Planning, Scheduling, and Execution on the Moon: The CADRE Technology Demonstration Mission

Gregg Rabideau*
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
gregg.rabideau@jpl.nasa.gov

Joseph Russino*
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
joseph.a.russino@jpl.nasa.gov

Andrew Branch
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
andrew.branch@jpl.nasa.gov

Nihal Dhamani
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
nihal.n.dhamani@jpl.nasa.gov

Tiago Stegun Vaquero
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
tiago.stegun.vaquero@jpl.nasa.gov

Steve Chien
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
steve.a.chien@jpl.nasa.gov

Jean-Pierre de la Croix
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
jean-pierre.de.la.croix@jpl.nasa.gov

Federico Rossi
Jet Propulsion Laboratory - California Institute of Technology
Pasadena, CA, USA
federico.rossi@jpl.nasa.gov

ABSTRACT

NASA's Cooperative Autonomous Distributed Robotic Exploration (CADRE) mission, slated for flight to the Moon's Reiner Gamma region in 2025/2026, is designed to demonstrate multi-agent autonomous exploration of the Lunar surface and sub-surface. A team of three robots and a base station will autonomously explore a region near the lander, collecting the data required for 3D reconstruction of the surface with no human input; and then autonomously perform distributed sensing with multi-static ground penetrating radars (GPR), driving in formation while performing coordinated radar soundings to create a map of the subsurface. At the core of CADRE's software architecture is a novel autonomous, distributed planning, scheduling, and execution (PS&E) system. The system coordinates the robots' activities, planning and executing tasks that require multiple robots' participation while ensuring that each individual robot's thermal and power resources stay within prescribed bounds, and respecting ground-prescribed sleep-wake cycles. The system uses a centralized-planning, distributed-execution paradigm, and a leader election mechanism ensures robustness to failures of individual agents. In this paper, we describe the architecture of CADRE's PS&E system; discuss its design rationale; and report on verification and validation (V&V) testing of the system on CADRE's hardware in preparation for deployment on the Moon.

KEYWORDS

Lunar exploration, Space robotics, Leader election, Planning architecture, Space exploration

* denotes equal contribution.

This work is licensed under a Creative Commons Attribution International 4.0 License.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

ACM Reference Format:
Gregg Rabideau*, Joseph Russino*, Andrew Branch, Nihal Dhamani, Tiago Stegun Vaquero, Steve Chien, Jean-Pierre de la Croix, and Federico Rossi. 2025. Planning, Scheduling, and Execution on the Moon: The CADRE Technology Demonstration Mission. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Distributed instruments hold great promise to unlock key questions in planetary science that are inaccessible to single-point measurements. The ability to collect time-synchronized and cross-calibrated measurements from spatially distributed locations is critical to investigations of subjects as diverse as atmospheric circulation on Mars, Venus, and Titan [14]; sub-surface compositions of rocky and icy moons [18, 28]; and seismic activity on Venus [23].

The traditional operations paradigm for space exploration missions relies on on-board execution of sequences, developed by human operators on the ground, that specify which activities should be performed at which time by the spacecraft or rover [25]; however, such an approach generally does not scale to multi-agent systems, which must cope with bandwidth and latency constraints between individual agents; heterogeneous availability of resources (e.g., thermal and power conditions that may vary among agents); and potentially time-varying participation, as some agents may become temporarily, or permanently, unavailable.

These considerations motivate the development of multi-agent autonomous planning, scheduling, and execution (PS&E) tools for future space exploration missions. Many approaches for multi-agent PS&E have been proposed in the literature; however, to date, no approach has been developed and tested to a level of technology readiness [16] sufficient for infusion in future spaceflight missions.

NASA's Cooperative Autonomous Distributed Robotic Exploration (CADRE) mission, shown in Figure 1, aims to fill this gap.

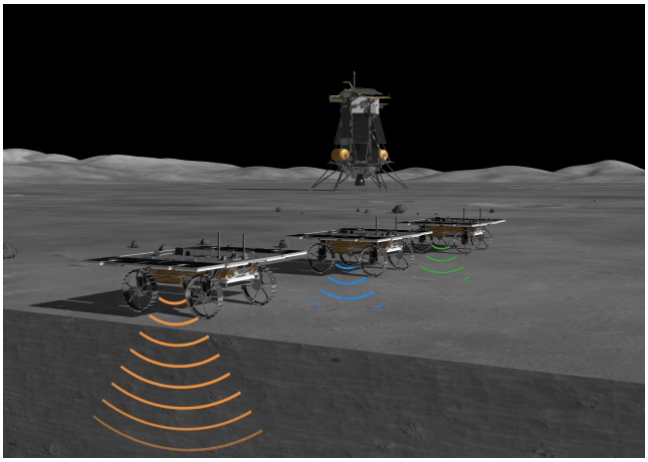

Figure 1: Overview of the CADRE mission.

