IAC-15-D1.4B.2

NAVIGATING THE DEPLOYMENT AND DOWNLINK TRADESPACE FOR EARTH IMAGING CONSTELLATIONS

Sreeja Nag

NASA Goddard Space Flight Center, MD; Bay Area Environmental Research Institute, CA USA
sreeja.nag@nasa.gov

Steven P. Hughes[1], Jacqueline J. LeMoigne[2]
NASA Goddard Space Flight Center, MD USA
[1]steven.p.hughes@nasa.gov [2]jacqueline.j.lemoigne-stewart@nasa.gov

Abstract

Distributed Spacecraft Missions (DSMs) are gaining momentum in their application to Earth Observation (EO) missions owing to their unique ability to increase observation sampling in spatial, spectral, angular and temporal dimensions simultaneously. DSM design includes a much larger number of variables than its monolithic counterpart, therefore, Model-Based Systems Engineering (MBSE) has been often used for preliminary mission concept designs, to understand the trade-offs and interdependencies among the variables. MBSE models are complex because the various objectives a DSM is expected to achieve are almost always conflicting, non-linear and rarely analytical. NASA Goddard Space Flight Center (GSFC) is developing a pre-Phase A tool called Tradespace Analysis Tool for Constellations (TAT-C) to initiate constellation mission design. The tool will allow users to explore the tradespace between various performance, cost and risk metrics (as a function of their science mission) and select Pareto optimal architectures that meet their requirements. This paper will describe the different types of constellations that TAT-C's Tradespace Search Iterator is capable of enumerating (homogeneous Walker, heterogeneous Walker, precessing type, ad-hoc) and their impact on key performance metrics such as revisit statistics, time to global access and coverage. We will also discuss the ability to simulate phased deployment of the given constellations, as a function of launch availabilities and/or vehicle capability, and show the impact on performance. All performance metrics are calculated by the Data Reduction and Metric Computation module within TAT-C, which issues specific requests and processes results from the Orbit and Coverage module. Our TSI is also capable of generating tradespaces for downlinking imaging data from the constellation, based on permutations of available ground station networks - known (default) or customized (by the user). We will show the impact of changing ground station options for any given constellation, on data latency and required communication bandwidth, which in turn determines the responsiveness of the space system.

Acronyms			
CR	Cost and Risk Module	SSO	Sun Synchronous Orbit
DSM	Distributed Space Mission	STK	Systems Tool Kit
ED	Executive Driver	TAT-C	Tradespace Analysis Tool for Constellations
EO	Earth Observation	TRL	Technology Readiness Level
FOV	Field of View	TSI	Tradespace Search Iterator
GMAT	General Missions Analysis Tool	TSR	Tradespace Search Request
GS	Ground Station		
GUI	Graphical User Interface		
ISS	International Space Station		
JSON	JavaScript Object Notation		
LV	Launch Vehicle		
MA	Mean Anomaly		
NEN	NASA Earth Network		
OC	Orbit and Coverage Module		
POI	Point of Interest		
RAAN	Right ascension of the ascending node		
RM	(Data) Reduction and Metrics (Computation) Module		

1. Introduction

Distributed Space Missions (DSMs) are becoming popular in government (e.g. NASA's Earth Science Technology Office 2030 Science Vision envisions 'distributed observations' and formation flight), academia (e.g. Europe's QB50 mission) and industry (e.g. Planet Labs, Google Terra Bella) to address the need for repeated, global measurements for Earth observations, monitoring and quick response. NASA's decadal surveys or their mid-term assessments have called for the consideration of

DSMs in areas of Earth science, astrophysics, heliophysics and planetary science.

Designing space systems is not only technically challenging but also involves making hundreds of decisions early in the design cycle for allocating limited resources across the system and optimizing performance and cost. Earth Observation or EO performance can be simplistically represented by spatial resolution, spatial range (swath, coverage), spectral resolution (wavelength bandwidth), spectral range (spectrum covered), angular resolution (number of view and solar illumination angles for the same image), angular range (spread of those angles), temporal range (mission lifetime), temporal resolution (repeat or revisit time), radiometric range (number of bits) and radiometric resolution (bits, signal to noise ratio). DSMs are gaining popularity in Earth Science[1] because they can make new measurements by enabling simultaneous observation sampling in spatial, spectral, temporal and angular dimensions and multiple satellites are now cost manageable due to smaller spacecraft and cheaper access to launch. Small satellites ~ 100 kg (and to some degree, Cubesats[2]) are now capable of high resolution imaging, high bandwidth communication and accurate attitude control[3].

DSMs have all the trades associated with monolithic systems *and* more associated with the network. Extra design variables include but are not restricted to the number of satellites and their individual masses, their orbits and inter-satellite spacing, existence and nature of inter-satellite communication and downlink schedules. These variables directly impact performance and cost. Performance variables, as defined, can be mutually conflicting across the spatial, spectral, temporal, angular and radiometric dimensions and within each dimension. For example, more launches allow wide spread in the constellation planes but more launch vehicles cost more, and are very susceptible to launch delays causing long waits to full science performance. Larger field of regard for an imaging sensor covers the globe faster, but at the cost of lower spatial resolution. Increasing ground station spread or number of satellites globally improves data latency both at greater cost. Such conflicting design variables are in plenty and need to be permuted to display architectures that show such trade-offs.

Constellations have so far been the most common type of DSM and NASA's Science Mission Directorate has recently flown and funded two constellation missions, CYGNSS and TROPICS respectively. NASA GSFC is leading the development of a Tradespace Analysis Tool for Constellations or TAT-C[4], which will allow scientists to explore constellation mission architectures, that minimize cost and maximize performance for pre-defined science goals, and will be aided by knowledge databases and machine learning. In a prior publication in 2016[5], we described the executive driver of TAT-C, which ingests user inputs, enumerates and searches the all possible architectures, calls all the other modules and arranges the results of each architecture neatly into a file tree. We further explained the tradespace search process run by the Driver and how it can be streamlined by combining physical rules, as well as well-designed orbit and coverage computations, thus yielding significant speed-ups. The orbit, coverage, data reduction and metric computation modules were also focussed upon. Two use cases were shown as representative examples of the utility of generated trades, and results are preliminarily validated against AGI's Systems Tool Kit (STK).

This paper will describe the inputs to the tradespace search organized as classes, improvements to the architecture enumeration process since 2016 and list the performance metrics generated after the aforementioned modules have completed their run. We will describe the enumeration of different types of constellations, simulations for their deployment via launch vehicle (LV) options and downlink via several ground station (GS) network options. Finally, we will show the impact that architectures generated by permuting these improved options have on the described performance metrics. To our knowledge, *the new contributions of this work* are: (1) detailed description of a tradespace search and evaluation tool for Earth imaging constellations with more design variables and performance outputs than published in academic literature before; (2) inclusion and analysis of new, unpublished types of constellations; (3) structured enumeration of existing types of constellations for analysis; (4) inclusion of a customizable imaging sensor capable of projecting any shape and computing Earth coverage.

2. Functionality of Relevant Modules

TAT-C has several modules, of which only a few will be addressed in this paper – those colored in pink and green in Figure 1. Once the user has entered his inputs through the TAT-C GUI, the Executive Driver also called the ED, picks up the key-value pairs of all the user inputs by means of a JSON file – a lightweight data-interchange format. The values in the JSON file may be numbers, tokens, ranges or paths to text files within the user's computer. After a
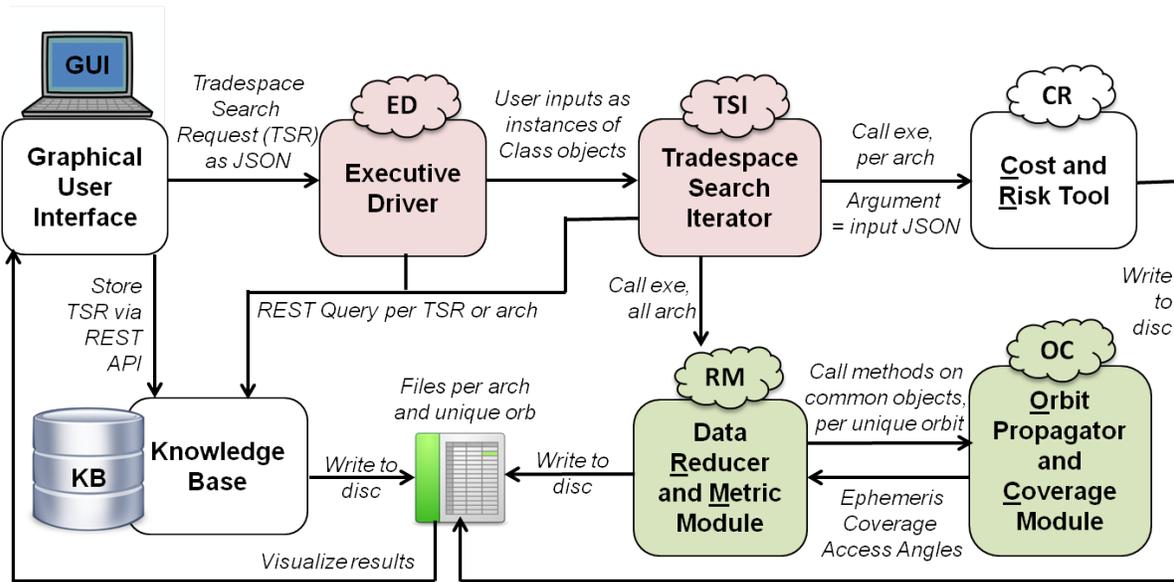
Figure 1: Information flow through the ED/TSI, RM and OC modules of TAT-C, marked in color, as they interface with the other modules. The modules are programmed in Python (pink) and C++ (green). Currently, the KB and TAT-C have different GUIs, but can be operated on the same user system and have access to the same user disc or folders.

sanity check on all the inputs, the ED initializes Python classes for each input category, as will be described in Section 2.1, and called the Tradespace Search Iterator or TSI module. ED permutes different combination of design variable values to generate a full factorial set of architectures. Specifically, the design variables considered so far are Constellation types and initial Keplerian elements for each satellite (discussed in Section 3), number of satellites, field of view dimensions of the imaging sensor (discussed in Section 2.4), altitude and inclination spread within user bounds, initial eccentricity and perigee spread per satellite, frequency of constellation maintenance, ground station (GS) options and combinations, communication bands used for data downlink and launch vehicle options and schedules (all discussed in Section 4).

