SSC24-WV-09

A Data-Efficient Model-based Task Decomposition Approach for Massive Satellite Constellations

Benjamin Cooper, John Grimes, Michael Planer, Letitia Li, Daniel Wallach
BAE Systems, Inc.
800 District Ave., Burlington, MA 01803; (781)-402-4966
Benjamin.cooper@baesystems.us

Hooman Sherkat-Massoom, Alex Fox, Alessandro Monteros, Ryan Mohr, Maria Fonoberova
AIMdyn, Inc.

Lael Rudd
DARPA

ABSTRACT

The rapid development and launch of low-cost satellites have led to constellations of hundreds of satellites, the proliferated Low Earth Orbit (p-LEO) constellation, and has changed the dynamic of space-based earth observation. The availability of massive numbers of satellites creates the opportunity to tackle larger scale problems, such as larger scale monitoring of wildlife, illegal fishing, and climate events. These opportunities must be met with increasing automation, preferably using an approach that executes and adapts on-orbit.

Optimizing distributed resources remains an astronomically challenging problem, even more so when computations are performed on the comparatively compute restricted hardware of satellites. We propose a data-efficient two-part system, modeled after the Actor-Critic architecture, that models the dynamics of p-LEO constellations and other data streams and generates optimized tasking. First, using the Koopman operator theory approach, we model an aggregate representation of a heterogenous (multi-modal sensing) satellite constellation to predict satellite availability, observation capabilities, and resource utilization. An aggregate representation of the constellation enables scalability to model hundreds to thousands of satellites, as well as being agnostic to particular identities (satellites may enter and leave the constellation). Second, we use the Hierarchical Bayesian Program Learning (HBPL) paradigm to formulate and learn a task decomposition and generation 'program'. Tasks are constructed and defined probabilistically while guided by expert informed structure and bounds, enabling efficient search and optimization of the task space. During execution, the HBPL component proposes sets of tasks (serving as the 'actor') which are scored by the Koopman model. The two methods described above are notable due to their low data requirements and speed of model training or updating, making them a stellar pairing for on-orbit applications.

This data-driven learning approach to task generation was explored to solve the task decomposition problem for the BAE Systems Collective Space Tasking and Assimilation Reasoning System (CoSTARS) under the DARPA Oversight program. The full architecture includes a distributed auction mechanism for task assignment, a data assimilation component, and updates on entities or objectives. The task decomposition approach is evaluated under this system architecture for the case of monitoring a set number of entities of interest.

PROBLEM STATEMENT

Decreasing launch costs have led to rapid growth in space-based earth sensing satellites. These new proliferated Low Earth Orbit (p-LEO) constellations are rapidly expanding in scale and sensitivity while also diversifying to include broader, multi-modal sensing capabilities. Payload orchestration, control, analytics, and data dissemination on short, relevant timescales is stressing and potentially a large cost driver for the constellation owners. Traditional ground control systems are stressed as those large data volumes are pushed down across limited ground and intra-satellite links, include large ground server farm requirements, and inefficient satellite re-planning timescales. The increased heterogeneity of the space systems also creates significant optimization challenges for the ground control systems. Increased space-based automation offers a more scalable solution, leveraging onboard data assimilation and intra-satellite coordination to address these challenges and support rapid re-planning time scales.

These new p-LEO constellations largely focus on geospatial intelligence capabilities. Customers include business intelligence, investment funds, insurance agencies, and non-government organizations focused on a wide range of use cases. Maintaining custody of illegal fishing vessels, wildlife poachers, and other entities is a particularly interesting focus of many space-based earth-sensing systems. Detecting, monitoring, and identifying Illegal fishing and wildlife poaching stress p-LEO constellations as it requires rapid and adaptable constellation orchestration and data assimilation.

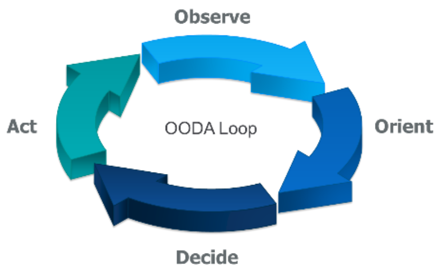
As part of DARPA's Oversight program, BAE Systems is creating the Collective Space Tasking and Assimilation Reasoning System (CoSTARS). CoSTARS leverages collective functions, originally designed for swarm autonomy, and modified for a proliferated LEO (p-LEO) environment to distribute tasking, sensing, signal processing, data assimilation, bandwidth optimization, and near real time analytics to maintain custody on illegal activities across regions of interest. CoSTARS creates a virtual cloud above a region of interest (e.g. protected fisheries, national park, …) pushing data and tasking to satellites as they flow through the sensing regions above. CoSTARS distributes the data and sensing by leveraging a collective auction function to create ownership of custody and function needs across constellations. This ownership changes as a function of upcoming sensing opportunities, bandwidth limits, computation resources, and power constraints. Lastly, CoSTARS contains Koopman AI models to learn and adapt for creation of tasking and behaviors matched to historical performance.

In this paper we focus on the problem of optimizing distributed resources within the comparatively compute restricted hardware of satellites. We propose a data-efficient two-part system, modeled after the Actor-Critic architecture, that models the dynamics of p-LEO constellations and other data streams and generates optimized tasking. First, using the Koopman operator theory approach, we model an aggregate representation of a heterogenous (multi-modal sensing) satellite constellation to predict satellite availability, observation capabilities, and resource utilization. Second, we use the Hierarchical Bayesian Program Learning (HBPL) paradigm to formulate and learn a task decomposition and generation 'program'. During execution, the HBPL component proposes sets of tasks (serving as the 'actor') which are scored by the Koopman model.

BACKGROUND AND CONTEXT

Illegal fishing activities are estimated to cost \$36.4 billion to the world economy every year according to an estimate of the world wildlife organization. Illegal fishing is difficult to detect and enforce due to the vastness of the maritime domain and limited protection resources. Space-based geospatial intelligence systems however offer the ability to monitor the maritime domain and enable fishery protection and enforcement of international rules. Space systems can monitor ships in a variety of modes including Electro-Optical (EO) imagery, Synthetic Aperture radar (SAR) imagery, and radio frequency (RF) emissions such as Automatic Identification System (AIS) transponders. Commercial companies addressing these sensing modalities include Spire, Capella Space, Umbra, IceEye, Planet, Maxar, and many others.

A common decision-making framework is the Observe, Orient, Decide, & Act (OODA) framework (Figure 1). Traditional timelines to have a satellite observe an activity, download the data to the ground station, analyze (orient) and identify the activity, decide how to respond, and then act (e.g. deploy to interdict an illegal fishing trawler) are days. That timeline is too slow and resource heavy for success at the necessary scale. This problem is even worse today with far more observation (satellite collects) across many different sensor types requiring data assimilation across the modalities and sending updates to the interdicting vessel. These capabilities all need to be pushed to autonomous and distributed systems to respond to the necessary scale of the problem.



Emerging and Evolving Low Earth Orbit Capabilities

Figure 1 Observe, Orient, Decide, and Act

The combination of reduced launch costs, advances in microelectronics and materials, and development of small satellites has led to a rise in interest for new applications of massive low earth orbit constellations[1], such as SpaceX's Starlink constellation serving connectivity to users through over 4000 satellites[2]. Communications, space Internet-of-Things, and Earth observation missions make up 96% of the proposed constellations[2]. There is a growing market for global space-based connectivity, as well as improved and more continuous monitoring of the Earth. This trend will place tens of thousands of satellites in orbit; however, this presents a new major challenge for efficient planning and coordination of all those resources.

Manual, Centralized Ground Operations to Distributed On-Orbit Autonomy

The remoteness of space, the reliability of communication, and the many hazards to human life have always made space operations a compelling target for remote and automated operation. In 1967, the Soviet Union performed the first automated docking in space. The first automated planning for space was the Spike scheduling system, in 1990, that used pointing, power, thermal, comms constraints to plan observations for the Hubble Space Telescope[3]. The Hubble planner was a ground-based system that calculated and uplinked its plans. In 2000, the Earth Observing-1 (EO-1) spacecraft was launched with onboard capabilities to perform mission planning and scientific analysis[4].