CADRE will fly a team of four agents, including three mobile rovers and a static base station, to the Moon's Reiner Gamma region in 2025/2026. The goal of CADRE is to demonstrate the high technology readiness of cooperative autonomy, that is, the ability for a team of autonomous agents to elaborate high-level commands from ground (e.g., "explore this region") into commands for the rovers' mobility system and on-board instruments. Specifically, the rovers will cooperatively explore a prescribed region and collect multi-static ground penetrating radar measurements; the base station will act as a communication gateway between the rovers and Earth, and aid the team by performing computational tasks.

CADRE's on-board autonomy encompasses autonomous planning, scheduling, and execution; multi-agent motion-planning; frontier exploration; and single-agent mobility, including localization, mapping, and single-agent motion planning. In this paper, we focus on the autonomous PS&E architecture, i.e., on the module that elaborates high-level commands into tasks executed by individual vehicles. We refer the interested reader to [8] for a discussion of CADRE's overall autonomy architecture.

1.1 State of the Art

Multi-agent planning, scheduling, and execution. A vast number of approaches have been proposed for multi-agent planning, scheduling, and execution in the academic community. Coordination approaches include *centralized planning*, where the multi-agent system is effectively treated as a single agent; *leader-election*-based approaches, in which an agent is elected as leader and performs centralized planning for the entire team; *explicit-coordination* approaches (including auctions and broadcast-decentralized algorithms), where agents either bid on tasks to be completed based on their state and resources (e.g., [7]), or explicitly broadcast contention information (e.g., [19, 33]); *shared-world* (or *implicit coordination*) approaches, in which each agent maintains a world model through agent-to-agent information sharing, performs planning for the entire system based on the world model, and executes its part of the plan (e.g., [1, 31]); and *emerging behavior* approaches, in which agents communicate with their immediate neighbors and select the next task based on simple heuristics (e.g., [30]).

We refer the reader to [22] for a survey of algorithmic approaches to multi-agent decision-making, including planning and scheduling.

Planning and scheduling in space. The majority of the approaches outlined above approaches have not been demonstrated at a technology readiness level (TRL) sufficient for infusion in future missions.

A notable exception is the Distributed Spacecraft Autonomy (DSA) mission [1, 2, 5] on NASA's Starling mission. DSA has demonstrated shared-world planning of scientific observations in Earth orbit, where all agents build a common world model, and the planning problem is solved as an integer linear program.

However, a key difference makes DSA's approach not immediately applicable to CADRE's concept of operations. DSA operates in low Earth orbit, where performance of wireless inter-satellite communication links is comparatively predictable; this makes a shared-world approach appropriate, since it is highly likely that, with proper design of the telecommunication system, all agents will converge to the same world model. In contrast, CADRE's domain of operations is the Lunar surface, where surface obstructions and the behavior of disturbed regolith may strongly affect inter-agent communications; this makes a shared-world approach less desirable, since communication failures can result in complex and hard-to-diagnose miscoordination between agents.

Several approaches to *single-agent* planning and scheduling have been demonstrated in space. A notable example is the Perseverance Mars rover's Onboard Planner [29], in operational use since 2023. The planner holds promise to significantly increase the rover's mission productivity compared to current ground-in-the-loop approaches (which see the rover idle for up to 28% of the time [9, 10]), and also reduce rover energy usage by over 10% for a given campaign, saving resources for more opportunistic science investigations. A second notable example is CASPER, flown on NASA's EO-1 spacecraft [6]. On-board event detection from hyperspectral imagery was used to detect thermal anomalies (e.g., lava flows), clouds, floods, and other features; CASPER then replanned observations in response to these detections. A key commonality between all of these approaches is the strong decoupling between information-sharing and planning — that is, the assumption that either the planner has access to full system information (in Onboard Planner and CASPER), or that every planner has access to the same information (as in DSA). This assumption is natural for single-agent systems and reasonable for orbiting distributed systems; however, it is significantly more restrictive for surface systems.

1.2 Contribution

Our contribution is the development and testing of a leader-election-based multi-agent planning, scheduling, and execution architecture designed for operations on planetary surfaces. The proposed approach explicitly reasons about the information required for planning and for execution, and does *not* assume that all agents have access to the same information; careful, coupled design of the planning architecture and the task definitions results in reduced inter-agent communications compared to shared-world and auction-based approaches, ensuring good performance as well as resilience to disrupted inter-agent communications.

The rest of this paper is structured as follows. In Section 2, we describe CADRE's PS&E architecture. Section 3 describes how the

proposed task definitions implement CADRE's concept of operations, and discusses how tasks are designed to use local information as much as practical to reduce communication. Section 4 discusses the testing, verification, and validation of the proposed architecture and task definitions. Finally, we draw our conclusions in Section 5.