At any level of variable permutation, the TSI automatically downselects acceptable bounds for the next level in the design tradespace. After each architecture is generated by the TSI, the Cost and Risk module, also called CR, is called to assess it (see [4] and [14] for more details). After all the architectures have been generated (Section 2.1), the reduction and metrics (RM) module, is called to evaluate them in terms of science performance. RM is responsible for in-memory calls to the orbit and coverage module, henceforth called OC, as required. RM is called after all architectures are known, unlike CR, because several architectures are expected to share satellites with exactly the same specifications and orbits. RM is optimized such that such common satellites are propagated and coverage computed just once, to improve computational efficiency by avoiding redundant processing. The model increases the risk of data loss in case the TAT-C simulation crashes midway and we are working to change our architecture to address that risk, without compromising on computational speed. RM processes all satellites and architectures (Section 0) and writes the results as csv files on the user's disc, which can then be visualized by the GUI.

2.1. Executive Driver

The role of the ED is to conduct the trade-space search in coordination with all the TAT-C modules, starting with the TSI, using the Tradespace Search Request or TSR. Figure 2 shows an example of a folder that the user is required to provide the location of, containing details of his/her TSR. 'TradespaceSearchRequest. json' is the file created by the GUI with the user's inputs (see Figure 3 in Reference[5]). 'Landsat_landImages.txt', 'InstrumentSpecifications. txt' and 'ObservatorySpecifications. txt' are the text files that the JSON file references for customized values, as provided by the user. Depending on whether the user inputs a range, an exact value, selects among available options and/or provides a text file path with specifications in 'TradespaceSearchRequest.json', the

ED checks the validity of user inputs, initializes the relevant classes and passes the objects to the TSI. If there are inputs the user has not provided, the ED is expected to throw an exception or populate it with default values. Reference[5] contains results from example runs of the ED and TSI. Each class corresponding to user inputs, its members and their functions are described below in terms of how they contribute to the architectures generated by the TSI (Section 2.2). The ED currently has access to text files within its internal, user-editable library containing parameters for the NASA Earth Network (NEN) ground stations, Deep Space Network ground stations, TDRSS, atmospheric density profiles and commercial launch vehicles available in the market.

2.1.1. *Mission Concept*
This class allows the ED to organize user inputs related to the mission concept, where in all temporal requirements are expected in seconds. The user can define the organization managing the mission, and the ED passes this information to CR for cost considerations. Start epoch is in UTC time, mission duration is the total time horizon of the mission, ending in de-orbiting all spacecraft, and performance period are a set of ranges when the user would like the RM to compute outputs. The user can specify the area of interest as a file path to a text file with multiple rows - each of which specifies a unique point's latitude (in degrees), longitude (in degrees), altitude (in km), a range of latitudes or a range of latitudes and longitudes. If only latitudes are specified, ED will consider the full longitudinal spread and create defaults accordingly. The user's points or area of interest are passed on to the RM by the ED for coverage computations. Objects of interest are for missions that allow occultations between a satellite and another external body through the atmosphere (e.g. Sun) or via a reflection point on the Earth (e.g. GPS radio).

Ground Station Options within the ED organizes the user's ground related inputs. She/he can select some stored ground station networks such as the NEN – Government stations, NEN – Commercial stations, NEN – all, DSN, TDRSS - Tracking and Data Relay Satellite System; or provide a file path to a text file with multiple rows - each of which specifies a unique GS's latitude (in degrees), longitude (in degrees), altitude (km), logical 0 or 1 for if it is rented or not and string of communication bands that station supports. This information allows the TSI in Section 2.2 to create architectures containing only those ground stations that can support a given satellite's orbit and transponder. If the user selects one of the

stored networks, the ED will use a text file from its editable library, associated with that network, whose structure is the same as that of the customized text file that the user can point to.

The ED's launch preference variables organize the user's selections. She/he can choose between 'Primary' or 'Secondary' launch options or provide a file path to a text file with multiple rows - each of which specifies the values in the class variables shown below. They include the unique launch vehicle's name, mass – dry and propulsion (kg), payload volume (cub.m), mean time between launches (mbtl), reliability, cost (dollars) and other properties such as number of booster relights. The 'Primary' or 'Secondary' options will cause the ED to look for one of two text files within its editable library, which contains multiple rows of primary of secondary launch vehicles respectively. If the user selects those options, one of those two text files are used as launch options for generating launch-related architectures. The Launch Vehicle (LV) class contains methods to read the TSR and populate an object corresponding to any LV, compute the allowable spread for precession constellations when using that LV, required fuel for maintenance, maintenance frequency, allowable number of satellites per launch, etc.

The ED also registers a user selected propagation fidelity and whether the mission will use propulsive maintenance. Fidelity is a number between 0 to 4, where 0 corresponds to the J2 propagator and 4 corresponds to the full GMAT propagator[6] including a detailed atmospheric model. Currently, the OC module is capable of running propagations including J2 gravity harmonics and including atmospheric drag. If the user would like propulsive maintenance to be used, per the 'propulsion' variable, the ED automatically turns off the effect of drag.

2.1.2. *Orbit Specifications*
This class allows the ED to organize user inputs related to constellation orbits, including the ability to complement an existing spacecraft or constellation, using a new constellation. Exiting satellite options allow the user to provide a list of satellite specifications that he/she would like to complement. He/she can provide these options as a filepath to a text file, in which every row corresponds to a unique existing satellite's orbital specifications and instrument specifications: eccentricity, inclination, semi major axis, perigee, RAAN and mean anomaly, followed by all the observatory and instrument specifications (described in Sections 2.1.3 and 2.1.4).

The TSI will generate only those new orbits whose outputs match requirements, as calculated in compliment with the existing satellite options. Altitude and inclination range of interest allows the user to set bounds or multiple ranges for either. The user may also select among 'LEO', 'MEO' or 'GEO' for an altitude choice, in which case TAT-C assumes a requested altitude range of 300-1000 km,19000-24000 km or exactly 35786 km respectively. If the user selects any available special orbits (the current options we provide are sun synchronous orbits/SSO, frozen orbits or ISS orbits), the TSI will force all the satellites to follow those constraints. Number of new satellites allows the user to enter the lower and upper bounds of the satellites allowed in the new constellation.

Since the ED and TSI have been programmed to eventually include heterogeneous constellations, therefore the user will be able to specify the number of satellite types, and then for each type, enter the satellite and instrument/payload specifications. The orbits class contains methods to read the inputs from the TSR and to compute initial Keplerian elements for the full tradespace of allowable satellite orbits. These methods can be called by the TSI for different types of constellations and user orbital bounds, and results passed on to RM/OC for propagation and coverage computation.

2.1.3. *Observatory Specifications*

This class allows the ED to organize user inputs related to the physical characteristics of the satellite/s he/she wishes to analyse as a constellation. All instruments are assumed to be body-fixed, however, the user can input rotational movements for the satellite in the form of the maximum angle that the satellite can swing (along and cross track) and the scan rate. He/she can select one or more communication bands supported by the satellite transponder among S, X, Amateur Radio, Ka, Ku and Laser. Finally, there is the option of entering the number of instruments per satellite. For each instrument or payload, specifications as per the next section, will have to be entered and the TSI will generate architectures permuting and combining them if heterogeneous constellations are allowed by the user (as per orbit specifications). So far, the ED and TSI can support one payload per satellite however the software and interface control is set up such that the number of payloads can be scaled up easily (albeit without checking for inter-payload interference).

For every numeric variable in this class, the user can specify a range of values (e.g.

alongTrackSlewOfCenter = 40° to 60°), in which case the TSI will generate unique architectures by uniformly sampling the provided range, allowing the user to see the trade-offs choosing one over the other. If he/she is not interested in a variation, a single numeric value should be entered. The Observatory class contains methods to read relevant sections of the TSR, compute number of LVs needed to launch the constellation (called by the TSI based on type) and appropriateness of available ground stations.

2.1.4. *Instrument Specifications*

This class allows the ED to organize user inputs related to the physical characteristics of the payloads or science instruments on each satellite. It contains methods to read relevant the inputs from the TSR, to compute approximate instrument size needed by the CR module based on the spectral bands, and total field of view (FOV), instantaneous FOV, sensor shape or solar conditions as needed by the RM/OC.

