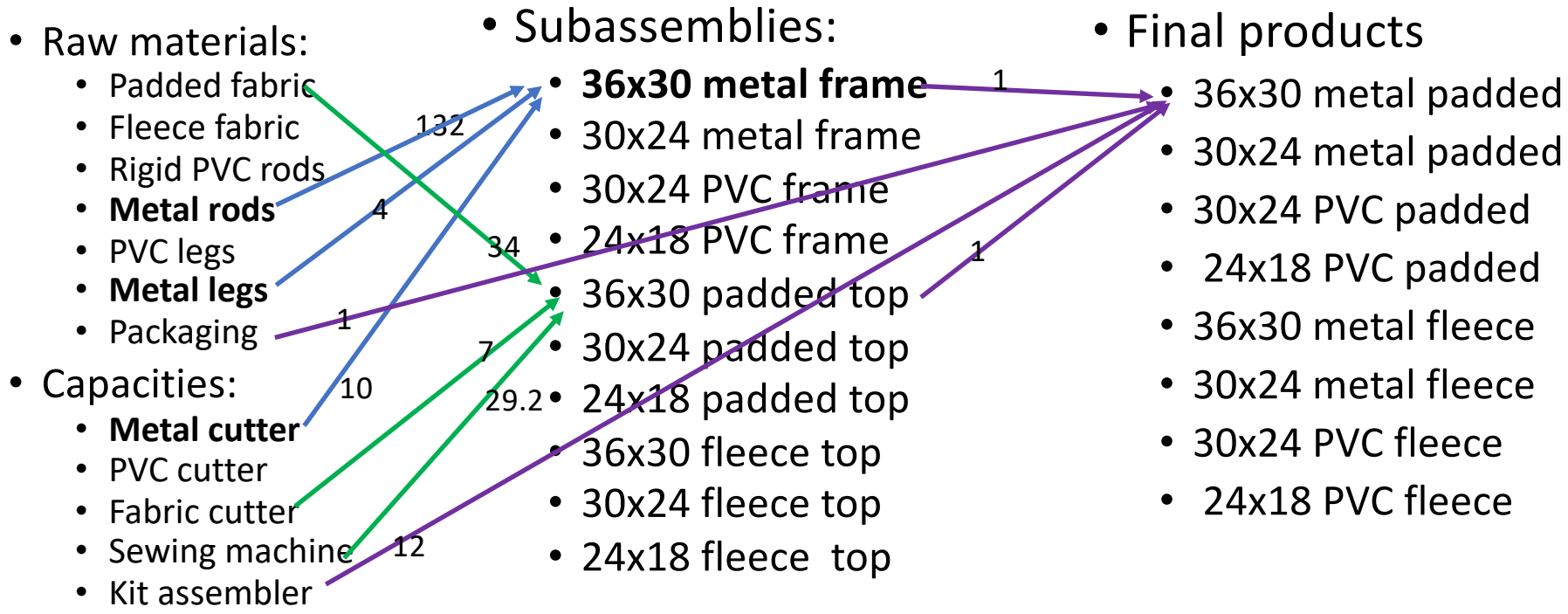
	demand	Supply	Unit rev (on time)	Unit rev (1 period late)	Shipment for current period	Shipment for 1 period earlier	OBJ value
Period 1	10	5	10	7	5		
Period 2	0	7	8	6	0	5	
Period 3	4	1	5	4	3		
							100

May want to also consider different supply scenarios that are more constrained

Use case for balance constraints

- Can test most function with a single level BOM structure
- Want both capacity and material
- Want multiple periods (3 is enough)
- Want demand in both periods
- Goal is to test that the material is being “carried over” to the next period, so can have all of the supply of the material in the first period.
- Goal is to test that capacity is not carried over, so should have capacity gating production in one of the first 2 periods.