2 PLANNING, SCHEDULING, AND EXECUTION ARCHITECTURE

CADRE's PS&E architecture, shown in Figure 2, relies on four sets of modules:

- a *leader election* module that selects one agent to perform planning for the entire team;
- a lightweight *shared state database* that synchronizes selected state information from all agents to the elected leader;
- a *strategic planning* module that plans and schedules tasks for all agents; and
- a set of *agent controllers*, one per agent, that execute the scheduled tasks; monitor task progress and constraint satisfaction; and inform the strategic planning module of task success and failure and of constraint violations that affect other agents' tasks.

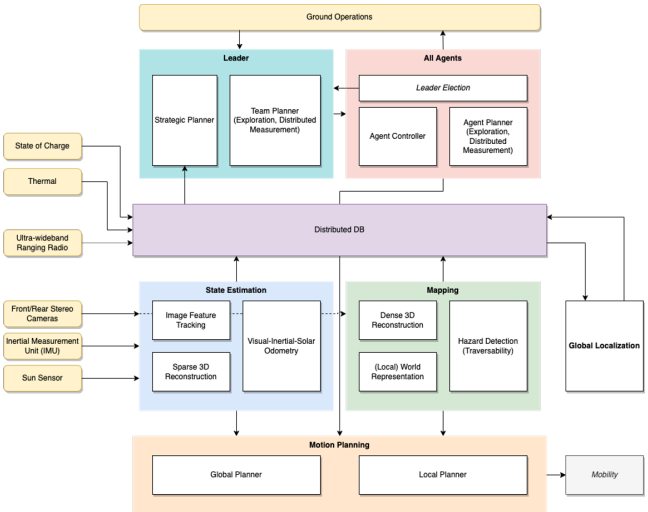

Figure 2: Architecture of CADRE's flight software. The second and third row of blocks from the top represent the PS&E architecture.

2.1 Leader Election

A leader election module ensures that every agent in the team agrees on the designation of a single agent as leader at any given time. A distributed minimum spanning tree algorithm, GHS [11], is used to select a unique appointer among the agents every ten seconds; the appointer then selects as leader the agent whose state is farthest from violating thermal and power constraints, and broadcasts the leader designation to all agents. A hysteresis mechanism is used to ensure that the leader is not changed unless large changes in its state are observed. The leader election module also periodically chooses a "designated survivor", an agent tasked with holding a copy of the leader's information, making recovery from loss of a leader faster. The architecture ensures that loss of the leader results in at most ten seconds of idle time before planning recommences on a new leader. We refer the reader to [3] for a detailed description of the leader election approach and implementation.

2.2 Shared State Database

A shared state database (SSDB) allows agents to synchronize key state information periodically. The shared state database is based on SQLite, and synchronization is implemented through a custom application layer. The implementation deliberately does not guarantee consistency between the agents in the sense of the CAP theorem [12] (i.e., different agents' databases may contain different information at any given time), prioritizing availability over consistency and providing robustness to network partitions. Critically, due to the elected-leader architecture, such lack of consistency will *not* result in uncoordinated action among the agents, although it may result in suboptimal behavior if the leader has stale information. State information that (i) is required for strategic planning and (ii) cannot be inferred locally is synchronized by all agents to the leader and the designated survivor periodically, every 10 seconds. All information required for planning is stored on at least two agents, ensuring robustness to loss of a single agent; information is only lost if both the leader and the designated survivor fail before a new leader and survivor can be elected (which happens within 10s), and before information can be synchronized to them. We refer the reader to [24] for a detailed description of CADRE's SSDB.

2.3 Centralized planning

A strategic planning module runs on the leader and plans and schedules tasks for all agents. The planner module continually evaluates the plan at a set cadence (currently, 1Hz) to evaluate replanning triggers, commit new tasks, delete old tasks, and check multi-agent constraints. Triggers for re-planning include conflict detection, task failure, as well as at specific milestones in the experiment (e.g., when an exploration or distributed measurement task is completed). When a task's scheduled start time falls within a certain window (currently set to 5 seconds), that task is handed off, or *committed*, to the relevant agent controller, described next. Since the planner has visibility into the state of all agents, planning of tasks that require cooperation between multiple agents is straightforward. Enforcement of coordination during decentralized execution is more complex, and is discussed next in Section 2.5. The planner reasons explicitly about agent availability, which is communicated to it directly from the agent controller modules, in order to avoid scheduling and committing tasks for an agent that is not actively running a controller module. The strategic planner and agent controller modules for CADRE are implemented using Multi-mission EXECutive (MEXEC) [26], a resource-aware onboard planning and execution software that uses task networks to generate and execute conflict-free schedules. In task networks, tasks are represented as a tuple containing a flight software (FSW) command to execute the task; an expected duration; a set of constraints; and a set of expected impacts. Constraints include both state constraints, which require that system states remain between prescribed maximum and minimum values; and precedence constraints, where one task must be completed before another can be started. State

constraints include both pre-execution constraints that must be satisfied before a task can be started, and maintenance constraints that must be satisfied throughout execution. Impacts represent the expected change in state from executing the given task (e.g., the expected power draw from driving). During planning, a task's impacts are used to predict the future evolution of the system state; constraints for subsequently-scheduled tasks are then verified against these predicted values. We refer the reader to [26, 27] for a detailed description of the task network representation. A priority-based insertion heuristic is used for planning, as detailed in Section 3.2.4.