Since the TSI will be expanded to support three types of instrument, the user is required to select which concept of operations will be used and enter its corresponding mass, power, volume, technology readiness level (TRL), resolution, time taken per image or measurement and its spectral characteristics. Depending on the type of payload, he/she may have to enter customized specifications. For example, if it is an occultation or stereo payload, the user will have to select the partner objects (for example, the GPS satellites) that the new constellation is occulting against or performing pair-wise functions with, respectively. For specifying this instruConopsPartner, the user can enter a filepath, where the file will have the same format as the existing satellites. If it is an imaging payload, he/she will have to specify the range of FOV – total and instantaneous – for making the images. The FOV can be along and cross track because the RM and OC modules allow for custom shaped sensors. If it is an occultation payload, the user will have to select the stationary object of interest (for example, the Sun) and enter the range of altitudes above the Earth that qualifies as an occultation to be measured. The other variables listed in the class, such as the payload data rate, radiometric resolution and measurement rate will inform performance and limits on satellite size.

As with the observatory, the user can specify a range of values for every numeric variable in this class (e.g. fovCT = 15° to 30°), in which case the TSI will generate unique architectures by uniformly sampling the provided range, allowing the user to see the trade-offs choosing one over the other. If he/she is

not interested in a variation, a single numeric value should be entered. The Instrument class contains methods to read relevant sections of the TSR and compute angular inputs for the RM and OC modules related to FOV, solar conditions and sensor shape. Currently, only one instrument per satellite is supported but the ability to add multiple-payload, heterogeneous satellites will be available in the future.

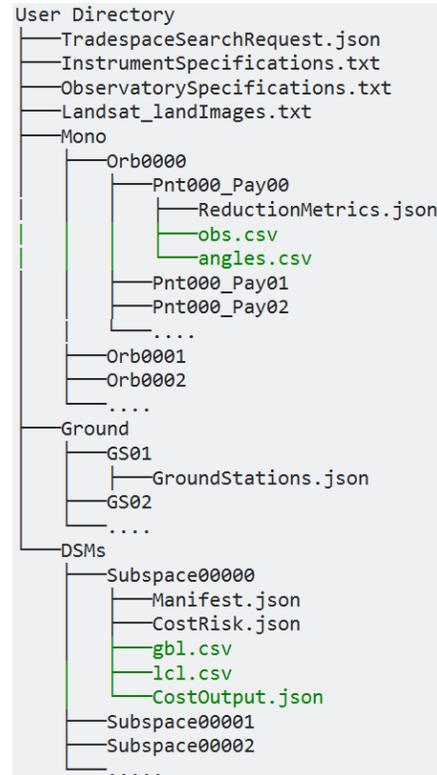*2.1.5. Output Options and Bounds*

This class allows the ED to organize user's preferences on outputs and minimum/maximum bounds on these outputs. The ED can instruct the RM to compute only those output variables he or she is interested among the full list of options (see Figure 3 and associated text), as well as any constraints on the range of those output values. For example, he/she can set revisit time to be between 1 day and 1 week and spatial resolution between 100m and 1 km, and the ED will ensure that the RM returns only those architectures that meet those requirements. The output options are listed will be discussed in more detail in Section 0. The Outputs class contains methods to read relevant sections of the TSR and filter the results produced the TSI and RM to meet the user's cut-offs.

2.2. Tradespace Search Iterator

The role of the TSI is to enumerate the full tradespace of constellation architectures that fit the user requirements and generate a file tree of results within a user selected folder. Architectures are currently full factorial combinations of number of satellites, constellation types, inter-plane and intra-plane distribution, LV options, GS options, and sampling of the full range of instrument and observatory specifications, orbital altitude and inclination. To avoid tradespace explosion, the TSI restricts the maximum number of unique altitude, inclination, observatory and instrument specification values to four, but the user can change this number easily. Any satellite in the DSM tradespace is defined as an object of the Spacecraft() class – initialization screenshot below. Spacecraft variables are objects of the classes defined in Sections 2.1.3 and 2.1.4 and a KeplerianElements() class, whose variables are the six Keplerian elements for any single satellite in the DSM tradespace and which the TSI ensures fits the requirements of orbit specifications in Section 2.1.2 and mission concept in Section 2.1.1. Any ground station an object with a list variable, defined by the ground network parameters in Section 2.1.1. A constellation architecture is a member of the Constellation() class whose objects have variables – satellite indices, ground indices and type.

```
class Spacecraft:
    def __init__(self,satSpecs=ObservatorySpecifications()
                 ,paySpecs=InstrumentSpecifications(),
                 orbSpecs=KeplerianElements()):
        self.satSpecs = satSpecs
        self.paySpecs = paySpecs
        self.orbSpecs = orbSpecs
```

```
User Directory
├──TradespaceSearchRequest.json
├──InstrumentSpecifications.txt
├──ObservatorySpecifications.txt
├──Landsat_landImages.txt
├──Mono
│   ├──Orb0000
│   │   ├──Pnt000_Pay00
│   │   │   ├──ReductionMetrics.json
│   │   │   ├──obs.csv
│   │   │   └──angles.csv
│   │   ├──Pnt000_Pay01
│   │   ├──Pnt000_Pay02
│   │   └── ....
│   ├──Orb0001
│   ├──Orb0002
│   └── ....
├──Ground
│   ├──GS01
│   │   └──GroundStations.json
│   ├──GS02
│   └── ....
└──DSMs
    ├──Subspace00000
    │   ├──Manifest.json
    │   ├──CostRisk.json
    │   ├──gbl.csv
    │   ├──lcl.csv
    │   └──CostOutput.json
    ├──Subspace00001
    ├──Subspace00002
    └── .....
```

Figure 2: Example of a file tree created after a full run initiated by the TSI, within the user's selected directory. The .csv files in green are generated after RM and OC complete their runs.

In Figure 2, the 'Ground/', 'Mono/' and 'DSMs/' file trees are generated by the TSI and each populated by the files shown in non-green text. The files in green text are created after the RM, OC and CR modules have completed their runs. The TSI creates a sub folder under the Mono/ (implying Monolithic spacecraft) corresponding to every unique satellite and its orbit in the DSM tradespace, which in turn contains a sub-folder corresponding to every unique payload and pointing strategy per unique orbit. ReductionMetrics.json contains the satellite, orbit, payload and pointing strategy specifications per the MissionConcepts(), Spacecraft() and Ground() classes. The TSI also creates a sub folder under

Ground/ corresponding to every unique ground station network and writes a GroundStations.json file with the network's specifications. It creates sub folders called 'Subspace' under the DSMs/ folder, corresponding to very unique architecture, it generates and writes the Manifest.json. The JSON file contains a list of pointers to subfolders in Mono/ and Ground/, per the Constellation() class - within it. An example for a DSM with two spacecraft and one GS is:

```
{
    "ConstellationType": 1,
    "GroundStations": [
        "GS000"
    ],
    "MonoSats": [
        "Orb0000_Pnt000_Pay00",
        "Orb0002_Pnt000_Pay00"
    ]
}
```

Sections 3, 4 and 5 in this paper will describe the TSI's architecture generation for different constellation types and ground station networks, and their impact on the performance metrics listed in Section 0. The RM module can be called based on the file tree, and the CR module needs some additional parameters related to the constellation architecture's maintenance – See [5] and Section 4.3. The JSON-based file I/O ensures that all specifications (TSR, ground, mono and DSM) can be easily read by a human user and a computer. The file tree structure ensures there is no repeated information taking up unnecessary disc space on the user's computer.

2.3. Data Reduction and Metric Module

The RM module can be called by the ED or TSI, once the entire file tree and all constellation objects have been created by the TSI, to create and populate results files within every folder – shown in Figure 2 (except CostOutput.json, which is a CR module output). The RM module traverses every architecture/subspace in the DSMs/ folder and identifies the Mono/ spacecraft it is made up of. It then processes all the Ground/ folders in the file tree, corresponding to all DSM architectures, and stores its characteristics in memory. RM calls the OC module, to propagate the individual satellites per architecture per time step, by giving OC the satellite's initial conditions, propagation time step, grid point size, lifetime, mass and volume and propagation fidelity. After every propagation step, the RM calls OC for every unique payload and stores the corresponding coverage events (for all the grid points within the user's area of interest and for all ground stations) in memory. Runtime calls are organized such that no same orbit is propagated twice and no orbit-ground point coverage event is computed twice, even by part. Then for each DSM architecture/sub space, the RM uses the coverage events (stored in memory) for its constituent spacecraft to compute total coverage for the constellation. Finally, it computes all the outputs requested per architecture, and stores them within the DSMs/ "Subspace" folders

Time [s]		TimeToCoverage[s]			AccessTime [s]			RevisitTime [s]			Coverage	NumOfPOIpasses			
t0	t1	TCavg	TCmin	TCmax	ATavg	ATmin	ATmax	RTavg	RTmin	RTmax	% Grid Covere	PASavg	PASmin	PASmax	
0	2592000	200098	27	906039	19.3808	0	54	242682	5832	906282	99.3647	12.2833	0	102	

	Data Latency [s]			NumGSpasses	TotalDownlinkT	DownlinkTimePerPass [s]			CrossSwath[km]			AlongSwath[km]			SpatialResolution [m]	
PASmax	DLavg	DLmin	DLmax	PassesPerDay	DLTimePerDay	DLTavg	DLTmin	DLTmax	CSavg	CSmin	CSmax	ASavg	ASmin	ASmax	SRmin	SRmax
102	15500	5454	36882	5.4	2101.5	2.40226	54	486	185.852	185.852	185.852	185.852	185.852	185.852	29.5368	32.0538