Example: 36x30 metal padded dog bed



The subassemblies can be made in advance and stocked until needed for increased efficiency in use of capacity
There **is independent demand** for subassemblies and parts (e.g., replacement parts)

Example

36 x 30 frame	Metal Rods	Metal Legs	Metal Cutter
usage	132	4	10

	Supply Metal Rods	Supply Metal Legs	Supply Metal Cutter	Demand 36x 30 frame	Production	Shipment For current period (rev 10)	Shipment for previous period (rev 5)	Scrap Metal cutter	OBJ	
Period 1	1320	**20**	100	10	5	5		50		
Period 2	1320	60	**70**	10	7	7		0		
Period 3	1320	40	140	10	13	10	3	10		

Could probably get away with just 2 parts and 2 periods, but I wanted to make sure it didn't try to ship period 1 demand in period 3

Final Computational HW

1. Construct a minimal test case for a multilevel BOM structure

2. Run your model on the two testcases in this lecture and on your test case for a multi-level BOM structure.

3. Validate the solution to your model on the large data set by

(1) Checking that it is feasible (you can build a spreadsheet to do this, or you can reduce the resource availability to the exact amounts used in each period and re-running. You should get the same solution. If you don't figure out why.

(2) Adjusting revenue values and checking that the solution moves in the right direction.

(3) Adjusting supply levels and checking that the solution moves in the right direction

4. If you are still having problems getting the model to work, collapse all time into a single period, aggregate supply and demand, and see if you can solve that problem.

Then bring in the time index.

Background – low earth orbit satellites

A **low Earth orbit** (**LEO**) is an Earth-centered orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of the earth) or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.

An object in orbit is, by definition, in free fall, since there is no force holding it up.

Objects in LEO orbit Earth between the denser part of the atmosphere and the inner Van Allen radiation belt.

A low Earth orbit requires the lowest amount of energy for satellite placement. It provides high bandwidth and low communication latency. Satellites and space stations in LEO are more accessible for crew and servicing.

Since it requires less energy to place a satellite into a LEO, and a satellite there needs less powerful amplifiers for successful transmission, LEO is used for many communication applications, such as the Iridium phone system.

Satellites in LEO have a small momentary field of view , only able to observe and communicate with a fraction of the Earth at a time, meaning a network of satellites is required to in order to provide continuous coverage

A majority of artificial satellites are placed in LEO, making one complete revolution around the Earth in about 90 minutes.

Examples

Earth observation satellites and spy satellites use LEO as they are able to see the surface of the Earth clearly by being close to it. They are also able to traverse the surface of the Earth.

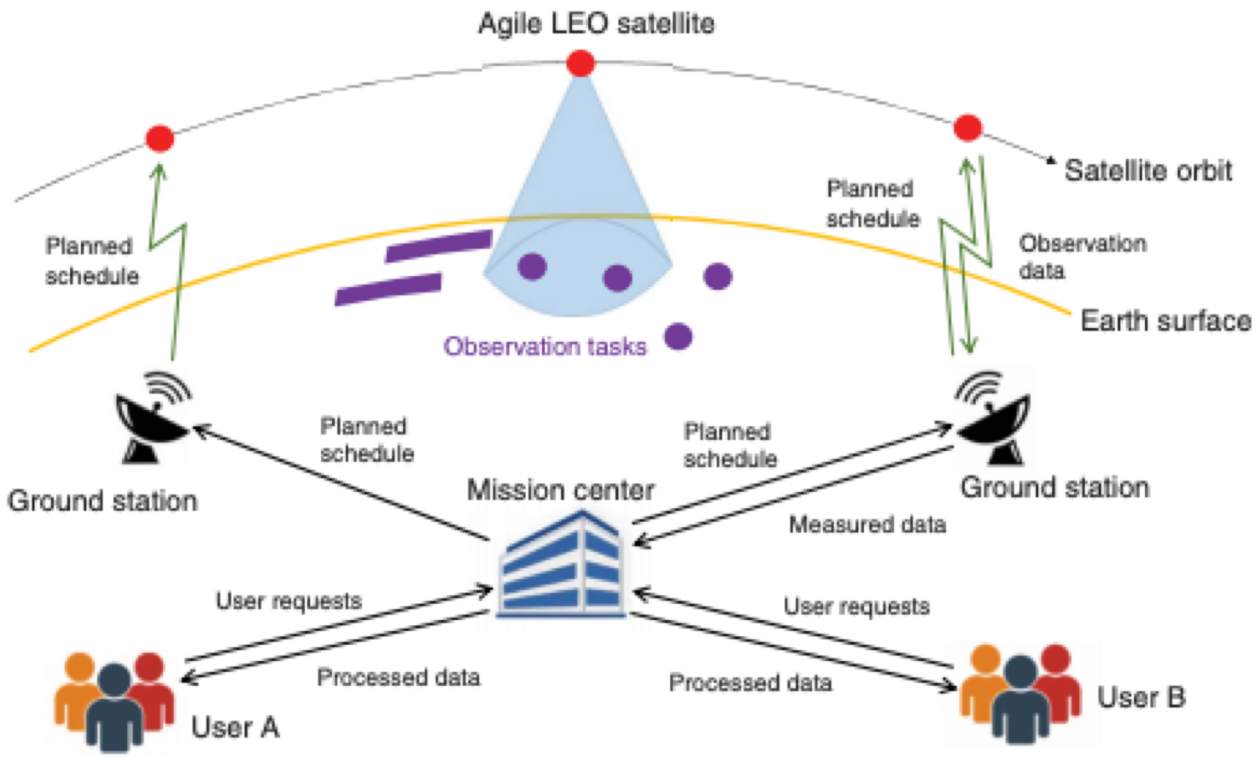
The International Space Station is in a LEO about 400 km (250 mi) to 420 km (260 mi) above Earth's surface, and needs reboosting a few times a year due to orbital decay.