2.4 Decentralized execution

Each agent runs an agent controller module, which is responsible for managing task execution. Each agent controller is only able to observe its own agent's state. The agent controller checks task constraints before and during execution; it has the authority to delay the start of a task if any of its starting constraints is unmet, and also to declare a task as "failed" if any of its constraints is violated during the course of execution. In the latter case, the controller issues cleanup commands needed to keep the agent in a recoverable state. The agent controller reports the execution state of all tasks under its authority back to the strategic planner on the leader, which uses this information to trigger re-scheduling.

2.5 Multi-agent constraint checking

Inter-agent coordination of tasks is maintained during execution through the enforcement of multi-agent constraints, performed on the leader. The information available to individual agents is limited to the state of that agent, with the exception of the leader. As such, each agent does not have the necessary information to evaluate all task constraints for tasks requiring coordination among multiple agents, where a constraint violation on another agent's state may trigger a stop. Constraints that cannot be evaluated on individual agents due to lack of information are referred to as "multi-agent constraints". These constraints are evaluated on the leader. Similar to regular constraints, multi-agent constraints include both pre-execution constraints and maintenance constraints. For multi-agent constraints, however, a pre-execution constraint cannot directly delay the start of a task, since the leader is not controlling the task. Instead, the leader does a one-time check to verify that the constraint was satisfied within a short time after the task start time. Maintenance constraints are continuously checked throughout the execution of a task. For any constraint found to be violated, an "abort task" message is sent from the leader to the agent executing the corresponding task. On receiving the "abort task" message, the agent's controller fails the task, causing the leader to replan.

2.6 Integration with leader election

Only the leader agent performs planning and scheduling. However, the leader may change at any time. For this reason, an instance of the strategic planner module exists on every agent; on the leader, the strategic planner is in an active state and schedules and commit tasks, whereas on the non-leader agents the strategic planner module is in an inactive state. To achieve an orderly transition from one leader to the next, (i) only one strategic planner may be actively scheduling and committing tasks at a given moment, and (ii) when a new leader is elected, no agent controller may be actively commanding or tracking tasks that were scheduled and committed by a previous leader.

To accomplish (i), the strategic planner listens to leader election updates and also to acknowledgement messages from each agent controller. When an active strategic planner receives an update from leader election indicating that its agent is no longer the leader, it commits no new tasks and immediately transitions to the inactive state. When an inactive strategic planner receives an update indicating that its agent is now the leader, it waits for acknowledgement messages from each participating agent controller, and only transitions to the active state once all agents have acknowledged that it is leader and that they are ready to accept tasks.

To accomplish (ii), the agent controller modules also listen to leader election updates. As soon as an agent controller receives an update from leader election indicating that the leader has changed, it purges the set of tasks that are under its authority. Any task that has not yet been commanded is immediately dropped. Cleanup commands are issued for any task that has them, and then the controller waits for all running tasks to complete. Only once all tasks under its authority are complete does the agent controller send acknowledgement to the strategic planner on the new leader that it is ready to receive tasks from it.

This approach allows the system to be resilient to unexpected loss of a leader, as the new leader does not rely on information shared from the previous one.

2.7 Remarks on inter-agent communication

We note that the PS&E architecture design assumes that a low-bandwidth, potentially high-latency, bidirectional communication channel is continuously available between agents and the elected leader. If the communication channel between agents may drop arbitrary packets (and potentially all packets), no coordination can be achieved in general — a problem known as the *coordinated attack* problem [13]. We strive to minimize the amount of information sent over links to maximize the likelihood of such information successfully being exchanged, and we design the system to be robust to latency in message delivery. Specifically, if state information from an agent does not propagate to the leader in a timely manner, the leader's plan will rely on stale information from that agent, which results in potentially infeasible plans; however, if the resulting plan indeed violates the agent's constraints, the agent controller will refuse to execute the task based on its local information, ensuring safety. Similarly, if a committed task fails to propagate in a timely manner from the leader to the agent that should execute it, the agent will re-evaluate whether the task remains feasible upon receiving it; if the task is still feasible, it will execute it anyway, and if it is no longer feasible, it will declare it as failed.