The rise in number of satellites as well as the increase in communications capabilities naturally opens the opportunities but also the challenges for constellation wide operations. Practical distributed satellite uses are well established through the GPS constellations for positioning, and communications constellations, such as IRIDIUM, for global telecom. However, the biggest new opportunity is for distributed earth observation, and the increased performance brought through multiple simultaneous observations across geographic locations, as well as increased revisit times[5]. Autonomy and AI can bring more optimal management of resources, reduced inactivity periods, responsive, flexibility and adaptability to distributed satellite systems[5]. However, pushing more of the Earth observation tasks towards on-board operations with a heterogeneous distributed constellation of satellites remains a challenge.

COSTARS ARCHITECTURE

Collective Space Tasking and Assimilation Reasoning System (CoSTARS) enables key technical and architectural innovations to meet Oversight program needs. CoSTARS utilizes collective objectives that are flexible and enable support to a wide range of space system implementations and capabilities. We focus on rapid adaptability and learning by CoSTARS to new spacecraft, sensors, sensor types, and contexts.

Functional View

The dynamical system complexity of both p-LEO space constellations and the custody mission highlight the importance of robust and adaptive system modeling and architectural design.

Figure 2 diagrams the activity and data flow through the custody maintenance process. CoSTARS automatically creates multi-option tasking based on entity quality, features, and expected behaviors. A distributed auction brokering system collectively works across CoSTARS and other space systems to dynamically assign and allocate based on the system's ability, availability, and cost to meet the necessary tasking. Once tasks are allocated, they are scheduled by the CoSTARS resource management system. Sensor data is then assimilated and assessed by our data assimilation and entity classification applications creating a fused picture. Custody needs are continuously re-assessed locally to examine needs for new tasking and resource allocations. Additionally, system and target behaviors are learned to automatically create optimal behaviors over the heterogeneous space systems. The spacecraft constraints on power, processing, and data management are fundamental to our design goals and implemented as part of the resource brokering.

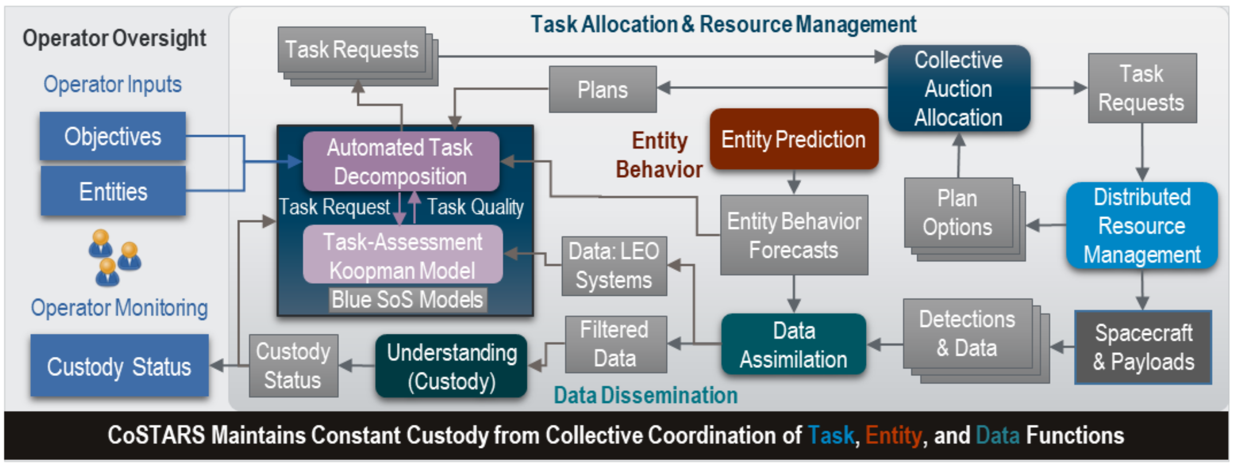


Figure 2 CoSTARS is designed to be adaptable to deployment needs and available resources, including on-orbit execution or a combination of ground and on-orbit deployment of different function services.

Collective Functions

As p-LEO constellation grow to 100s of satellites it is natural to examine scalable optimization and organization functions. Research in swarm theory has shown significant value in controlling heterogenous, distributed systems[6,7]. This work, based on observations of bees, ants, and other colony animals often enables complex and near-optimal behaviors based on locally evaluated collective functions. These individual behaviors are organized by locally determined collective functions that work across the swarm to produce the intended behaviors.

While the physics of space preclude significant active control to create persistent, high-density clustering, the concept of collective functions expands easily to p-LEO space coordination. Over any geographic region the individuals within the collective region are changing rapidly on timescales of a 1 - 10 minutes. We use the concept of collective to organize actions and activities over the local region. These activities are based on individual regional needs, e.g. monitoring for illegal fishing in a specific exclusion zone. Our analysis has shown that collective functions creating local assignments based on distance from area of interest, satellite resource constraints, and bandwidth linkages assigns resources in an efficient manner. These resources include functions such as auction broker, data assimilation engine, and other components in the overhead region. These results will be discussed further in a poster session at Small Satellite 2024[8].

Custody Maintenance

The CoSTARS system functionally implements an OODA loop to implement each operator objective. In this case, we focus on regional monitoring of illegal fishing. We define custody maintenance as the objective for this regional monitoring. We refer to custody in the sense that our system has sufficient capability to localize and identify an entity at any point in time. Where sufficient capability considers both our knowledge about the entity's behavior and the available sensing capabilities across different modalities to observe, localize, and identify the entity in question. Therefore, custody maintenance is the process by which we assess our current level of custody and act to maintain that custody in the future.

Leveraging collective function assignment and auction algorithms satellite sensor tasks are created and assigned. Collected data is routed to the appropriate data assimilation engine and filtered movement data and ID features are created. This filtered data is associated back to the mission need and a common intelligence picture is created. The intelligence picture is evaluated and again the system decides on whether to create new sensor tasking or alert for a governmental interdiction. Learning is applied both at task creation and the common intelligence picture application to improve overall system performance.

DATA-DRIVEN TASK DECOMPOSITION ARCHITECTURE

The task decomposition component of the overall CoSTARS architecture is a data-driven model to generate tasks that efficient satisfy the objective of maintaining custody over the entities of interest. More plainly, the task decomposition approach falls under the banner of machine learning methods that are trained based on data that are from or representative of the true system and environment of interest.

The problem setup of creating tasks, that when executed, influence the state of the 'information environment' and result in a delayed reward of entity custody updates maps well to a reinforcement learning (RL) interpretation.

Actor-Critic Motivation

Within the reinforcement learning approach there are three major paradigms, model-based, policy-based, and value-based learning[9]. Model-based RL seeks to learn a model or representation of the underlying environment or system and use that model to support the policy. In our setting, the underlying environment and systems to be modeled include the constellation state dynamics, which will use the Koopman modeling, serving as the critic, described later. Policy-based RL updates the policy directly, and for our problem is related to the program that generates tasks. Value-based RL focuses on estimating the value function, or the cumulative reward observed after taking the optimal action. Maintenance of custody provides the overall signal of value in our problem, and as an intermediate measure, the Task Likelihood described later provides a measure of value for a given entity and constellation state with the generated action, i.e., the state-action pair. One could consider pure value-based or policy-based learning approaches to solve this, there are several considerations that lead towards taking a two-model actor-critic approach.

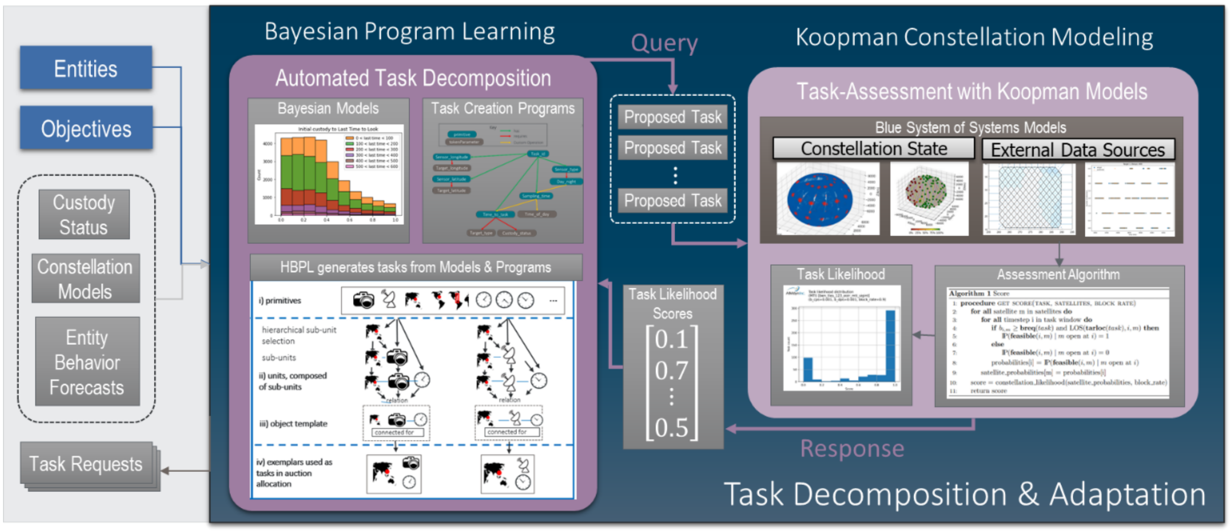
A recent paper by Sutton and Barto reviews the history and motivations behind the Actor-Critic Architecture and point to one powerful concept in actor-critic like algorithms, that they offer a good way to introduce prior knowledge into the learning process[10]. The architecture and other inductive biases of learning methods have been an instrumental tool in separating out expert understanding of a problem to what needs to be learned, a separation of concerns. Our problem has a natural separation of concerns in the following two ways. The action of generating a task is more dominantly concerned about aspects of the entity that determine when to observe, what type of sensor to select, and how to look (e.g. what angle to point). The critic performing the evaluation of the task, can model the more complex dynamics and features inherent in both the satellite constellation and environmental factors (e.g., weather)