Time [s]		POI	[deg]	[deg]	[km]	AccessTime [s]			RevisitTime [s]			TimeToCove	Number of Passes
t0	t1	POI	lat	lon	alt	ATavg	ATmin	ATmax	RvTavg	RvTmin	RvTmax	TCcov	numPass
0	2592000	128	82.8571	0	0	14.2105	0	27	46232.2	5859	83187	1485	57
0	2592000	129	82.8571	11.6129	0	18.4737	0	27	46227.9	5859	83160	1458	57
0	2592000	130	82.8571	23.2258	0	20.25	0	27	47066.9	5859	83160	1431	56
0	2592000	131	82.8571	34.8387	0	21	0	27	47273.5	5859	83160	78705	54
0	2592000	132	82.8571	46.4516	0	20.5	0	27	47385	5859	83160	72792	54
0	2592000	133	82.8571	58.0645	0	19.1455	0	27	46399.5	5859	83160	72765	55
0	2592000	134	82.8571	69.6774	0	16.5	0	27	47278	5859	83160	66852	54
0	2592000	135	82.8571	81.2903	0	11	0	27	47283.6	5859	83187	66825	54
0	2592000	136	82.8571	92.9032	0	5.89091	0	27	46304	5859	83187	66798	55
0	2592000	137	82.8571	104.516	0	2.41072	0	27	45572.6	5886	83187	60885	56

Time[s]	Ecc[deg]	Inc[deg]	SMA[km]	AOP[deg]	RAAN[deg]	MA[deg]	Lat[deg]	Lon[deg]	Alt[km]
43	3.82E-16	98.2081	7083.14	0	0.00049052	2.60613	2.59503	97.9013	705.047
86	2.87E-16	98.2081	7083.14	0	0.00098105	5.21226	5.18983	97.3483	705.177
172	3.33E-16	98.2081	7083.14	0	0.00196209	10.4245	10.3778	96.2315	705.692
215	6.10E-16	98.2081	7083.14	0	0.00245261	13.0306	12.9704	95.6645	706.073
258	4.30E-16	98.2081	7083.14	0	0.00294314	15.6368	15.5618	95.0894	706.532
344	5.62E-16	98.2081	7083.14	0	0.00392418	20.849	20.7401	93.9072	707.667
387	3.73E-16	98.2081	7083.14	0	0.00441471	23.4552	23.3264	93.2956	708.335
473	1.44E-16	98.2081	7083.14	0	0.00539575	28.6674	28.4919	92.0186	709.841

Figure 3: Example of gbl.csv (top in green) and a small snapshot of lcl.csv (middle in blue) and obs.csv (bottom in red), per Subspace folder, generated by the RM. Each column represents a different output variable. gbl.csv has 3 columns but 32+ columns, thus shown as two snapshots. lcl.csv and obs.csv can have thousands of rows, depending on time steps.

as two csv files: 'lcl.csv' contains outputs per point of interest (POI) on the Earth across all time periods and 'gbl.csv' contains average, maxima and minima of each output across all POIs, per time period. The presented outputs are in the context of Earth imaging only. Future efforts will extend the presented tool and principles to occultation and bi-static radar missions.

Figure 3 shows the science performance attributes the RM can output per DSM architecture or Subspace # (lcl.csv, gbl.csv) and per Mono/ sub folder (obs.csv), as seen in the Figure 2 tree. In addition to this, it also outputs the one 'angles.csv' file per Mono/ sub folder. 'obs.csv' contains rows equal to the number of time steps requested by the user in the performance period, and columns contain the Keplerian elements, latitude, longitude, altitude for the monolithic satellite whose folder it is stored in. 'angles.csv' contains the contains the view zenith, view azimuth, solar zenith, solar azimuth for every POI, per time step specified by the user over the performance period, for that particular satellite. If the user does not select a time step, the output defaults to the orbit propagation time step. The information from these files per architecture or satellite is made available to the user information for interactive visualizations or simple download.

In lcl.csv, metrics are listed per POI (numbered in column 3) over the time horizon from t0 to t1 (column 1-2). Access time shows the average, minimum and maximum continuous time period that POI has been accessed by any satellite in the given constellation. Revisit time is shows the average, minimum and maximum revisit time for that POI over all satellites in the given constellation. Time to Coverage is the time taken for the constellation to access that POI and number of passes totals the number of times it was accessed thereafter. Note that revisit time is the frequency at which a POI is revisited at any view angle while repeat time is the frequency at which it is revisited at exactly the same view geometry. A consistent repeat time is very hard to maintain without significant propulsion. In gbl.csv, the metrics are computed over the entire mission duration (t0 to t1 in the first two columns) and over all POI, therefore no time or POI series. If the user is interested in those metrics for a specific section of the mission duration, he/she may a different time range for the key 'PerformancePeriod' in the TSR (Section 2.1.1). 'Time to Coverage' is the time taken by the constellation to cover every POI, as an average minimum, maximum. TCmax is the time to achieve global POI coverage. Revisit and Access time statistics are computed by RM module over the corresponding lcl.csv values. The percentage of grid points covered within the requested Area of Interest allows the user to determine the

relative fraction of interested regions covered among the different architectures. For example, an ISS orbit will provide better revisits of the mid-latitudes but will not cover the poles. The user can select a few DSM architectures of interest, based on these overall architecture comparisons, and then use time series data per satellite in the selected DSMs to run a more science specific, observing system simulation experiment to evaluate the architectures against one another at higher fidelity.

Statistics on the number of POI passes indicate the number of times any satellite has seen any POI. RM also outputs ground downlink related metrics – data latency i.e. the time taken between any two downlink accesses, number of passes over any GS by any satellite, total available downlink time from all satellites in a day and in one GS pass. For the spatial metrics - along or cross swath correspond to the total ground image size, and 'Spatial resolution' contain the statistics of the ground pixel size the given constellation supports.

Future versions of the RM will be able to compute the signal to noise ratio (SNR) of every satellite in every DSM relative to a reference satellite in that DSM, not an absolute SNR, because factors common to all satellites such as the noise temperature, antenna gain, etc, are not architecture differentiating and DSMs can be better compared to each other using relative SNR. Reference[5] contains results from the RM tested on two use cases, and validated against results from AGI's Systems Tool Kit –
http://www.agi.com/products/engineering-tools.

2.4. Orbit and Coverage Module
The orbits and coverage component, or OC, is responsible for modelling orbital motion and identifying the earth grid points in view at each propagation step. The OC is always called by the RM module using C++ method calls. Orbital models were selected to balance performance and accuracy appropriate for early design phases. Coverage is computed based on grid points that can be optionally computed via standard grid spacing algorithms or provided by the user. OC is also responsible for generating the POI grid, per the grid size provided by the RM (such that the FOV is Nyquist sampled), if the user has provided latitude/longitude bounds to his/her area of interest and not a full POI list. Improvements to OC since Reference[5] include the ability to enter a custom sensor module, instead of being limited to a conical sensor, and modelling of atmospheric drag in propagation computations.

The OC module is capable of supporting sensors in any shape, including combs or rhombus or other irregular shapes, and efficiently computing all grid points that lie within it at any given time. Figure 4 shows an example of a sensor's dynamic simulation where a grid point is computed as being in (left) or out (right), at the shown snapshot of time. The RM can specify any custom sensor for the OC by defining a vector of cone and clock angles (as pairs) that defines a closed sensor field of view. Cone angles are measured from +Z sensor axis. If xP, yP, zP is a unit vector pointing nadir from the satellite to the Earth, the cone angle for the detic point is 180 - asin(zP). Clock angles (right ascencions) are measured clockwise from the + X-axis, therefore the detic point above has a clock angle of atan2(y,x). Currently, the TSR is structured to allow users to set conical or rectangular sensors only, and input the corresponding along and cross track FOV and iFOV. The RM module converts the FOVs into OC readable cone and clock angles using the equations below.

$$cone = \cos^{-1}(\cos(alFOV/2)\cos(crFOV/2))$$
Equation 1

$$clock = \sin^{-1}\left(\sin(crFOV/2)\middle/\sin(cone)\right)$$
Equation 2

Future versions are expected to allow users to customize sensors by providing cone and clock angles as well as input scanning dynamics, especially for runs requesting the highest level of fidelity. The RM computes the required grid size, in the case of a non-custom grid, and time step by Nyquist sampling the along track FOV.
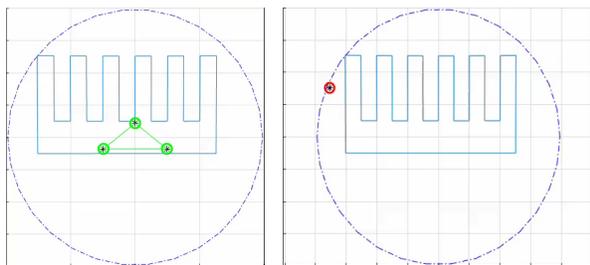
Figure 4: 2D visualization of the custom sensor where three given POI acknowledge that they are within a comb-shaped sensor (green – left) or outside it (red – right), as part of a dynamic simulation.