For tasks that rely on coordination between agents, task failure on one agent requires stopping all other agents and re-planning; this is accomplished by communicating the task failure from that agent to the leader, triggering a multi-agent constraint failure, which in turn causes the leader to send "abort" messages to other affected agents. If such messages are delayed, agents may act in an uncoordinated manner for some time. We acknowledge this as a limitation of the proposed approach, and conjecture that it may be a fundamental

limitation in presence of communication latency; to minimize the likelihood of occurrence, we reduce the size of "abort" messages to a single task ID, in order to empirically maximize the likelihood of timely delivery even on a disrupted communication link.

3 PLANNING CADRE'S MISSION

3.1 Concept of Operations

CADRE will perform two experiments to demonstrate multi-agent autonomy: exploration and distributed sensing with multi-static radar. An FPGA, not controlled by autonomy, controls a timer that shuts down all agents at 25 and 55 minutes past the hour, and restarts them 0 and 60 minutes past the hour [8]. Within each 25-minute cycle, the team must make progress towards the active experiment as follows.

3.1.1 Exploration. During the exploration experiment, ground operators uplink the boundary of a region to explore to the team. The content of the region is initially unknown to the agents. The goal of the experiment is for the agents to collectively observe every portion of the region that is reachable from their initial locations with on-board stereo cameras; and to classify every reachable portion of the region as "traversable" or "obstacle". To do this, the CADRE team partitions unknown portions of the region into as many sub-regions as there are rovers, and assigns each sub-region to a rover; each rover then explores its own sub-region using frontier-based exploration [32], as shown in Figure 3. Maps are periodically synchronized to the leader, which merges them in a joint map, and sub-regions are periodically re-computed as the unknown portions of the map shrink. We refer the reader to [17] for a detailed description of the exploration architecture.

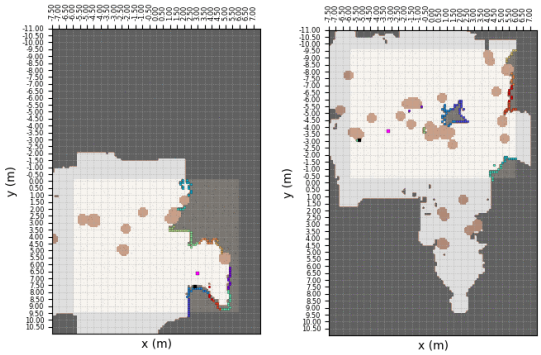

Figure 3: Exploration: two rovers (magenta dots) explore the sub-region assigned to them (lighter rectangle) by selecting a frontier (colorful segments) between traversable (off-white) and unexplored (light gray) portions of the sub-region.

3.1.2 Distributed Sensing. During the distributed sensing experiment, rovers are tasked with driving in formation between assigned waypoints while collecting multi-static ground-penetrating radar readings. Ground operators specify a waypoint to reach as a team, typically tens of meters away from the rovers' initial location; a formation to maintain, specified as a set of inter-rover distances; and a maximum allowable deviation from the prescribed formation that will, if not violated, ensure sufficient quality of the radar data. Agents must plan a set of trajectories that will reach the prescribed waypoint while staying in formation. The map of the region where the experiment is performed may be partially unknown to the rovers (that is, the distributed sensing experiment may stray outside the exploration region area). Thus, the rovers are likely to encounter initially-unknown obstacles that require replanning. Team-level planning is performed by the leader through sampling-based motion planning [15] on the rovers' joint state space; individual rovers are then assigned time-stamped corridors around the computed team trajectory (shown in Figure 4). If every rover remains within its own time-stamped corridor, the maximum allowable deviation between rover distances is guaranteed not to be violated. If a rover is unable to remain within the allocated corridor, all other rovers must stop; the leader then computes a new collision-free team trajectory.

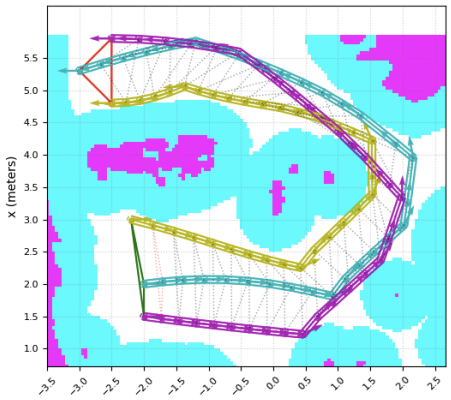

Figure 4: Distributed sensing: each rover is assigned a "tube" around its nominal trajectory, and must travel inside the tube while satisfying temporal constraints.