The action of generating a task is a mix of discrete and continuous action spaces, already leading towards policy-based methods to support continuous spaces. The natural distributed nature of satellite constellations may benefit from or even require the use of the actor-critic architecture which is extensible to distributed and multi-agent formulations. Furthermore, as will be discussed in the next two sections, this separability of the actor and critic roles allows for different algorithmic approaches to be taken in each, particularly with respect to data efficiency. This departure, in the actor-critic architecture, from the standard neural network-based implementation is why we refer to is as more "actor-critic inspired."

Inputs and Outputs of each model

The Task Decomposition (actor) and Task-Assessment (critic) components operate in a query and response loop, as seen in the diagram of Figure 3. When the full Task Decomposition architecture receives an entity to monitor, the information about the entity is input to and processed by the Hierarchical Bayesian Program Learning (HBPL) component. The entity should include information about the type of entity (animal, ship, etc..), including any defining features such as length, color, etc.., the current estimated position, or other known state information. The entity monitoring request may be accompanied by additional objectives such as position accuracy desired. The HBPL component then may generate one or more Proposed Tasks to submit to the Task-Assessment component. The HBPL Task Decomposition component can propose a diversity of tasks to maintain supervision of the entity, such as different rates to revisit, or use of different sensing modalities.

The Koopman-based Task-Assessment component receives and processes the Proposed Tasks to return a Task Likelihood score for each submitted task. Based on the type of entity and parameters selected in the task (time to observe, sensor type, ...) the Task-Assessment component will select a combination of geometric and data-driven models to look forward in time and estimate the likelihood of task success. The calculation of Task likelihood is covered later, but considers predictions on constellation workload, power levels, and observability factors. The component then response to the Task Decomposition component's query with the corresponding task likelihoods.



Bayesian Program Learning
Query
Koopman Constellation Modeling
Entities
Objectives
Custody Status
Constellation Models
Entity Behavior Forecasts
Task Requests
Automated Task Decomposition
Bayesian Models
Task Creation Programs
HBPL generates tasks from Models & Programs
i) primitives
hierarchical sub-unit selection
sub-units
ii) units, composed of sub-units
relation
iii) object template
connected for
iv) exemplars used as tasks in auction allocation
Proposed Task
Proposed Task
Proposed Task
Task Likelihood Scores
0.1
0.7
0.5
Task-Assessment with Koopman Models
Blue System of Systems Models
Constellation State
External Data Sources
Task Likelihood
Assessment Algorithm
Response
Task Decomposition & Adaptation

Figure 3 The Task Decomposition module consists of two components, one to create proposed tasks, and another to score the tasks in a Query-Response process.

Overall Algorithm

The overall algorithm, as in Figure 4, is a straightforward loop over all the entities received. For each entity, the Task Decomposer will generate a set of proposed tasks. This could include one or more tasks and would be part of the configuration of the Task Decomposer. The model behind the Task Decomposer is probabilistic, therefore we can sample from the underlying task generation distribution. We could just as well take the maximum value over the task distribution, which in probabilistic decision-making terms would be a pure exploitation move. However, we would like to generate a diversity of tasks for a few reasons. Firstly, exploring the task space will improve learning for both the HPBP task decomposition models and the Koopman system of system constellation models. Secondly, we don't expect the HBPL model to capture all relevant factors in its task generating models. The Koopman task-assessment component captures and provides additional understanding, and the task which is mostly likely in the HBPL task distribution space may miss some insight that the Koopman model has captured.

After the task-assessment component has returned the task likelihood scores, the algorithm selects one or more of the tasks. The implementation of the `select` function can be as simple as taking the task with the max likelihood score from the assessment model. In more sophisticated approaches, it could select multiple tasks based on a weighting between the assessment score and factors in the HBPL model. The select function could also take some form of risk into consideration, such as the rates of false positives and negatives in the underlying models.

Finally, the set of selected tasks for this entity are submitted to the constellation for assignment and scheduling. The loop continues until all entities have tasks generated and submitted to satellites for assignment.

Algorithm 1: Algorithm Query and Response

```
Input: Entity (and attributes), Objective
Output: Set of Tasks
Components: TaskDecomposer, TaskAssessor
1:  for each entity:
2:      entity_tasks = []
3:      proposed_task_set = TaskDecomposer.generateTasks(entity, objective)
4:      task_scores = TaskAssessor.assess(proposed_task_set)
5:      entity_tasks = select(proposed_task_set, task_scores)
6:      AssignmentComponent.submit(entity_tasks)
7:  end
```

Figure 4 The overall Task Decomposition component algorithm is a Query and Response process for each entity in the entity list.

CONSTELLATION MODELLING WITH KOOPMAN OPERATOR

Operating an effective custody maintenance capability requires matching the observation needs of each entity with the available satellite resources. The increasing numbers of satellites and the push towards on-board decisions making challenges this distributed resource matching problem. The challenge motivates the approach to use data-driven modeling of constellation load dynamics. We make the following observations. First, due to the distributed and asynchronous nature of the auction allocation process, it is not possible to predict which individual satellite will be assigned a particular task. However, given a set of historical data, we can model the aggregate behavior and loading on the constellation under various conditions. Second, we can generate tasking that is more likely to be successful assigned and executed by the aggregate constellation, and we are agnostic to the identity of which satellite executes the task.

Furthermore, we can model various individual and aggregate quantities including sensor usage, battery state, compute load, and thermal state. We can model these quantities using a combination of geometric information and measured data. Geometric or analytical models might include the Satellite-Earth geometry based on line-of-sight, and solar exposure calculations to inform battery charging rates. More data-driven models may include weather data, battery performance data, and historical payload utilization rates can be used to model likelihoods of task success.

Finally, even when we don't have control over an observation system, but do have access to the data produced, we can model the type and rates of receiving information about types of entities or specific geographic regions. An example would be information about cargo and fishing vessels reported through the AIS system[11]. For the case of wildlife management, the Argos System company provides several sensors and solutions that could feed data on specific entities or could be used to infer information about a population of entities[12]. We can include expected external information to determine the need for generating a task, given the nature of the entity or region.

Koopman Operator

Koopman operator theory stems from dynamical systems theory as an alternative viewpoint to the classical, geometric, state space view[13,14]. Consider a (nonlinear) dynamical system $x(n+1) = S\big(x(n)\big)$, where x lives in a state space X and $S: X \rightarrow X$. The classical viewpoint would be to try to find special solutions such as fixed points and periodic order or limit cycles and determine their stability, i.e., the behavior of the nearby trajectories in state space.

Koopman operator theory takes a different approach. Instead of following trajectories in state space, one looks at the evolution of observables (functions) $g: X \rightarrow C$ that are defined on the state space[15,16]. Let F be a vector space of vector-valued functions, which we assume is closed under composition with S, i.e., $g \circ S$. The Koopman operator $K: F \rightarrow F$ is defined by this composition operation $Kg(x) = g \circ S(x)$.

Koopman Operator theory can capture the dominant nonlinear behavior of the underlying dynamical system and, in certain cases, perform better than traditional state space and linearization techniques. The price one must pay in exchange is the operator is now infinite-dimensional since it acts on an infinite dimensional function space. Figure 5 shows a schematic of this lifting process.