Lower orbits also aid remote sensing satellites because of the added detail that can be gained. Remote sensing satellites can also take advantage of sun-synchronous LEO orbits at an altitude of about 800 km (500 mi) and near polar inclination.

The Hubble Space Telescope orbits at about 540 km (340 mi) above Earth.

The Chinese Tiangong-2 station orbits at about 370 km (230 mi).

The Joint Space Operations Center, part of US Strategic Command currently tracks more than 8,500 objects larger than 10 cm in LEO. However, a limited Arecibo Observatory (radio telescope in Puerto Rico) study suggested there could be approximately one million objects larger than 2 millimeters, which are too small to be visible from Earth-based observatories.



Communication with ground stations is not continuous.
Data can be received, and new instructions transmitted, only when in communication with a ground station.
Two types of tasks to be scheduled: observation tasks (requested by users) and download tasks.

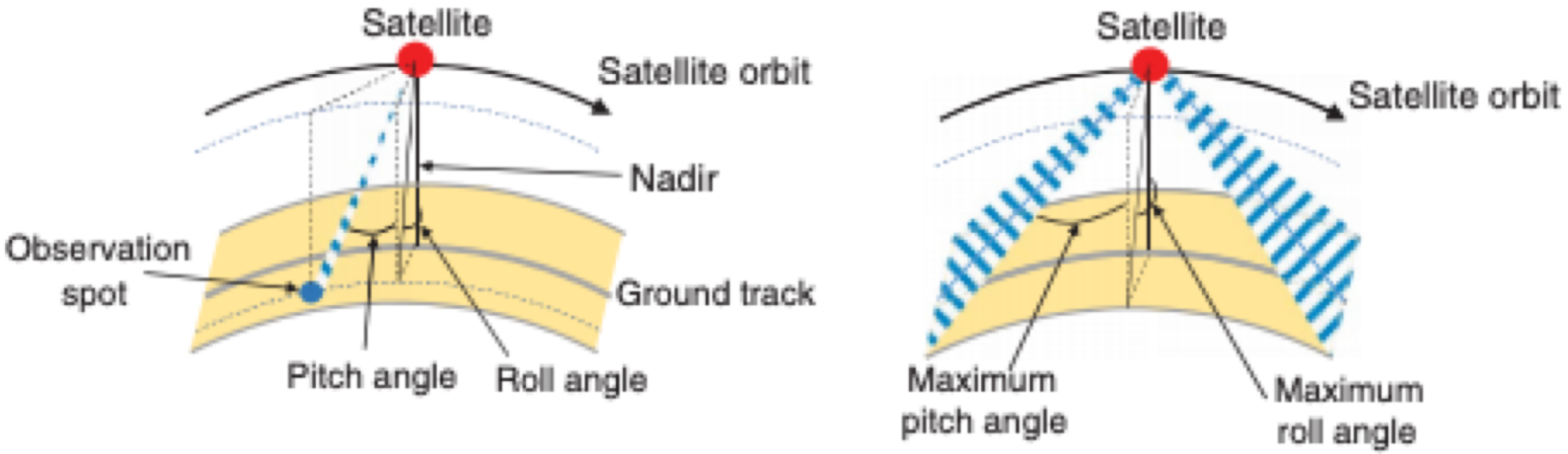


Fig. 2 Illustration of relevant angles (left) and maximum roll and pitch (right).