3.2 Task Network Encoding

3.2.1 Tasks. To encode the exploration and distributed sensing behaviors, we build upon single-agent localization, mapping, and mobility tasks to create higher-level behaviors that allow the system to operated autonomously as a whole. PS&E's role is to manage the interaction between these tasks, and their system-level impacts (e.g., their power consumption and impact on rover temperature). To accomplish this, we model a set of abstract tasks sufficient to predict system impacts and constrain agent behavior to safe and desirable actions. Specifically, the tasks considered in PS&E are:

- *SSDB map synchronization*: every rover copies its new maps and pose to the leader; this task must be executed prior to exploration and formation navigation planning to ensure that the leader has access to up-to-date maps from all rovers;
- *SSDB backup*: the designated survivor maintains a backup of all agents' SSDB to insure against data loss and enable efficient leader reelection;
- *exploration navigation planning*: the region to explore is divided in as many subregions as there are rovers; the task is executed on the leader during the exploration experiment;
- *driving to explore*: each rover is tasked to explore its sub-region via frontier-based exploration; the task is executed on each participating rover during the exploration experiment;
- *formation navigation planning*: a set of trajectory "tubes" is computed for the participating rovers; the task is executed on the leader during the distributed sensing experiment;

- *driving in formation*: each rover is tasked with following the tube around its trajectory; the task is executed on participating rovers during the distributed sensing experiment;
- *stopping a drive early*: while driving tasks stop on their own when they reach their destination, this task is used to stop early when time or resource constraints are close to violation;
- *shutting down the agent software and electronics*: if time and resources continue to be insufficient, the task puts the agent in a low power mode.

Figure 5 shows the set of tasks considered for the formation driving experiment, and selected inter-task constraints.

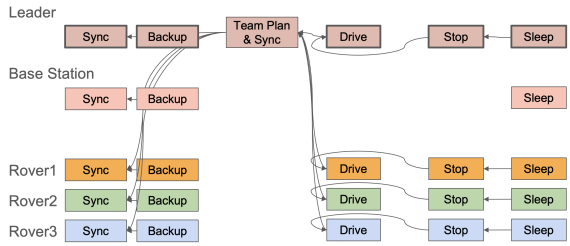

Figure 5: Abstract representation of the task network for both formation sensing and exploration. Arrows represent precedence constraints. Resource and multi-agent constraints are not shown.

Each task (and in particular driving) involves a complex set of individual behaviors, which are modeled with a combined sets of constraints and impacts. The energy consumed and heat produced by the CPU during image processing, for example, is rolled up with the power and heat from driving motors. Closed-loop replanning is used to correct for inaccuracies in the impact models. Constraints are used for both ensuring the proper task ordering as well as preventing over-subscription of system resources.

3.2.2 Resources. The three modeled system resources are:

- time: tasks must be performed within the maximum allotted 25 minute interval;
- battery state-of-charge (SOC): rover batteries are recharged by solar arrays, but drained while performing tasks; the base station is powered by the lander, and has no battery;
- CPU temperature: operating for long durations close to Lunar noon can exceed the maximum operating temperature.

The strategic planner must anticipate the FPGA-commanded shutdown at 25 and 55 minutes past the hour; stop driving; and put the agent in a low-power mode before this timer expires. This allows logs and other data to be saved before power off. In addition, the battery SOC must be maintained about a minimum threshold (e.g. 20%). Simulation and analysis has shown that, during periods with the sun high overhead, stopping the drive and other tasks is sufficient to allow the solar panels to recharge the batteries. At other times, the agents must be put into a low-power mode through the shutdown task. Temperature, however, limits agent operations at exactly the opposite periods of the lunar day. When the sun is low in the sky, lunar temperatures are low enough that there is no risk of overheating the CPU. But around lunar noon, the rovers can only drive for a few minutes before the CPU temperature exceeds its operational limit of 65° C.

3.2.3 Resource Models. Both battery SOC and temperature are measured through onboard sensors; whenever replanning is performed, predicted impact for future tasks are applied starting from this current state. Because of this, and the comparatively short planning horizon, impacts use simple models of resource change.

For battery SOC, we use a roll-up of the power loads from all behaviors started by the task. Using this, and the battery capacity, we get a linear prediction of the battery SOC. For heating discharge, and re-charging provided by the solar arrays, we use a coarse estimate based on the lunar time-of-day. The power required by heaters impacts a background discharging rate, which is reduced during power-consuming tasks to account for supplemental heating from power loads of that task (assuming that a uniform 80% of the load will be dissipated as usable heat). Degradation in solar array performance due to rover tilt is omitted for simplicity.

Temperature proved more difficult to model. The impact of tasks on rover temperature depends in a nonlinear way on the vehicle's starting temperature, the ambient temperature, rover tilt, and the power loads being applied. Simulations were performed at solar angles between 20° and 90° with 10° steps; each simulation cycled through the different rover operating modes (i.e., the set of tasks being executed) multiple times. These simulations helped identify which factors have the largest effect on temperature rate-of-change. Sun angle and rover operating mode were identified as the two key factors affecting temperature rate of change. We also identified the CPU temperature as the most constrained temperature resource in all simulations; accordingly, we only model CPU temperature in planning, effectively treating each rover as a one-node thermal model. Surprisingly, we found that the CPU temperature at the start of an operating mode had a minimal impact on the heating rate; therefore, we omitted it from the planning model. We did observe that the rate of change of temperature was markedly nonlinear, with an initial steep change that leveled off after about two minutes — in line with the expected physical behavior of heat conduction and radiation. The existing task model in MEXEC, however, only supports linear rates. We considered implementing a more accurate model that divides tasks into multiple parts to better capture this behavior. However, simulations and analysis showed that the added complexity would provide a relatively small benefit compared to more frequent replanning; accordingly, a linear model was used.