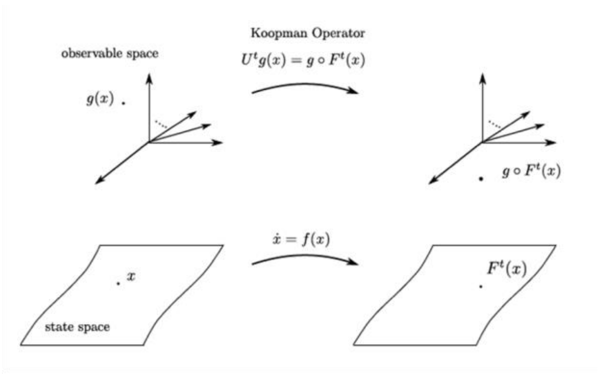


Figure 5 Koopman lifting schematic. A nonlinear ODE $\dot{x} = f(x)$ has a flow map $F^t(x)$. Instead of looking at state space trajectories, a vector-valued observable $g: X \rightarrow C^n$ is defined on the state space. The flow map induces a the Koopman semigroup of operators U^t which defines the evolution of the observable. This family of operators are linear albeit infinite-dimensional.

Since the Koopman operator is linear, it often has a spectral decomposition, allowing an observable to be decomposed into a linear combination of triplets --- eigenvalues, eigenfunctions, and Koopman modes, which are the projection coefficients for the vector-valued observable onto the eigenspace[16]. Additionally, there may exist a continuous part of the spectrum due to the operator being infinite dimensional. Even though the operator is infinite, we can get good approximations of the eigenvalues and modes using data-driven methods based on fast numerical linear algebra algorithms. This requires the judicious choice of observables to get good approximations.

Constellation Data and Modelling Process

We make use of an approximation of the Koopman Operator that derives a finite matrix to evolve the state of the system forward in time. The procedure to produce the finite Koopman matrix is typically some form of the Dynamic Mode Decomposition (DMD). In addition, the delay embeddings technique is used to transform the data to capture different types of behaviors observed in the constellation data.

While any of the constellation data recorded over time can be used in the constellation dynamics modeling, we will illustrate the technique using the state trajectories of the set of satellites in the constellation. As in

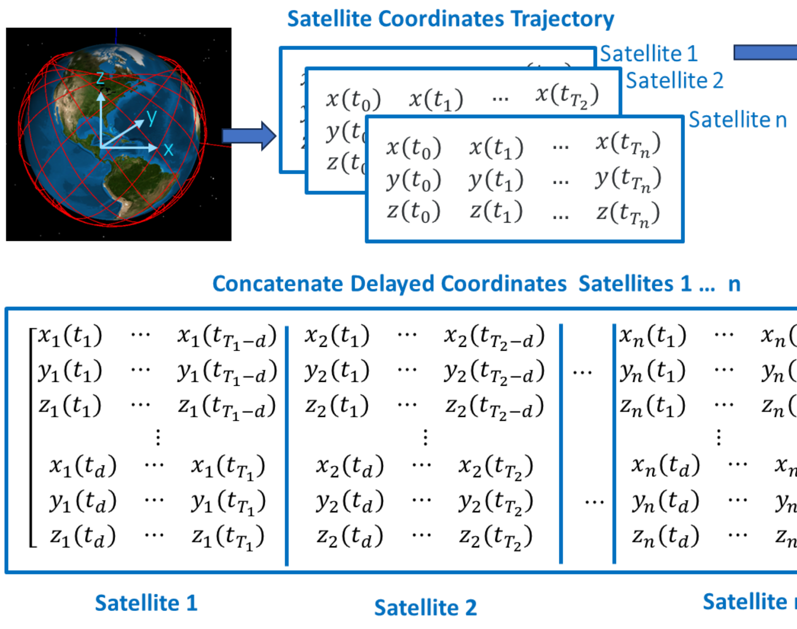

Satellite Coordinates Trajectory
Satellite 1
Satellite 2
Satellite n
$x(t_0)$ $x(t_1)$ … $x(t_{T_n})$
$y(t_0)$ $y(t_1)$ … $y(t_{T_n})$
$z(t_0)$ $z(t_1)$ … $z(t_{T_n})$
Concatenate Delayed Coordinates Satellites 1 … n
Satellite 1
Satellite 2

Figure 6, we take a data set of satellite trajectories recorded over a sufficient amount of time to capture the important dynamics. Each satellite coordinate history is transformed using the delay embeddings

approach to produce a Hankel matrix with d delays. The Hankel matrix is a square matrix in which each ascending skew-diagonal from left to right is a constant, and often appears in system identification approaches. Delay-coordinate approaches are often used in time-series prediction, going back to even the original Koopman paper[13,16]. Then, each satellite's delayed coordinates are concatenated together. This process of delaying and concatenating is visualized in the flow diagram in

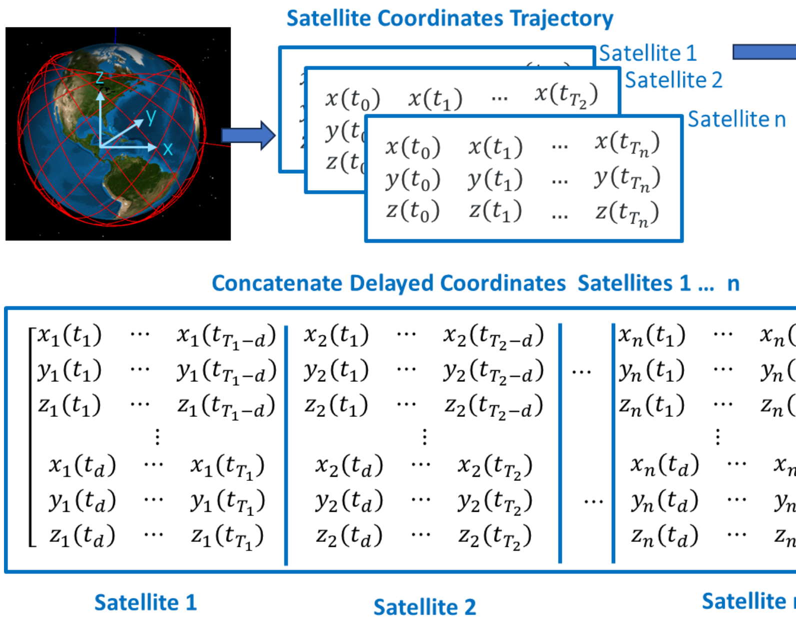

Satellite Coordinates Trajectory
Satellite 1
Satellite 2
Satellite n
Concatenate Delayed Coordinates Satellites 1 … n
Satellite 1
Satellite 2
Satellite

Figure 6.

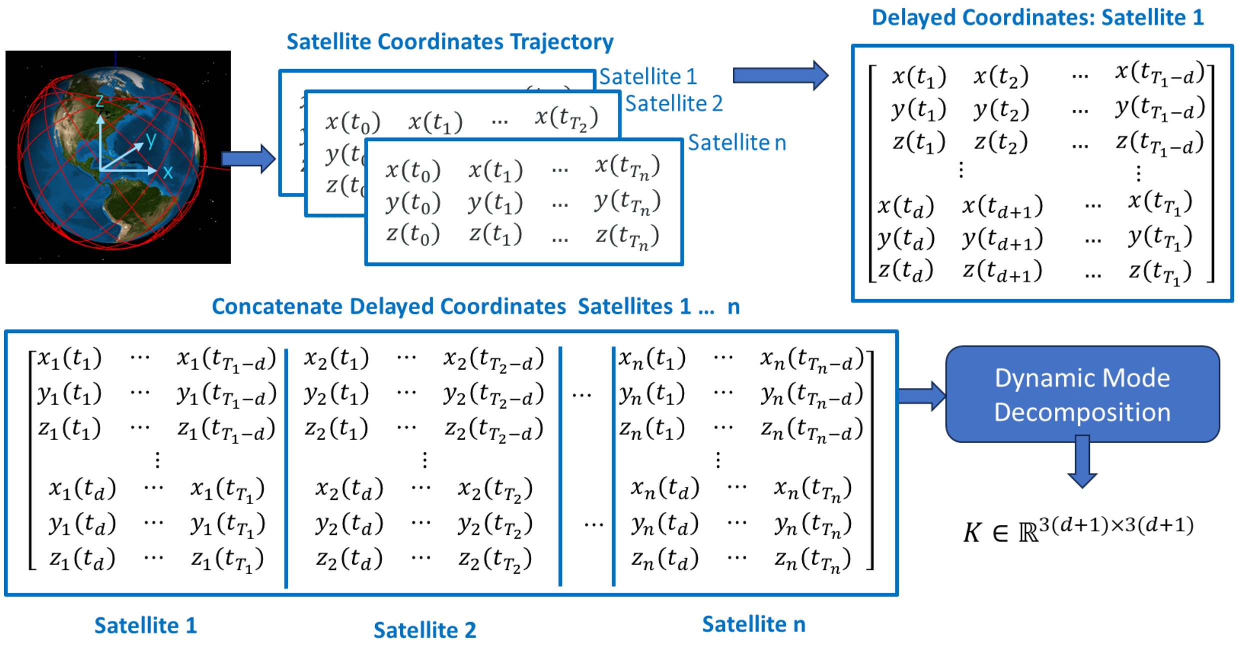

Satellite Coordinates Trajectory
Satellite 1
Satellite 2
Satellite n
Delayed Coordinates: Satellite 1
Concatenate Delayed Coordinates Satellites 1 … n
Dynamic Mode Decomposition
$K \in \mathbb{R}^{3(d+1)\times 3(d+1)}$
Satellite 1
Satellite 2
Satellite n