Two angles can be controlled: Pitch and Roll.
These determine the field of view at any position in orbit.
Therefore, the visible earth locations can be computed for any instant in time.

Adjusting pitch and roll takes time.

The ground station also requires time to align its antenna with the satellite it is communicating with.

Some additional complexities

The satellite has limited onboard data storage.

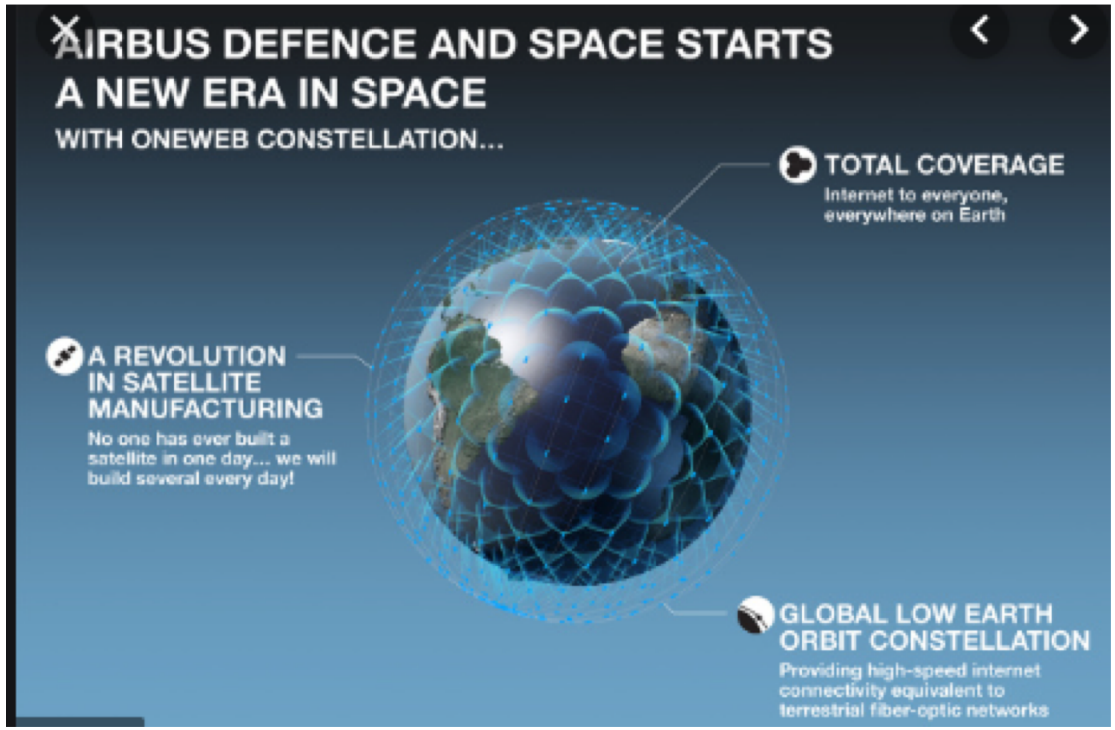
The amount of image data it can usefully collect between downloads is limited.

The satellite is powered by a finite capacity battery, which can recharge only when the satellites solar panel is facing the sun.

The “work” that can be done between recharges is limited by the battery capacity.

“Work” includes angle adjustments and image capture.

Multiple satellites orbiting in formation



Design Decisions:

- Satellite specifications
- Position of orbits
- Position of ground stations
- Launch schedule

Operational decisions:

- Ground station linkage schedule
- Image capture schedule

Mission scheduling – simple case

Set of jobs $J = \{1,2, \dots, n\}$ (these are locations to be imaged)

d_j is the duration of job j.

r_j is the revenue for completing job j.

Set of satellites $S = \{1,2, \dots, k\}$

$W_{ji} = [W_{jib}, W_{jie}]$ is the window of time in which satellite i is in a position to perform job j.