3.2.4 Planning and Scheduling Algorithm. When the system replans, it always has one of two goals, namely, to continue exploring or performing distributed measurements. The tasks performed during the previous wake cycle are not explicitly part of the state considered for planning: their impacts are fully captured by the rovers' maps for exploration, and the rovers' locations for distributed measurement. This allows us to use one simple scheduling process, regardless of when and why the system performs re-scheduling:

- un-schedule any task not committed for execution;
- read and apply the current system state;
- greedily schedule un-executed tasks in priority order, scheduling each task as close as possible to its preferred start time while ensuring constraint satisfaction, and rejecting any task that cannot be scheduled without constraint violations.

Pseudocode for the planning algorithm is reported in the Extended Version [21]. The choice of this simple scheduling process

has implications on how the task network is defined. First, priority order scheduling meant that we cannot assume that lower priority tasks will exist when a higher priority task is scheduled. Second, because no additional tasks are created during planning, the task network must contain all of the tasks we will ever need. For example, because driving requires a team plan to be generated, the team planning task must be higher priority, while the driving task must be constrained to prevent scheduling if a team planning task does not exist. Moreover, because multiple driving cycles could be scheduled, the task network includes multiple instances of all tasks. But a second instance of driving should not be scheduled if one already exists, unless that first drive was stopped or completed. Constraints were used to ensure proper task ordering and allow multiple exploration and distributed measurement cycles.

3.3 Multi-agent Task Networks

For tasks that must be executed on all three rovers (e.g. map synchronization and driving), a separate task instance is created for each rover; constraints for that task are tied to the specific rover's state. The state representation is identical for all rovers. Since the base station has no battery and doesn't drive, its state representation does not include battery SOC, and no driving tasks are created for it. The task network references a leader ID for tasks that must be executed on the leader (namely, team planning); but it does not have explicit knowledge of which agent is assigned as the leader.

For exploration, driving tasks for the participating rovers are independent of each other. If one rover is unable to drive for any reason, this does not impact the ability to schedule and execute driving for the other rovers. The distributed measurement experiment, however, requires coordinating tasks across all participating rovers. To achieve this, task hierarchy was used, a construct where a parent task decomposes into subtasks assigned to different rovers. The parent task is defined with the constraints and impacts of all subtasks, to ensure that the parent is scheduled at a time that is valid for all rovers. Then, the subtasks are constrained to be scheduled at the exact start and end of the parent. A multi-agent constraint is used to enforce continuous coordination of the tasks during execution. At the start of a formation drive, the parent task changes an internal state to "coordinated", while each subtask requires this state to be "coordinated". During execution, if one of the rover drives fails, it changes the state to "not coordinated". This creates a multi-agent constraint conflict that triggers an abort of the drive tasks on the other rovers, as described in Section 2.5.

Managing agent participation. The CADRE concept of operations allows operators to manually designate which agents participate in a given experiment. By default, all agents are included; however, ground operators can exclude one or more agents in case of failures, or to support anomaly investigations. The task networks support these alternate configurations using task constraints that require the state of the assigned agent to report it as "participating". For exploration, because tasks execute independently, the participation constraint is implemented similar to any other constraint. For formation sensing, however, driving is only scheduled if the tasks for all *participating* rovers can be scheduled. We implement this by creating alternate hierarchies, one for each of the possible subsets of the three rovers. Parents with larger subsets are assigned higher priorities, so that they will be tried first. A subset that contains a subtask for a non-participating rover will fail that constraint, causing the full subset to fail. However, if a subtask for a rover fails to schedule for any other reason, we do not want to schedule a formation drive using the remaining rovers. Instead, we want to wait for all participating rovers to be ready to drive. This means that, for example, the subset for driving with only Rover 2 and Rover 3 includes a constraint that requires Rover 1 to be non-participating.

3.4 Pytasknet

CADRE's declarative task model was created using a Python library called Pytasknet. This library includes the basic classes and functions for creating tasks, and greatly simplified the effort of creating CADRE's task networks. The Python objects and references abstract away many of the details of creating references in MEXEC's XML task network representation. Moreover, the Python language provides the full power of a procedural language to create these task networks. For CADRE operations, where many tasks are identical except for the designated rover, the ability to create tasks within functions and loops was essential.

4 EXPERIMENTAL RESULTS AND V&V

Throughout the development and V&V process, the PS&E subsystem for CADRE was tested over multiple venues at increasing levels of fidelity, as shown in Table 1.