OC can also propagate orbits with drag included, if the TSI calls the RM with the no maintenance required flag on. In this case, the RM and OC use the satellite's physical specifications to compute mass and drag area, as passed via the JSON file within each Mono/ sub folder, seen in Figure 2. Drops in satellite altitude due to drag can be monitored via the outputs generated by the RM in obs.csv, seen in Figure 3-bottom.

3. Types of Constellations

This section discusses several different DSM architectures that the TSI can generate, by permuting four types of constellations *for the same design variables of altitude, inclination, number of satellites and instrument FOV*, leading to improved diversification of the design space. The differentiating element between the constellation type architectures is the spread of the initial Keplerian elements of every constituent satellite, as passed by the TSI to the RM within the JSON file inside each Mono/ sub folder (Figure 2). Constellation type has impact on deployment strategy, LVs and maintenance (Section 0), i.e. inputs for cost, and constellation performance (Section 4.1), therefore presents interesting trade choices.

Currently, the TSI can generate Uniform Walker Constellations, where all the satellites have the same altitude, inclination and equal satellites per plane; Non-Uniform Walker constellations (combinations of uniform Walker constellations over different altitude, inclination and possibly satellites per plane); Precessing constellations where satellites are dropped off by a single LV at differential altitude and inclination causing them to disperse over time for a large RAAN and mean anomaly (MA) coverage; and Ad-Hoc Constellations where satellites are launched as per the next available launch option per ED's LV database. To our knowledge, precessing constellations have not been analysed in a tradespace before and we provide a new approach for enumerating ad-hoc and heterogeneous constellations, while keeping the tradespace tractable.

3.1. Homogeneous Walker Constellations

This type of constellation contains satellites with similar orbits, eccentricity and inclination so that any perturbations affect each satellite in approximately the same way[7]. This allows preservation of the geometry without excessive fuel usage. Walker constellations have been extensively studied over the last few decades and optimized for Earth coverage[8], [9].

For any given number (N×S) of satellites in the tradespace, the TSI first computes all ways in which they can be arranged in N planes with S satellites per orbital plane. For every factorization of N×S, Walker constellations can be of at least two types – Star and Delta. Prior analysis has shown Delta to be better in terms of the "Time to achieve global coverage" metric[10]. However, at any given instant of time, the

fraction of the Earth seen by Star is more. Additionally, any of the above Walker designs can be arranged in N ways due to N types of phasing between the satellites in the same plane. Therefore, for this constellation type alone, there can be n(altitude)×n(inclination)×n(FOV)×n(NS) ×n(factors(N×S))×N×2 architectures, where n(X) is the cardinality of the design vector X. To make the design space more tractable, the TSI generates only Walker Star constellations with inter-satellite phasing of floor(N/2) because the design allows the most instantaneous coverage of the Earth. This reduces the number of architectures by a factor of 2N. The TSI also assumes only circular orbits.

3.2. Heterogeneous Walker Constellations

This type of constellation is arranged like a Walker with circular orbits and equal number of satellites per orbit, but the orbits can be of different inclinations and altitudes. Since each orbital plane in a Walker has to be launched by a separate LV, it allows flexibility is choosing a different orbit at every launch. Lower inclination orbits cover equatorial regions more frequently but miss the poles entirely, while higher orbits cover the higher latitudes frequently at the cost of lower ones. A combination of both could get the best of both. A similar argument can be made for combining orbits at different altitudes – some give higher coverage and others higher spatial resolution.

The TSI uses all the orbital planes generated in the homogeneous Walker type as a design vector and permutes them in all possible ways to generate heterogeneous architectures. For example, let's say altitude = [A1 A2], inclination = [I1 I2] and all other variables are held at constant values for simplicity. Homogeneous Walker will produce four variations of alt-inc: {A1,I1}, {A1,I2},{A2,I1}, {A2,I2}. Every homogeneous Walker plane created, i.e.

N×n(factors(N×S)) for N×S satellites in a constellation, will have four alt-inc variations. Thus, when creating heterogeneous Walker constellations, with a given number of planes (factorize again), the TSI will have up to 4N×n(factors(N×S)) planes to choose from. While these combinations increase the number of architectures, we have found these extra architectures sometimes outperform the homogeneous ones, at slightly higher maintenance cost.

3.3. Precessing Constellations

This type of constellation can be initialized by a single launch vehicle (assuming all satellites fit in the payload bay). RAAN and MA spread can be achieved over time, owing to the differential J2 effects between the satellites, caused by differential altitude and inclination and leading to differential precession of their RAANs and MA. Figure 5 shows an example of six satellites in such a constellation when deployed, after three and then six months. Only LVs with relight capability will be able to deploy them because each drop is at a different altitude and inclination, and the booster is expected to fire to achieve the delta-V required. The satellite RAANs and MAs will continue to precess after maximum spread has been achieved, unless they have propulsive means to correct their altitude and inclinations to a common value.

The altitude and inclination of the first drop-off will be called the chief orbit, and the combination can be selected from the regular alt-inc tradespace described in Section 3.2. The time required to spread out in RAAN is a function of the chief orbit and the small differentials of all others with respect to it. For example, Figure 6 is a contour plot of the days required by two deployed satellites to spread by 90° in RAAN, for any combination of differential inclination (Y-axis) and altitude (X-axis) for two chief orbit

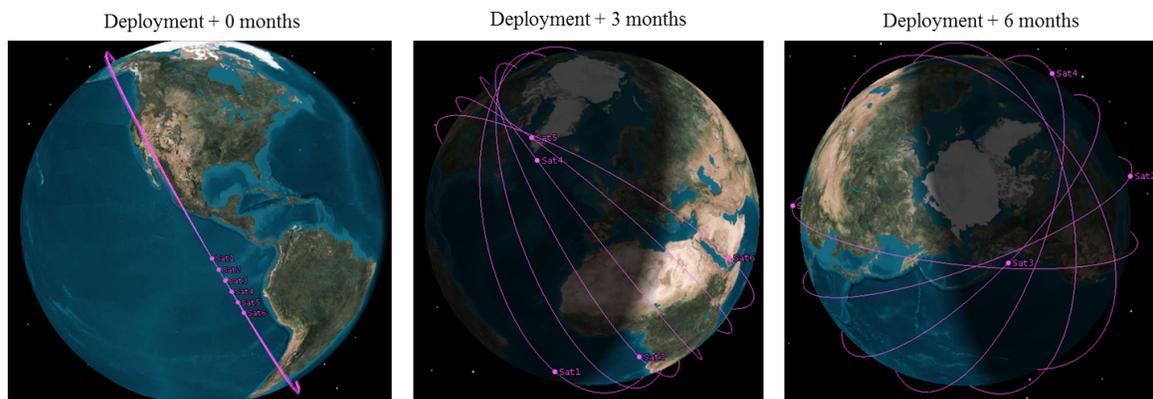| Deployment + 0 months | Deployment + 3 months | Deployment + 6 months |

Figure 5: Evolution of a precessing constellation over six months after deployment, generated on AGI's STK.

altitudes (blue vs. Red) and inclination (solid vs. dashed lines) combinations. The simulation in Figure 6 used three relights to drop off four satellites. Comparison between the red and blue near-parallel lines shows that higher chief altitudes precess slower. Comparison between (criss-crossing) solid and dashed lines shows that chief inclinations impact rates in a more complex manner. Higher inclinations reduce the slope of the rate contours, i.e. so the chief orbit determines which differential will get better rate returns. Equatorial orbits benefit more from differential altitude than inclination increases, at the scale shown in the figure. However, the current scale of the two axes is like comparing apples to oranges, and simulations of ΔV and fuel required to connect the two.
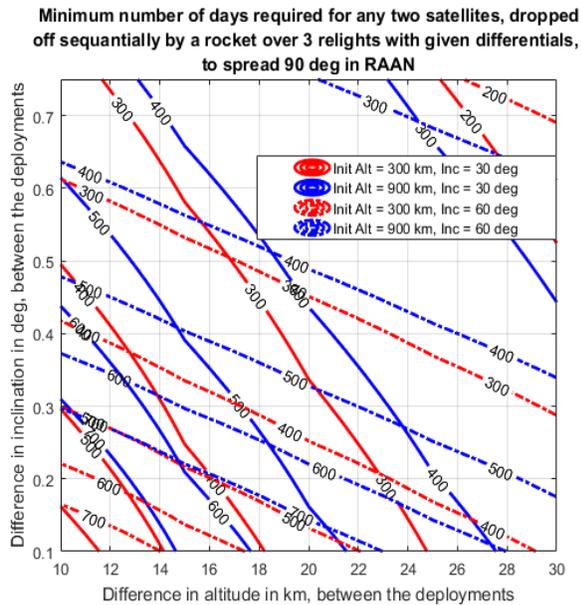
Figure 6: Time required for the first and last satellites to be positioned in perpendicular orbital planes after being dropped off by a Pegasus rocket deploying a four sat constellation.