Figure 6 Starting with satellite trajectory data, the coordinate histories for each satellite are transformed according to the delay embedding approach for d delays. The delayed satellite coordinates are then concatenated together, and this serves as the base data for performing the DMD to produce the Koopman matrix.

Training of the Koopman matrix K is performed by DMD using the concatenated matrix described above.

Specifically, if $\left(x_i^k\right)_{i=1}^{m_k+1}$ is a time series of satellite motion for each satellite $k = 1, \ldots, n$, then let X be the matrix with columns

$$\left[x_1^1, \ldots, x_{m_1}^1, \ldots, x_1^n, \ldots, x_{m_n}^n\right]. \tag{1}$$

And let Y be the matrix with columns

$$\left[x_2^1, \ldots, x_{m_1+1}^1, \ldots, x_2^n, \ldots, x_{m_n+1}^n\right]. \tag{2}$$

Then, DMD is performed with X and Y to solve $KX \approx Y$ for the Koopman matrix K. The Koopman evolution matrix, K, captures the constellation dynamics (and other dynamics included in data) and allows for the prediction of any arbitrary satellite, therefore allowing the modeling and prediction of satellites coming in or dropping out of the constellation.

Lifting the satellite data to the aggregate density model

The previous section described modeling process to learn a matrix that evolves a satellite's state over time to predict availability and capability status for a satellite. Since the dynamics have been learned as a linear operator, we can generate these predictions quickly for a large number of satellites. A useful dual of this state-based prediction is to leverage the true insight of the Koopman operator and generate models based on so-called "observables" or transformations of the underlying state.

Working in the observable space can have a few advantages. Often an observed or measured description of a system is simpler or more useful to work with, this is especially true when working with large quantities of units. For instance, we could understand the thermal state of a system (e.g., the temperature) based on calculations of all the interactions of the atoms in a certain region, which would be a very intensive calculation. Or we could model the temperature of a system based with a few parameters that are based on observations or measurements of the system. Likewise, as the number of satellites increases, we may think less about which satellite has what capability at what time, and more about the dynamics of capability and availability in the aggregate sense.

We commonly refer to this process of transforming the data as 'lifting' and operating over Koopman models in the 'lifted space'. A useful lifting technique is to transform the data points into a 'density'. In this formulation, the concentration of a data type is captured as a region with higher density. For the case of satellite position, the 'availability density' describes the number of satellites within that sub-region over some interval of time, as visualized in Figure 7. These densities evolve over time when the Koopman matrix is applied to the density state of the constellation.

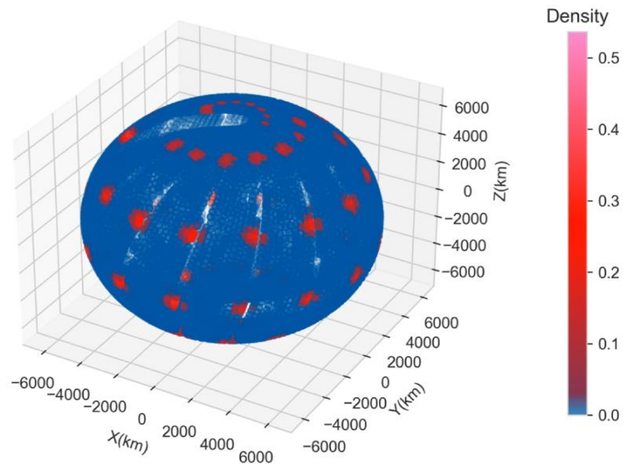


Figure 7 Visual representation of the constellation satellite density

EARTH OBSERVATION TASK DECOMPOSITION WITH HBPL

Maintaining custody of a revolving and growing entity list using an evolving set of heterogeneous resources requires a flexible task decomposition system. Hard-coded templates for task decomposition and hand-crafted models of satellite resources become stale and time consuming to maintain at scale especially given the evolving nature of p-LEO space systems. Our auto-generated tasks are distributed to our distributed planning and scheduling services which coordinate and agree on task assignment and resource utilization across heterogeneous assets. Agreement must be reached under dynamically evolving entity behaviors and requests on custody status. CoSTARS provides Task Decomposition to model and flexibly generate tasks.

Taking the reinforcement learning viewpoint of task decomposition, there is a limited action space for creating a task. For a given entity, the actions or decision points in the observation task include when to look (time of task, revisit rate), where to look (lat/long position), and how to look (sensor type, angle to the horizon, frequency band). While these actions could be explored randomly and independently, there are some dependencies between actions that can guide the decision-making process of creating a task.

Consider the example of monitoring wildlife which has been tagged with a transmitter. First, it may be necessary to consider the location and characteristics of the animal to determine when to look. If the animal is slow, has predictable behaviors and lives in an open environment like a grassy plain, it is not necessary to observe that often. Second, depending both on the location and characteristics of the animal and how soon you need to observe, different sensors may apply. A daytime observation can be performed with an electro-optical sensor, but nighttime observations would have to leverage the rf transmission or an available infrared sensor. Additional context like weather and terrain can factor into the prior decisions as well.

Following from the actions space dependency argument above, we make use of Hierarchical Bayesian Program Learning (HBPL) to model and generate tasks based on historical data and an expert informed template structure to guide decision points in the task.

Hierarchical Bayesian Program Learning (HBPL)

HBPL provides a framework for learning probabilistic programs that represent high-level concepts (types) and allows them to be constructed as a combination of an extensible set of parts (primitives) and their combined relationships in a hierarchical manner. These learned concepts or types serve as a template for the program to generate new instances based on a given input. As an example, HBPL has been demonstrated in modeling handwritten characters[17,18], where alphabet characters serve as types that are modeled. Each character is modeled using a set of pen-strokes (primitives) that are probabilistically combined as sub-units which form a program describing how a character is drawn. Programs then serve as generative models for creating new exemplars of hand-drawn characters.

In our basic task decomposition model, the primitives consist of the type of sensor, the location to point, and a time to perform the observation task. An example of the HBPL process for Earth observation tasking is shown in Figure 8. Following the though process from the wildlife example, the entity and current location helps determine the time to observe, which in turn can guide the choice of sensor type to use. This composition of sub-units (primitives) yields the task template ψ. This provides the general structure for a task generation program based on that template, $P(\psi)$. Then under a specific input Φ, which could include target location and characteristics, an exemplar task or instance of the task template is created from program $P(\mathrm{T}|\psi,\Phi)$. Variation in resulting exemplar tasks is captured through the Bayesian probabilistic framework of HBPL. This is the aspect of HBPL that allows for multiple tasks to be generated for a given input, to be further evaluated by the Koopman Task-Assessment component.