Output:

A set of timelines $T_1, T_2, \dots, T_k$

Where T_i, the timeline for satellite i , has at most one job assigned at each instant in time.

Main decision variables:

$$x_{ij} = \begin{cases} 1 \; if \; job \; j \; is \; assigned \; to \; satellite \; i \\ 0 \; otherwise \end{cases}$$

We do not require that all jobs be assigned, but a job can be assigned to at most one satellite:

$$\sum_{i=1}^{k} x_{ij} \leq 1, \qquad for \; j = 1,2,..,n$$

The objective function is easy: Max $\sum_{j=1}^{n} r_j (\sum_{i=1}^{k} x_{ij})$

The hard part is making sure that the jobs assigned to the same satellite can be ordered in a way that does not introduce overlaps in imaging time.

Getting the order right

If jobs j and l are both assigned to satellite i, then either job j is completed before job l starts, or either job l is completed before job j starts.

Let $y_{ijl} = \begin{cases} 1 \; if \; jobs \; j \; and \; l \; are \; both \\ \quad assigned \; to \; i, and \; j \; is \; processed \; before \; l \\ 0 \; otherwise \end{cases}$

This is a common trick to use when order matters

$$x_{ij} + x_{il} - 1 \leq y_{ijl} + y_{ilj} \leq \frac{1}{2}(x_{ij} + x_{il}), for \; all \; i, j, l$$

$$y_{ijl}(s_{ij} + d_j) \leq y_{ijl} s_{il}, for \; all \; i, j, l$$

We also need the constraints $s_{ij} \geq W_{jib} \; x_{ij}, for \; all \; i, j$

$$s_{ij} + d_j \leq W_{jie} x_{ij}, for \; all \; i, j$$

Number of binary variables is kn^2.

The complicating factors

Adjusting angles

The time required depends on

(1) Final position at conclusion of previous job
(2) Location of satellite when job begins

The relative order variables y_{ijl} say that j is before l.

We need to know the sequential pairs

Let $z_{ijl} = \begin{cases} 1 \; if \; job \; j \; is \; processed \; immediately \; before \; l \; on \; i \\ 0 \; otherwise \end{cases}$

We need a dummy first (0)and last job (q) on each satellite.

It helps to keep track of order: $O_{ij} = order \; number \; for \; job \; j \; on \; satellite \; i$

$$\sum_l z_{ijl} = x_{ij} \quad (one \; successor) \; for \; j \neq q$$

$$\sum_j z_{ijl} = x_{il} \quad (one \; predecessor) \; for \; l \neq 0$$

To capture set up time approximately

$$(s_{ij} + d_j + t_{ijl}) - (1 - z_{ijl})M \leq s_{il}, \quad for \; all \; i, j, l$$

Where t_{ijl} is an approximation of the time to adjust the angles.

To capture this time exactly we would need to also track the angles as decision variables.
Feel free to give it a try…..

Trivial to update column generation to include the approximated setup time.

Mission scheduling – Alternative approach

It is easy to filter the date to see which jobs can possibly be done on each satellite.

It is easy to generate feasible timelines for each satellite.

- Having a satellite do nothing at all is feasible, but generates 0 revenue
- Having it do only one job is also feasible.

Given an ordered list of jobs, it is easy to check whether they can be done, in that order, on a satellite i.

- Let $j_1, j_2, \ldots, j_m$ be the ordered list.
- $s_{j_1 i} = W_{j_1 ib}$, if $s_{j_1 i} + d_{j_1} > W_{j_1 ie}$ $then\ the\ sequence\ is\ infeasible\ .STOP$
- $s_{j_2 i} = Max\{W_{j_2 ib}, s_{j_1 i} + d_{j_1}\}, .If\ s_{j_2 i} + d_{j_2} > W_{j_2 ie}\ then\ the\ sequence\ is\ infeasible$. STOP
- ….
- $s_{j_2 m i} = Max\{W_{j_m ib}, s_{j_{m-1} i} + d_{j_{m-1}}\},\ If\ s_{j_m i} + d_{j_m} > W_{j_m ie}\ then\ the\ sequence\ is\ infeasible$.STOP
- Otherwise, the sequence is feasible with the computed start times

It is also easy to calculate the revenue of any individual feasible timeline.