The possible differentials in Figure 6's axes are a function of the deployer rocket's ΔV and relights available. Figure 7 shows the achievable combinations of differential altitude and inclination between consecutive satellites dropped off when four satellites are deployed using three relights, as a function of chief orbit altitude and total ΔV available in the rocket. The TSI keeps a 30% margin on fuel estimates to account for uncertainty in pre-Phase A designs. The results are independent of the satellite mass and inclination of the chief orbit. Equation 3was used to compute the differentials – Δh and Δi – from a given total ΔV,

assuming the drop-offs are evenly spaced. Note the same axes range between Figure 6 and Figure 7, and the larger contribution of Δi in achieving spread.

$$\Delta V_{total} = \sqrt{\mu/r} + \sqrt{\mu/[r + n\Delta h]}$$
$$+ 2\sin[\Delta i/2]\sum_{m=0}^{n-1}\sqrt{\mu/[r + m\Delta h]}$$
Equation 3
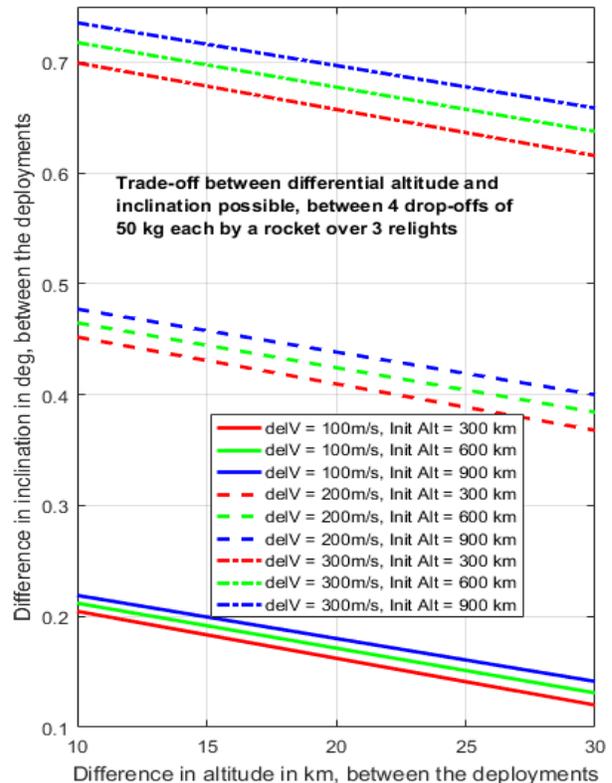
Figure 7: Differential altitude and inclination between consecutive satellites dropped off as a uniform spread, for different chief orbits and ΔV.

Uniform drop-offs are rare in a practical deployment scenario because ΔV available is a function of fuel and mass left on the rocket. Instead, launch providers allocate approximately equal amounts of fuel for each relight and the Δh and Δi is slightly more for each subsequent drop-off. Figure 8 shows the total ΔV available for N+1 payload drop-offs for N relights by the Orbital ATK's Pegasus rocket. We assume the Hydrazine Auxiliary Propulsion System (HAPS) manoeuvring stage is attached for precise insertion into the chief orbit. The system has a dry mass of 177 kg including wiring, can carry up to 57 kg

of Hydrazine propellant, support upto 200 kg of payload and relight at least seven times[11]. The payload mass per drop-off is computed as 200kg/(N+1). While the simulation shows more total ΔV available for more relights, due to the advantage of reducing payload mass, this will likely be countered in reality by inefficiencies in re-starting the booster. For any given rocket and chief orbit, the TSI computes the available ΔV, then a tradespace of combinations for Δh and Δi based on the number of drop-offs (Figure 7) and/or the number of days within which required coverage needs to be achieved (Figure 6).
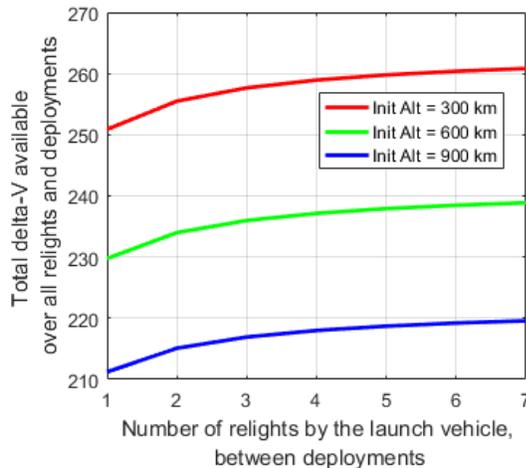
Figure 8: Total expended ΔV over all satellite drop-offs for a Pegasus rocket at variable chief altitudes and relights.

3.4. Ad-Hoc Constellations

Ad-Hoc Constellations have been investigated over the last few years[12] in connection to cheaper options for launching Cubesats as secondary payloads. Additional advantages are that different launch opportunities can be utilized to tailor a constellation for a specific region or mission objective and augmented using multiple launch opportunities. Such secondary launch opportunities exist not only for Cubesats via the Poly-Picosatellite Orbital Deployer but also for <180 kg class satellites due to the availability of the EELV Secondary Payload Adapter ring on Ariane-V class rockets. Spaceflight Inc makes upcoming launches with secondary space available through their website multiple years in advance – http://spaceflight.com/schedule-pricing/ making secondary launches an important option in EO tradespace exploration. The downsides are that the satellite orbit cannot be selected by its developers and final approval resides with the owner of the primary payload due to the potential increased risk the secondary spacecraft could pose to *their* mission. Also,

Ad-hoc launches are separated in time, causing a delay in full operations if many satellites are to be launched.

Typically ad-hoc constellations have been simulated by choosing from upcoming launches, for e.g. using Spaceflight's website. We propose an alternative simulation of ad-hoc constellations using the fully functional Planet [13] Flock constellation - currently the largest constellation in history. Planet, earlier known as Planet Labs Inc.[13] is excellent example of ad-hoc constellations because they launch their 3U CubeSat imagers (called Doves) on secondary launches, many at a time, whenever launches become available. As of August 2017, Planet has 192 functional satellites in orbit, and as of February 2017, have been imaging every point on the globe daily. The most current states of the Planet Labs satellites, is available open-access available online at: http://ephemerides.planet-labs.com/.
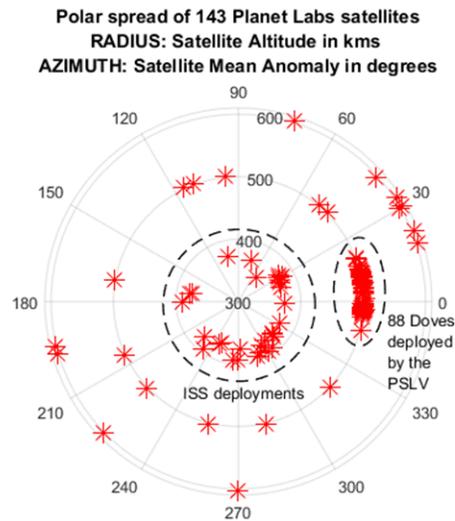
Figure 9: Orbital spread of the Planet satellites as analysed on Feb 18, 2017

As of February 2017, 143 satellite states have been extracted from Planet's open access database and stored within TSI's library, and can be updated easily. The TSI simulates an ad-hoc constellation by randomly choosing from the library, for any given number of satellites in the tradespace. Figure 9 shows their spread in MA (azimuth) and altitude (radius). The orbit distribution is as follows - 100 Doves in a 500+/-3 km SSO, 11 Doves in 600+/-3 km SSO, 32 Doves in the ISS orbit below 400 km. The figure shows the 88 Doves deployed on Feb 15 2017 by the PSLV rocket, as analysed two days later. The MA spread then was 27.5° since they were deployed within 10 minutes, and has spread evenly since then. The Planet database thus provides a representative set to base ad-hoc constellations on.

4. Simulation Results of Different Types of Constellations

The RM module outputs all results as csv files per DSM and single satellite folder, after completing its run. All performance metrics described in Section 2.3 can be generated currently for any user-defined inputs from Section 2.1, making this among the largest tradespace exploration tools for constellations in open literature. Figure 3 shows snapshots of three of four csv files for a DSMs/ architecture with one satellite in a Landsat-like orbit (710 km, 98.2°), a 15° rectangular FOV for the instrument and a single ground station located at Wallops Island (VA, USA), simulated over one month. Its state for the first 473s at user defined time steps (~40s) is also shown, as stored in the Mono/ folder. The swath does not vary over the performance period (same min, max, average) because it is a rectangular sensor on a circular orbit. The slight variation in spatial resolution is between the nadir and limb pixels. The downlink results (~5 passes per day approximately 4-5 hours apart) are expected for the Wallops station. As confirmed with the NEN loading manager, this allows ground operators to work only during business hours and yet support single spacecraft flagship missions. Almost (but not completely) 100% of the Earth was covered in 30 days because the orbit is an SSO, thus the poles cannot be seen. The average and maximum revisit time is 3 days and 11 days respectively, which has been validated against STK simulating the same scenario. Simulation results for the same observatory/payload and spatio-temporal distribution of metrics has been detailed in Reference [5], for up to eight satellites restricted to homogeneous Walker constellations only. It also contrasts these results against another scenario with up to 12 satellites containing wide-angle radiometer payloads.

The TSI's ability to simulate more types of constellations than Walker improved the diversity of the tradespace, increased trade-offs and revealed better performing architectures. For example for precessing constellations alone, higher altitudes allow lower available ΔV per Figure 8 and precess slower per Figure 6, therefore are not a good choice for achieving RAAN spread faster. However, they do provide more coverage and faster revisits than lower altitudes for the same FOV. Performance analysis over the full tradespace is therefore essential for assessing a good balance.