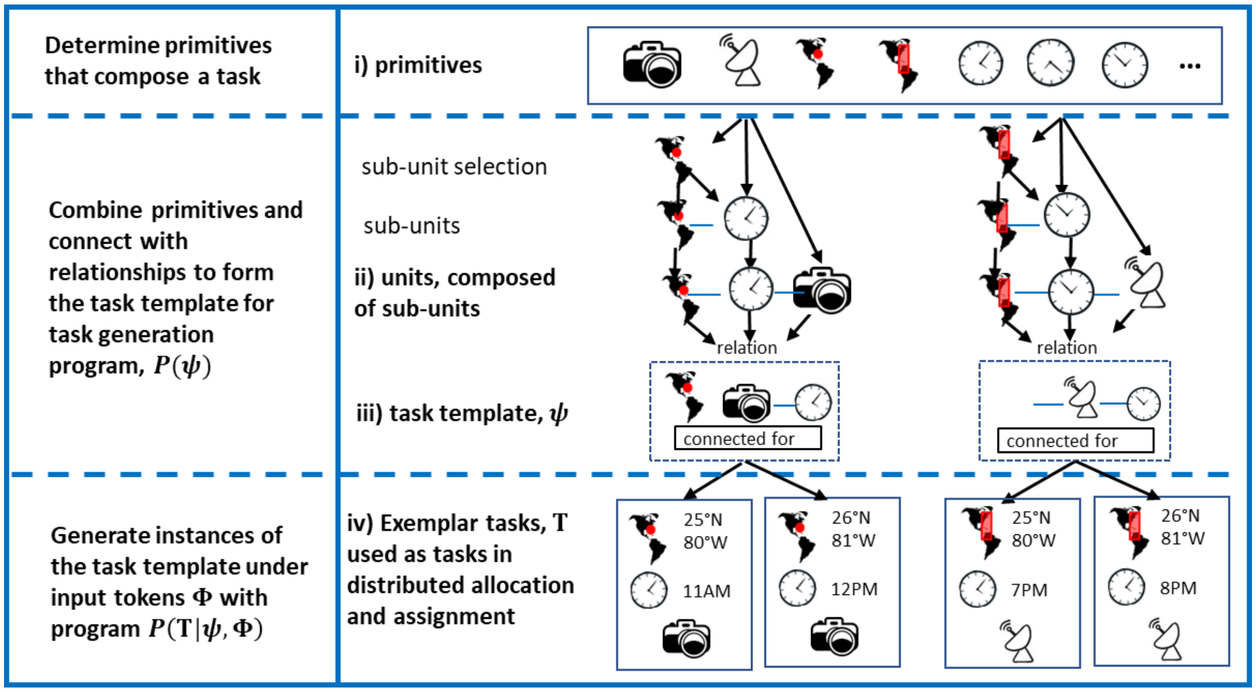
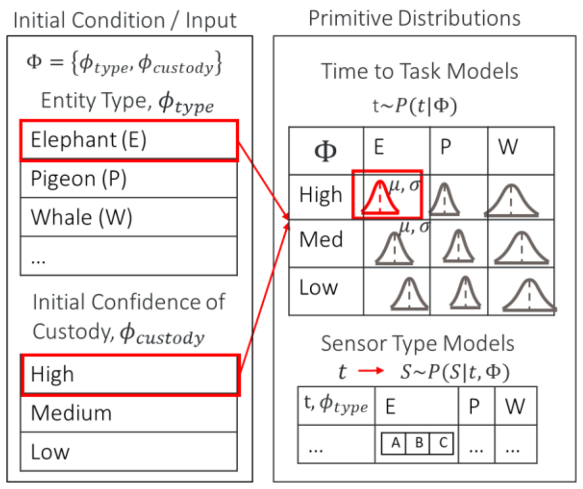

Determine primitives that compose a task
i) primitives
...
Combine primitives and connect with relationships to form the task template for task generation program, $P(\psi)$
sub-unit selection
sub-units
ii) units, composed of sub-units
relation
relation
iii) task template, ψ
connected for
connected for
Generate instances of the task template under input tokens Φ with program $P(\mathrm{T}|\psi,\Phi)$
iv) Exemplar tasks, T used as tasks in distributed allocation and assignment
25°N 80°W
11AM
26°N 81°W
12PM
25°N 80°W
7PM
26°N 81°W
8PM

Figure 8 HBPL template construction and instance generation procedure based on a set or primitives, relationships, and input tokens.

Bayesian Modeling for Task Timing

Each primitive in the HBPL program has one or more probability distributions that model the desired quantity (time to task, sensor type, …). An important part of the Bayesian modeling process is understanding the data generating process and determining and appropriate distribution to represent the primitive. Figure 9 illustrates one representation of program that selects the 'time to task' and the sensor type. Each part of the model may be conditioned on some initial input such as the entity type and some measure of custody confidence. Model components may also be conditioned on values generated from an intermediate step, such as the use of the time to task value in the sensor type selection model.



Initial Condition / Input
Primitive Distributions
$\Phi = \{\phi_{type}, \phi_{custody}\}$
Entity Type, ϕ_{type}
Elephant (E)
Pigeon (P)
Whale (W)
…
Initial Confidence of Custody, $\phi_{custody}$
High
Medium
Low
Time to Task Models
$t \sim P(t|\Phi)$
Φ
E
P
W
High
Med
Low
μ, σ
Sensor Type Models
$t \longrightarrow S \sim P(S|t, \Phi)$
t, ϕ_{type}
A
B
C
…

Figure 9 The inputs or initial conditions to the model might include the type of the entity and an initial confidence level of the custody. This directs the HBPL model to draw a prediction from the Time to Task distribution updated under those conditions.

Here, we focus on the modeling question of when to do the task. We are concerned with the efficient utilization of resources and suggest that there is a "just right" time to observe the entity when it's not too late to observe, but also not unnecessarily early. To illustrate the concept, consider the simple scenario, in Figure 10, in which an entity's position is known at T=0 with some level of confidence (red ellipse). The entity travels along at a constant velocity but may make some slight turns that cause the area that contains the predicted location to grow over time (expanding position covariance). We assume that if we capture the entity anywhere in our field of view (transparent blue square), then we are sure to detect it and maintain custody. A high frequency of observations will maintain custody, but since we are concerned with the efficient utilization of resources, we prefer to wait longer and therefore the value of that observation increases monotonically over time, until the point in which the entity could be outside the field of view and value sharply drops.

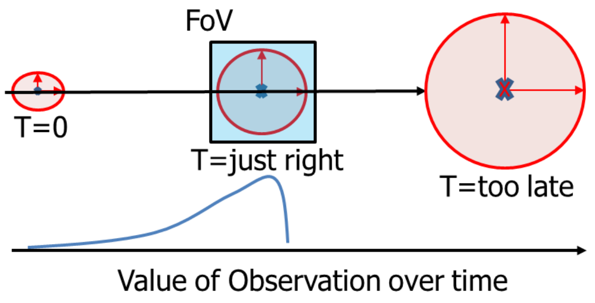


Figure 10 Consider that the value of an observation increases in time until the potential location of that entity leaves the extent of the field of View.

A baseline model can be obtained with the following data generating approach. We consider a constant velocity model which undergoes a random direction change at discrete intervals of time. Specifically, we describe the two-dimensional motion of an entity as:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + V \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \qquad (3)$$

Where the entity makes a random turn or change in its velocity vector according to angle θ which is a random variable distributed according to $\theta \sim 2\pi U(0,1)$, where $U(0,1)$ is a uniform random value between 0 and 1.

The location prediction model assumes that the entity maintains the same constant velocity vector, or equivalently uses a fixed angle θ_p to determine future position. For simplicity, the field of view has a circular ground footprint with a radius of 10 miles. The center of the field of view is pointed at the predicted location of the entity. Custody is maintained if the true location of the entity is anywhere within the circular field of view.

TASK LIKELIHOOD CALCULATION

As described in the earlier constellation modeling section, the Koopman-based Task-Assessment component receives and processes the Proposed Tasks to return a Task Likelihood score for each submitted task. This task likelihood score is intended to be a flexible measure on how valuable a specific task configuration is predicted to be. The current task likelihood score evaluates the feasibility of a task configuration. As the CoSTARS system develops and different types of data are available to process and model, the type of measure this score describes can expand. The earliest use for this task scoring is to evaluate the likelihood that this task will be successfully assigned through the distributed task allocation and resource management. Some considerations for this stage of the modeling includes prediction on resource availability to meet task requirements, including if a sensor is available or if the power levels are sufficient for the task. As modeling develops and data availability increases, this calculation can capture more measures such as, likelihood of successful detection given a successful scheduling, and likelihood this task contributes to custody given successful assignment and detection.

We approach the problem of formulating the task likelihood by considering the combination of two classes of models, analytic/geometric and data-driven. The analytic or geometric parts of the modeling can include basic line of sight calculations, analytical models of battery draw and solar charging (with respect to the Earth-Sun-satellite geometry). The data-driven modeling pieces include those harder to formulate aspects like the resulting task load across satellites after the distributed allocation, weather modeling, or more complex battery or thermal dynamics models.