This is a case where column generation will obviously work, and will probably work better than the naïve formulation.

Note that the underlying set of jobs may be feasible for a satellite, even if the specified order is not.

A traveling salesman problem with time windows can be used to find a feasible sequence of a set, if such a sequence exists. (UPS example)

The column-based formulation

Let $\mathcal{C}_i$ be a set of ordered lists of jobs, such that for each $C \in \mathcal{C}_i$, the jobs in C can be done in that order by satellite i.

Let $r(C) = \sum_{j \in C} r_j$ be the reward for the jobs in C

Ignoring the download and energy considerations, we can formulate the reward maximization problem

$$Max \sum_i \sum_{C \in \mathcal{C}_i} u_{iC}\, r(C)$$

Subject to

$$\sum_i \sum_{C : j \in C \in \mathcal{C}_i} u_{iC} \leq 1, \qquad \forall j$$

$$\sum_{C \in \mathcal{C}_i} u_{iC} \leq 1, \qquad \forall\, i$$

$$u_{iC} \in \{0,1\}, \ \forall i, \forall\, C \in \mathcal{C}_i$$

Note that the pricing problem will favor columns that include uncovered jobs. The subproblem involves generating feasible columns that include these jobs.

Multiple time windows for each job

The direct formulation requires main variable x_{ij}^{k} to indicate that job j is done by satellite i in the time window $W_{ji} = \left[W_{jib}^{k}, W_{jie}^{k}\right]$

This superscript is needed on the detail variables (y, z) too.

The feasibility check for a sequence is easily modified to deal with multiple windows.

There is a constraint logic programming approach that appears to work well on the TSP with multiple time windows. So column generation for this generalized problem is possible.

Complicating Factors – Energy limits

We know how much energy is required for each sequence dependent setup and for each observation and data download.

We know when the satellite is able to recharge through solar panels

We know the battery capacity and the initial charge

It's straightforward to calculate the remaining charge at any point in time, under the assumption that the battery is charging whenever possible.

If the remaining charge goes negative, the sequence is infeasible.

If it remains positive throughout, it is feasible.

Complicating factor –ground station contact

We know when contact with each ground station is possible.

The authors of this paper first solve the ground contact problem, basically maximizing a weighted combination of the total download time and the amount of contact time for the satellite with the least contact, subject to the possible times, the constraint that a ground station can only connect with one satellite at a time, and that some time is required to point the ground station antenna at the satellite.

Since they give no actual problem data, it is not clear whether all of the time allocated for download is actually needed.

Also, they should probably also be minimizing the maximum time between downloads, since the real goal is to prevent the onboard data storage limit from being exceeded, and it is not obvious that either of their two objectives address this.

Assuming the ground station contact times are fixed in advance, the feasibility check can easily be modified to account for that time not being available for other jobs.

Since the ground contact time determines the amount of data that can be downloaded, we can also check any proposed sequence for data feasibility.

Data download as a task

If we view data download as a task, we have time windows and angle settings for it. We don't have a duration, but we have an upper bound on the duration which is the time required to download the full capacity.

Further, if data download jobs are included in the sequence, we can compute how much data can be downloaded (min of stored data and amount that can be downloaded in the available portion of the relevant window), and as with energy, determine whether the sequence results in a data overload.

This will give us valid columns, but eliminates the possibility of stopping the download to begin another job.

Without a sense of the actual values (amount of data per job, data capacity, download rate) we don't know whether this is a significant issue.

Augmenting columns

We can augment our notion of columns, to include for the download tasks the duration of time allocated for the download.

So a column becomes a sequence of jobs, with the download jobs including a start time and an end time.

Note, we implicitly have the start and end time for all the jobs.

We have the complicating constraint that a ground station can be in contact with only one satellite at a time.

And we have these constraints for every ground station at every point in time.

(Or at least for every point in time at which ground contact is begun)

Rather than explicitly including the constraints, I would solve the problem without them, look for conflicts, see if there is an adjustment of the columns that eliminates the conflict (e.g., shortening the duration of the download on one or both columns), and adding a constraint preventing selection of more than one of the pair of columns only if the conflict can not be resolved.