4.1. Impact on Science Performance

The effect of constellation type on performance is shown below for a scenario of four satellites in low Earth, SSO orbits with a. 89.45° FOV sensor – in consultation with NASA GSFC's Global Modelling and Assimilation Office (GMAO). The primary performance metric was global coverage over a 6 hour period, to aid weather predictions and research. Other metrics seen in Section 2.3 have been analysed, but have not been discussed here, for brevity.
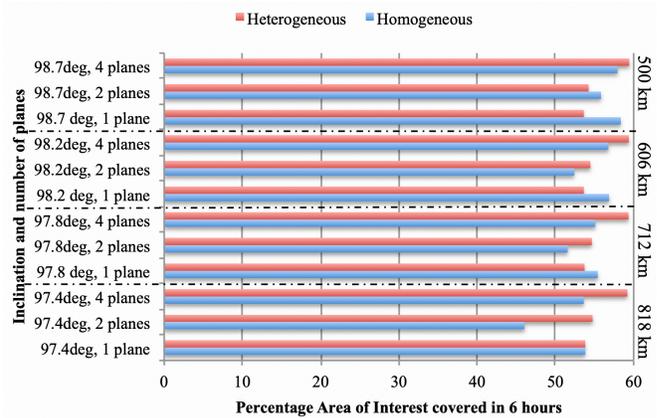

Figure 10: Variation of metric "% Grid covered" for homo and heterogeneous 4-satellite Walker constellations, arranged in 1-4 planes and 4 different inclinations. Orbital altitudes can take values from [500, 606, 712, 818] km.
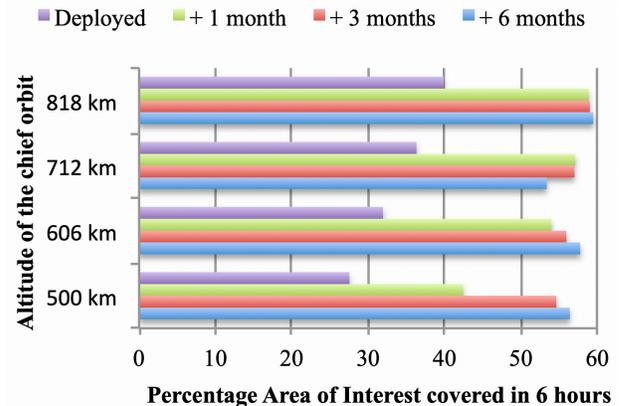

Figure 11: Variation of metric "% Grid covered" for precessing type 4-satellite Walker constellations for varying altitudes, restricted to SSO, as analysed just after to up to 6 months after deployment.

Heterogeneous constellations cover more than homogeneous constellations at the same inclination, as seen in Figure 10 for altitude heterogeneity of 500, 606, 712, 818 km. While increasing inclination slightly (4 blocks segregated by the dashed lines) obviously increases global coverage, the difference was is negligible. For homogeneous constellations, the four blocks correspond to 818, 712, 606 and 500 km (top to bottom) for all their orbital planes respectively, so greater altitude meant slightly more coverage
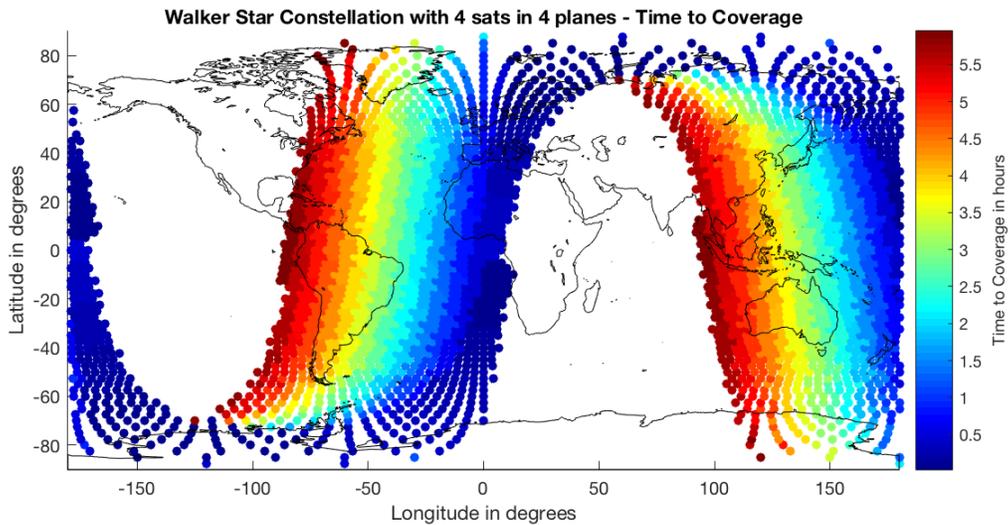
Figure 12: Spatial distribution of time to coverage of every global grid point (in hours) for the best performing 4-satellite constellation in the simple scenario presented in Section 4.1. Such spatial distributions can be generated from the lcl.csv file in any architecture within DSMs/, after selecting them by comparing simplified statistics in gbl.csv.

respectively. For both homo and hetero constellations, more number of planes led to more coverage, all else being equal. For heterogeneous constellations, the orbital planes always had different altitudes, so the 4-plane heterogeneous case over the four blocks separated by dashed lines indicated almost the same constellation, therefore same coverage (verification). Two plane constellations performed worse than one-plane ones – a counter-intuitive result, but one that points to better coverage at better cost.

There is an obvious improvement in performance by precessing constellations over time (Figure 11), so much so that it outperforms homogeneous constellations after three months and matches heterogeneous constellations, with the advantage of lower altitude and better spatial resolution. The metrics in Figure 11 are computed over a 6-hour period starting 0, 1, 3 and 6 months after deployment (easily changeable by the user using the PerformancePeriod variable discussed in Section 2.1.1). Comparing the similar colored lines shows improved coverage with altitude, as expected, especially closer to deployment. The orbits spread out over time to make coverage more uniform, altitude notwithstanding. This simulation shows that if this mission can wait for a few months after deployment to achieve required performance, 4-satellite precessing constellations can provide as much coverage as a 4-plane Walker constellation (whose four launches may take a few months anyway), while using one LV instead of four.
Figure 10 and Figure 11 have been made by processing the 'Coverage - % Grid Covered' metric in the gbl.csv

files, for the GMAO scenario run on MS Excel, to compare RM-generated results obtained from TSI-generated architectures. See Figure 3 as an example for one DSMs/ folder's csv files. The 'best performing' constellation over six hours was a heterogeneous one that achieved 59.4% coverage, per the 'Coverage - % Grid Covered' metric among all gbl.csv files. Figure 12 shows the spatial distribution of grid points covered in terms of when they were first accessed, by processing on MATLAB, the 'TimeToCoverage' metric in the lcl.csv file within the best-performing DSMs/ folder. This simple scenario run shows that our tool allows easy perusal and visualization of results. TAT-C will have its own GUI which will process the .csv results and display results, therefore will be independent of MATLAB or MS Excel. Note that the 'best' performing constellation is among only those types described in Section 3, including a full factorial of variables described, and we expect further improvements in performance when more targeted, complex constellation types (e.g. Flower, secure route) are optimized to maximize a given specific output.

Ad-hoc constellations are a valuable alternative in performance only in much larger numbers than primary launch options. Reference[5] shows far lower maximum and average revisits by Ad-hoc constellations compared to homogeneous Walker, for the same satellite number. This explains Planet's goal[13] of keeping hundreds of satellites in orbit to match performance of smaller constellations, yet possibly cheaper due to launching ad-hoc on secondary

LVs. Results from RM and CR are expected to show the user similar trade-offs.

4.2. Impact on Launch Vehicle Manifest

This section discusses the TSI simulated launch strategy for each constellation type and its impact on performance and potential cost. The CR module currently assumes that the same LV is used to launch the entire constellation. For any DSM architecture, the TSI provides CR with the possible LVs that can launch it, selected from the ED module's textfile library described in Section 2.1.1, and the number of instances for any LV needed. For all satellite, instrument, orbit and GS parameters remaining equal, changing the LV generates new DSM architectures with different performance (because every LV's capacity is different, thus different manifest, spread and schedule) and cost. To avoid tradespace explosion, the TSI restricts the maximum number of unique LVs per DSM to four. The user can change this number easily.

The TSI computes one LV per orbital plane to be launched for Walker constellations. If LV's payload capacity is less than the sum of all satellites per orbital plane, multiple LVs are added per plane. The TSI allocates one LV per precessing constellation. If the LV's payload capacity is exceeded, multiple LVs are used and the launches are spread uniformly over 360° RAAN. For example, if each satellite in the GMAO scenario in Section 4.1 weighed 90 kg and the Pegasus LV is used (200 kg payload capacity), the TSI will determine the need for at least 2 LVs for any DSM architecture. The precessing architecture will be launched such that the 2 LVs deliver their 2 satellites 90° apart from each other in RAAN, for better spread at the same chief altitude and inclination. Since the precessing constellation spread is very intricately tied to LV characteristics (Section 3.3), the number of precessing DSM architectures is equal to the number of unique LVs selected from the ED's database. To avoid tradespace explosion, the TSI restricts the maximum number of unique LVs to four (user-changeable). The user can change this number easily. Last, the TSI allocates the minimum number of LVs needed to launch an ad-hoc constellation because secondary launch costs are per unit satellite mass and no consideration is given to orbit or arrangement.