In this work, we focus on the task likelihood concept of successful assignment of a task to a satellite, or a task feasibility likelihood. The key concept in this calculation is that of a 'blocking rate' in which we characterize the available time slots for a task on any given satellite to be blocked or available and can be thought of as the outcome of a weighted coin flip.

Consider a task T with a pointing location $\mathbf{loc}(T)$, and a resource requirement $\mathbf{req}(T)$. Let I be the number of timesteps within the provided window and let M be the number of satellites of the correct sensor type for T. Let $\mathbf{feasible}(i, m)$ be the event in which the mth satellite can accomplish task T at timestep i. For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$.

We consider that satellites are independent, then

$$P(\mathbf{feasible}(i,m) \text{ for some } i \in [I],\, m \in [M]) = 1 - \prod_{m\in[M]} (1 - P(\mathbf{feasible}(i,m) \text{ for some } i \in [I])). \tag{4}$$

Fix a satellite $m \in [M]$, and for a subset $J \subseteq [I]$, let $\mathbf{open}_J$ be the event in which exactly the timesteps of J are open (not blocked) for satellite m. Then, by the law of total probability, this allows us to separate out the expected availability (open slots) of satellites, and the capability (feasibility) present, as

$$P(\mathbf{feasible}(i,m) \text{ for some } i \in [I]) = \sum_{J\subseteq[I]} P\left(\mathbf{open}_J\right) P(\mathbf{feasible}(i,m) \text{ for some } i \in [I] \mid \mathbf{open}_J) \tag{5}$$

We model the availability or blocking rate of a satellite by independent Bernoulli trials across timesteps. Let $r \in [0,1]$ be the blocking rate, so $P\left(\mathbf{open}_J\right) = r^{I-|J|}(1-r)^{|J|}$. To determine $P\left(\mathbf{feasible}(i,m) \text{ for some } i \in [I] \mid \mathbf{open}_J\right)$, observe that

$$P\left(\mathbf{feasible}(i,m) \text{ for some } i \in [I] \mid \mathbf{open}_J\right) \geq \max_{i\in J} P(\mathbf{feasible}(i,m) \mid m \text{ open at } i). \tag{6}$$

Since once the obstacle of blocking is removed, the feasibility likelihood over all timesteps is at least the likelihood at any individual timestep. We make a conservative assumption that the above inequality is an equality to state that

$$P\left(\mathbf{feasible}(i,m) \text{ for some } i \in [I] \mid \mathbf{open}_J\right) = \max_{i\in J} P(\mathbf{feasible}(i,m) \mid m \text{ open at } i). \tag{7}$$

While conservative, for the purposes of using the task likelihood score for decision making, this assumption says that if the task is likely to fit in at least one time slot, that is sufficient.

Next, we describe a particular implementation of $P(\mathbf{feasible}(i,m) \mid m \text{ open at } i)$. Let $r_{i,m}$ be the resource available for satellite m at timestep i. Let $\mathrm{LOS}(\mathbf{loc}(T),\, i,\, m)$ be the event in which satellite m in in line of sight of $\mathbf{loc(T)}$ at timestep i. We make the assumption that for a given satellite and timestep, the resource availability and satellite position are independent. Then we can state a particular feasibility formulation,

$$P(\mathbf{feasible}(i,m) \mid m \text{ open at } i) = P\left(r_{i,m} \geq \mathbf{req}(T)\right) P(\mathrm{LOS}(\mathbf{loc}(T),\, i,\, m)). \tag{8}$$

Both parts of the expression on the right side can be replaced or modeled using data-driven methods, particularly when we consider complex resource dynamics and additional factors online of sight including weather, occlusions, or other forms of interference.

INITIAL FINDINGS

This paper has proposed a new learning-based actor-critic architecture to solve the problem of creating tasks for a distributed satellite system to maintain custody over many targets. We have developed each piece, HBPL task decomposition and Koopman constellation modeling and task assessment, in parallel and the coordination of both components in the overall actor-critic architecture and algorithm is under active development. We present the initial findings for this new approach.

Koopman Constellation Modeling

An initial focus of the constellation state modeling was capturing the propagation of the satellite position in both the state formulation and the lifted density formulation, as seen in the constellation modeling section. This modeling served as the foundation for layering in other types of state data, including the availability of resources. The task likelihood development has focused on the availability of a satellite to perform a task based on an estimated blocking rate. We can also include the blocking rate as an estimated value derived from historical missions and satellite activity. Figure 11 includes two graphs illustrating the blocking rates based on simulated data in which satellites were tasked to observe a set of 20 entities. These evaluations of the blocking rate are based on a task window of 10 minutes, in which at the current time, t_now, we consider different future tasking times as a "look ahead." Each color in the graph corresponds to a different satellite, and the graphs compare the blocking rate for a look ahead of 6 and 30 minutes.

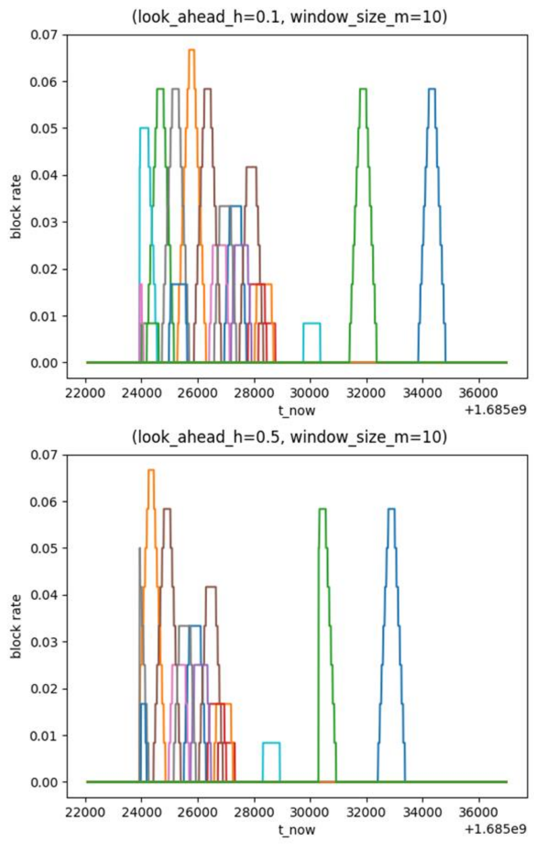


Figure 11 When evaluating a task to be done in the future, given the current time t_now, a specific satellite, a desired task window size, we can predict the blocking rate for a look ahead time into the future.

HBPL Task Decomposition Bayesian Modeling

We simulate 100,000 trajectories of the entity movement pattern described above. For each trajectory we choose a random time to look and predict the future location based on our constant velocity model, and check if the entity is within the field of view. Figure 12 shows the outcome of all the tasks that were successful, e.g., where custody was maintained. The position of each colored circle represents the true location of the entity when we choose to look, and the color is the time of observation. The density (and color) of the circles indicates that observing earlier and therefore closer to the known starting position has a higher frequency of maintaining custody.

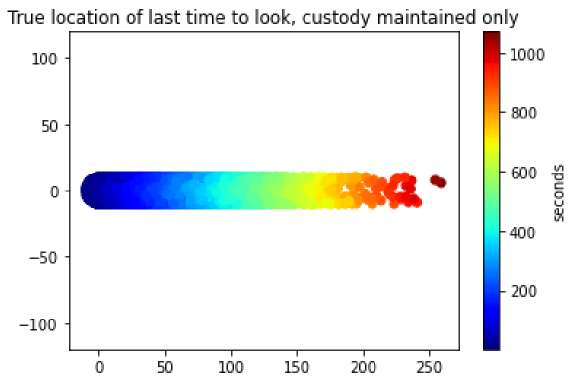


Figure 12 Density of circles indicate how frequent custody was maintained. Color of circles indicates how far into the future was the last successful time you could look.

The circle color and position distribution follow from the intuition that observing earlier is more likely to maintain custody. However, given that we are interested in the success rate of observing later, we are interested in the distribution of successful observations if taken later in time. Additionally, we recognize that successful maintenance of custody is influenced by the starting custody state. Figure 13 captures a few trends in the modeling of the optimal 'last time to look'. The data in the histogram represent all the successful instances of maintaining custody using a task assigned to observe zero to 600 minutes into the future. Each instances has a random initial custody error between 0.0 and 1.0, in which 0 represents no or very low uncertainty of custody, and 1 represents complete uncertainty about entity custody. Focusing on the first histogram bucket near the value 0.0 on the x axis, we can see a high count for a last time to look between 0 and 100. As the last time of successful tasks increases, we see that a doubling of time to the 100 to 200 bucket decreases success rates by about 20%, a tripling reduces success rate by about 60%, and so on. If we look across the x-axis and focus on a single 'last time' range, we see that the influence of initial custody is negligible until about 0.5 custody status.