The differences in the LV manifest per constellation also affects schedule, which affects cost and performance. Multiple launches have lags between them, averaging "mbtl" from Section 2.1.1 and the LaunchVehicle class, but with significant standard deviation. This may lead to performance plateaus over time that reaches full potential only after the whole constellation has deployed. In the absence of propulsion, the earlier orbits and instruments may start to degrade by this time. While the TSI, RM and OC currently captures the orbital aspects, DSM analysis should account for such time-dependent interactions between all design choices. For example, small differences in RAAN between deployments can be addressed by propulsive drop-offs by the same LV or powered EPSA ring, and orbital spares can decrease the risk of performance drops in case of satellite failures. We are planning to develop a separate launch module that can be called by the TSI or CR for higher fidelity allocation of launch, and assessment of its impact on performance and cost.

4.3. Impact on Maintenance Requirements

Different constellation types also affect maintenance requirements, as computed by the TSI and passed to the CR module. Maintenance is computed only if the user selects no propulsion, as described in Section 2.1.1. If not, the RM module ensures that the OC module is called with full drag and J2 enabled. Propulsive in-track maintenance is essential to hold uniform spacing between satellites and ensure continuous coverage, and computed by the TSI for Walker architectures. The TSI computes simple ΔV and manoeuvre frequency for altitude maintenance, to correct 5% or more drop in altitude for Walker or precessing constellations. See Section III.C in [5] for more detail. Maintenance is not computed for Ad-Hoc constellations because the spacecraft is not expected to be in the user's control or budget.

5. Simulation Results over Ground Station Networks

Ground station related design variables in Section 2.1 affect the performance metrics related to downlink and data latency in Section 2.3. The user can select from NEN, DSN or TDRSS networks or input his/her own. Currently, if TDRSS is selected, the RM outputs zero latency and 24/7 passes. Low altitude orbits (e.g. ISS) have downlink eclipses in spite of TDRSS, and this outage will be included in future work by improving the RM and OC module to compute space-to-space coverage events. This change is expected to support all space-based delay relays, such as Iridium or IntelSat. All the GS networks that TSI generates for all DSMs are stored in the Ground/ folder. The number of GS assigned to any DSM architecture depends on user requirements on downlink and latency. For all satellite, instrument, launch and orbit parameters remaining equal, increasing number of GS generates an increasing number of new DSM architectures with different performance and cost. To avoid tradespace

explosion when a user has not bounded his/her requirements, the TSI restricts the maximum number of GS to one per four satellites in the constellation. The user can change this number easily.
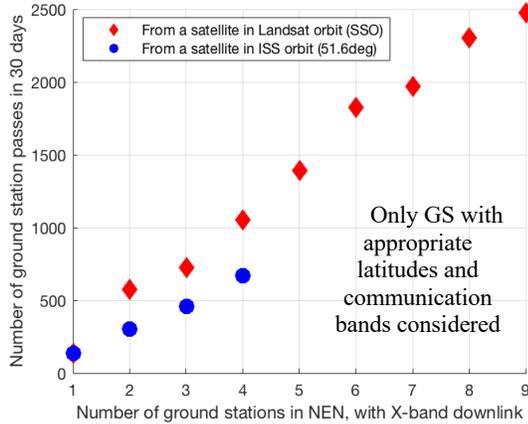
Figure 13: Variation of the number of GS passes with number of available GS, for a satellite with an X-band transponder.
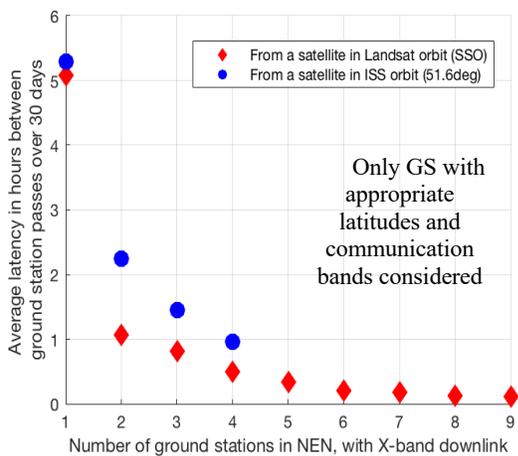
Figure 14: Variation of the average downlink latency with number of available GS, for a satellite with an X-band transponder.

This section describes results from the RM and OC module for a single satellite in two different but representative orbits, with an X-band downlink transponder and varying GS architectures. All GS were restricted to the NEN stations, both commercial and government. These results are used to streamline the number of GS required for a user-bounded latency. Figure 13 and Figure 14 show how the GS passes and average downlink latency, respectively, changes for increasing the number of GS for an ISS orbit (blue circles) and a Landsat orbit (red diamond). These values were computed by processing the "Data

Latency-DLavg" and "NumGSpasses" metrics respectively, in gbl.csv files over all DSMs/ folders (see Figure 3 as example) on MATLAB. The ISS orbit can access only four NEN stations because of its limited inclination of 51.6°.

Such plots can be generated for any DSM architecture by post-processing the RM module's csv outputs. Currently, the RM allows GS multiple access, however going forward, we will allow only one satellite to access a GS at a time. Latency is expected to decrease with increasing number of satellites, however single access will cause this decrease to taper off very soon unless the number of GS is increased. The complex interaction between number of satellites, orbits, transponders and GS networks, in terms of several user-defined downlink metrics, will be well captured using the tradespace analysis tools presented in this paper.

6. Summary and Future Work

This paper has described some of the components of the Trade-space Analysis Tool for Constellations (TAT-C) developed by NASA Goddard. TAT-C will provide a framework that facilitates DSM Pre-Phase A investigations and optimizes DSM designs with respect to a-priori Science goals. In this paper, we described the development of the executive driver, tradespace search iterator, orbit and coverage, and data reduction and metrics computation modules of TAT-C. We highlighted improvements to these modules since our last publication[5], especially the addition of new types of constellations in the TSI, inclusion of LV permutations, options to downlink to several options of GS networks and the availability of many new metrics such as swath, spatial resolution, downlink and latency. The effect of the improved tradespace was demonstrated via a few simple case studies. Our proposed modules have more design variables and performance outputs than published in academic literature before, enumerate launch-affordable constellations and support a customizable imaging sensor.

We will continue to improve the described modules and integrate with the rest of TAT-C. We will develop a high fidelity instrument module, to capture more details of imager payloads, include other payloads such as occultors and radars, and expand the TSI to allow multiple, heterogeneous instrument missions. This will also include modification of the RM and OC module to allow space-to-space observations and other concepts of operations, aside of nadir viewing imagers. The TSI attempts to keep the tradespace tractable by restricting: the combination of several types of

constellations (e.g. some satellites deployed as ad-hoc and others as Walker, non-uniform RAAN constellations), phasing between Walker planes to a pre-computed assumption, differential altitude or inclination combinations to a reasonable value, enumeration of ad-hoc constellations, samples within the range of altitude/ inclination/ FOV ranges to a handful of data points, combinations of GS and LV options to a maximum, etc.

References
[1] M. D&'Errico, *Distributed space missions for earth system monitoring*, vol. 31. Springer Science & Business Media, 2012.

[2] S. S. Board, National Academies of Sciences, Medicine, and others, *Achieving Science with CubeSats: Thinking Inside the Box*. National Academies Press, 2016.

[3] ARC Mission Design Center, "Small Spacecraft Technology State of the Art," NASA Ames Research Center, Moffet Field, CA, NASA/TP–2015–216648/REV1, Aug. 2016.

[4] J. LeMoigne *et al.*, "Tradespace Analysis Tool for Designing Constellations (TAT-C)," in *IEEE International Geoscience and Remote Sensing Symposium*, Fort Worth, TX, 2017.

[5] S. Nag, S. P. Hughes, and J. L. Moigne, "Streamlining the Design Tradespace for Earth Imaging Constellations," in *AIAA SPACE 2016*, American Institute of Aeronautics and Astronautics.

[6] S. P. Hughes, R. H. Qureshi, D. S. Cooley, J. J. Parker, and T. G. Grubb, "Verification and Validation of the General Mission Analysis Tool (GMAT)," in *AIAA/AAS Astrodynamics Specialist Conference, San Diego, CA*, 2014.

[7] J. G. Walker, "Satellite constellations," *J. Br. Interplanet. Soc.*, vol. 37, pp. 559–572, 1984.

[8] T. J. Lang, "Walker constellations to minimize revisit time in low earth orbit.," *Adv. Astronaut. Sci.*, vol. 114, p. 16, 2003.

[9] D. D. Chang and O. L. de Weck, "Basic capacity calculation methods and benchmarking for MF-TDMA and MF-CDMA communication satellites," *Int. J. Satell. Commun. Netw.*, vol. 23, no. 3, pp. 153–171, 2005.

[10] S. Nag, J. LeMoigne, D. W. Miller, and O. L. De Weck, "A Framework for Orbital Performance Evaluation in Distributed Space Missions for Earth Observation," in *IEEE Xplore, Aerospace Conference 2015*, Big Sky, Montana, USA, 2015.

[11] "Pegasus Users' Guide." Orbital Science Corporation, Apr-2010.

[12] G. G. Zohar, "Ad-hoc regional coverage constellations of CubeSats using secondary launches," California Polytechnic State University, San Luis Obispo, 2013.

[13] C. Boshuizen, J. Mason, P. Klupar, and S. Spanhake, "Results from the Planet Labs Flock Constellation," presented at the AIAA/USU Small Satellite Conference, Logan, Utah, 2014.

[14] V Foreman, J Le Moigne, O De Weck, "Satellite Constellation Cost Modeling: An Aggregate Model", NASA Earth Science Technology Forum, Annapolis, MD, 2016.