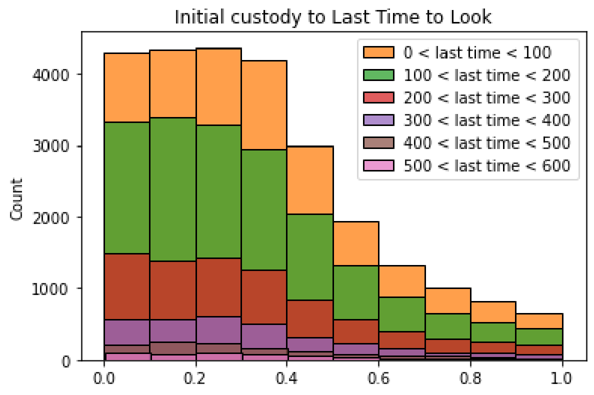


Figure 13 A initial custody error of 0 indicates high confidence, whereas custody error of 1 is complete loss of custody. The data generating experiments reveal the frequency at which custody is maintained for later observation times.

Task Likelihood

We evaluate some initial task likelihood calculations by looking at the effect of blocking rates on the scoring of tasks. The setup used in producing Figure 14 consists of a set of 30 satellites in a low earth orbit with wide fields of view. The tasks under consideration are all in one geographic area, and the timing of the tasks are randomly sampled across 90 minutes. In the left plot in Figure 14, we can see that under a low blocking rate, we predict nearly all potential tasks have a high likelihood score. The small set of zero likelihood occurs for tasks whose timing is not aligned with any satellites having coverage at that time. As seen in the right plot of Figure 14, with a higher blocking rate, the likelihood of our task distribution shifts downward.

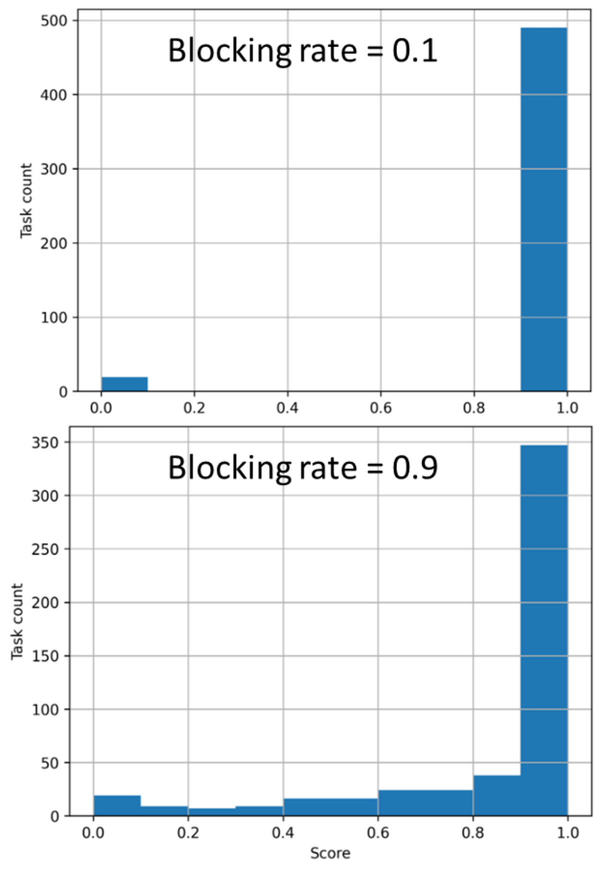


Figure 14 Blocking rate shifts the task likelihood score downwards.

CONCLUSIONS AND FUTURE WORK

We presented the problem of generating and distributing tasks to a large heterogeneous constellation of satellites to support the custody maintenance of a set of entities. The overall CoSTARS system architecture addresses the full pipeline of task decomposition and allocation across a distributed satellite system, and the dissemination and consolidation and filtering of observation data to support custody updates. This paper focused on a flexible and adaptable automated task decomposition architecture that drew inspiration from the successful reinforcement learning actor-critic architecture. We presented the development of each sub-component for Task Decomposition. Hierarchical Bayesian Program Learning was used in the task generation, which serves as the actor. In parallel, we developed the Koopman-based constellation modeling component that evaluates proposed tasks and gives a task likelihood score, serving as the critic. Each of the techniques has lower data volume requirements and faster training times than other state-of-the-art approaches such as neural networks, supporting data-efficient and on-orbit execution.

We presented initial findings on each of the elements of the learning architecture including Koopman constellation and resources modeling, Bayesian task timing modeling, and the resulting task likelihood scores based on a different levels of blocking rates. Future work includes expanding the set of primitives and Bayesian modeling for the HBPL component. We plan to capture more constellation dynamics in Koopman models across different resources and interactions. We will extend the task likelihood calculation to include custody likelihood as more metrics and data are available. Finally, we will integrate all the components of this architecture in a feedback loop and evaluate within the modeling and simulation environment we have developed to characterize the overall learning dynamics.

Acknowledgments

This material is based on work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. N6523623C8006. We are grateful to Dr. Lael Rudd and his team for their support.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the Department of Defense. This paper was approved under DISTAR case 39851 with Distribution Statement A – approved for public release, distribution unlimited.

References

1. Lappas V, Kostopoulos V., "A Survey on Small Satellite Technologies and Space Missions for Geodetic Applications," *Satellites Missions and Technologies for Geosciences*, 2020.

2. Curzi, G., D. Modenini and P. Tortora., "Large Constellations of Small Satellites: A Survey of Near Future Challenges and Missions," *Aerospace*, 2020.

3. M.D. Johnston, G. Miller., "Artificial intelligence scheduling for NASA's Hubble Space Telescope," in *The Fifth Annual AI Systems in Government Conference.*, 1990.

4. Page, Christine, Kerri Cahoy, and Evana Gizzi. , "Developing Intelligent Space Systems: A Survey & Rubric for Future Missions," in *Small Satellite Conference*, 2023.

5. Araguz C, Bou-Balust E, Alarcón E. , "Applying autonomy to distributed satellite systems: Trends, challenges, and future prospects," *Systems Engineering*, 2018.

6. Trianni, V., Campo, A., Fundamental Collective Behaviors in Swarm Robotics, Springer Handbook of Computational Intelligence, 2015.

7. Grimes, J., Cooper, B., Wallach, D., Trichos, M., Rudd, L., "Collective Space Tasking and Assimilation Reasoning System (CoSTARS)," in *Poster presented at Small Satellite Conference 2024*.

8. Grimes, J., Cooper, B., Wallach, D., Trichos, M., Rudd, L., "Collective Space Tasking and Assimilation Reasoning System (CoSTARS)," in *Poster presented at Small Satellite Conference 2024*.

9. Feng G, Zhong H., "Rethinking Model-based, Policy-based, and Value-based Reinforcement Learning via the Lens of Representation Complexity," *arXiv preprint,* 2023.

10. Barto AG, Sutton RS, Anderson CW., "Looking back on the actor–critic architecture," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* 2020.

11. "AIS Access," 24 5 2024. [Online]. Available: https://www.marinecadastre.gov/accessais/.

12. 24 5 2024. [Online]. Available: https://www.argos-system.org/products/airtime-connectivity/.

13. Koopman, B. O., & Neumann, J. V. , "Dynamical systems of continuous spectra," in *Proceedings of the National Academy of Sciences of the United States of America*, 1932.

14. Koopman, B. O. , "Hamiltonian systems and transformation in Hilbert space," in *Proc of the Nat'l Acad of Sciences of the United States of America*, 1931.

15. Mezic, I. , "Spectral properties of dynamical systems, model reduction and decompositions," *Nonlinear Dynamics*, pp. 41(1-3), 309-325, 2005.

16. Arbabi, H., & Mezic, I. , "Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator," *SIAM Journal on Applied Dynamical Systems,* pp. 2096-212, 2017.

17. Lake, B. M., Towards more human-like concept learning in machines: Compositionality, causality, and learning-to-learn, Ph.D. dissertation, Massachusetts Institute of Technology, 2014.

18. B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum., "Human-level concept learning through probabilistic program induction," *Science*, pp. vol. 350, pp. 1332–1338, 2015.