Table 1: Testing venues used for validation of CADRE PS&E

	Scheduling	Nominal exec.	Off-nominal exec.	FSW integration	Actual driving	Outdoor driving	Full sensor suite	Specialized HW	Flight HW
Batch planning	✓								
ROS sim	✓	✓	✓						
Dragonfarm	✓	✓	✓	✓				✓	
Dev. Models	✓	✓	✓	✓	✓	✓		✓	
Flight Models	✓	✓	✓	✓	✓		✓	✓	✓

Initial evaluation of the task networks and the scheduling implementation were performed using MEXEC batch planning, a one-shot process that produces a schedule based on a task network and a set of initial state values. To test the interplay between planning and execution, we conducted simulations first in a ROS-based [20] simulator, with stand-ins for all system behaviors that have an impact on the planned tasks and the states they interact with. The simulator emulates task execution by idling for an appropriate duration, then returning the same fault/success codes used by FSW, and updating states based on the tasks' impact models. Individual tasks can return early, return late, or report failure, based on operator input. For the ROS testing campaign, we developed a test matrix, Table 2, that crossed all of the tasks with all of the possible execution behaviors of a task. Each execution behavior (e.g. starting late, failing) can have a different impact on the re-planning and execution of subsequent tasks. In each relevant case, we verified that the proper response was taken by both the Agent Controller and the Strategic Planner. The ability to quickly and easily perform

these tests was key to the development of the task network, and allowed us to identify multiple bugs where a task definition exhibited undesirable behavior during execution or re-scheduling.

Table 2: Testing campaign with ROS simulations. As the first task to be scheduled, SSDB sync is not impacted by previous tasks and not expected to start late. Similarly, the stop and sleep tasks are not expected to have variable duration or fail.

Task	Nominal	Starts late	Runs late	Ends early	Fails
SSDB Sync	✓		✓	✓	✓
Team Planning	✓		✓	✓	✓
Formation	✓	✓	✓	✓	✓
Exploration	✓	✓	✓	✓	✓
Stop	✓	✓			
Sleep	✓	✓			

Higher-fidelity simulations were then run on the *Dragonfarm* testbed, which consists of a set of the same ModalAI VOXL system-on-a-chip modules used on the rovers, connected via Ethernet to provide a platform that can execute the full CADRE flight software deployment (which uses the F' framework [4]), with the same computing resources as flight hardware. In this context, we were able to perform extensive end-to-end simulation of both the exploration and formation sensing experiments with a variety of leader and participant configurations and initial state conditions.

Extensive testing was also carried out on mobility platforms – specifically, Development Models (DM) and Flight Models (FM), shown in Figure 6. The FMs are the hardware that will fly to Reiner Gamma; the DMs are reduced-fidelity hardware that present the same computing and actuation as the FMs, but a reduced sensor suite, and are suitable for outdoor testing on Earth.

Figure 6: CADRE's development models (DMs) in the Mars Yard (left) and flight models (FMs) in the cleanroom (right).

This multi-venue approach offered a comprehensive view of system performance. Testing on the FM rovers, despite their stringent operational constraints (e.g., being confined to the cleanroom environment, and subjected to rigorous safe handling guidelines), provided critical insights. The FM units were equipped with temperature and state-of-charge sensors not available on DMs, providing a complete picture of system performance and yielding high-fidelity data for testing the PS&E system under flight-like conditions. In contrast, the DM rovers were not restricted to cleanroom conditions and were driven extensively in the Mars Yard outdoor testing venue, which offered more freedom and terrain realism and allowed testing interactions with the Guidance, Navigation, and Control (GNC) subsystem. Balancing the strengths of both DM and FM testing was essential in ensuring that the autonomy system could handle both realistic driving tasks, and noisy sensor measurements.

Key tests on the DM hardware focused on running exploration and formation sensing experiments with flight-like wake-sleep cycles. In these tests, the rovers autonomously woke up in autonomy mode, configured their sensors and components, and loaded the experiment task network into the strategic planner. The PS&E stack handled all task planning, monitored task constraints during execution, and replanned as needed — adjusting durations or replanning tasks without operator assistance or intervention.

Exploration testing was conducted across regions ranging from 6x6m to 22x13m. For distributed sensing, goal situated 6m to 20m from the agents' initial locations were tested. We verified that GNC could trigger replanning when the formation trajectory tube couldn't be followed, and that multi-agent constraint checking performed as expecting by stopping all agents in response. We also verified the ability to store and access state information across wake-sleep cycles, and demonstrated the ability to change the elected leader during different stages of the experiments. Testing under varying terrain configurations — from sparse to cluttered rock and crater fields, and under the harsh shadows of night-time testing — further validated the system's robustness.

Key testing on the FM hardware involved Autonomy Day-in-the-Life experiments, where the stationary FM rovers performed tests under FPGA-driven wake/sleep operations. These tests ran with continuous cycles for over 9 hours, transitioning between autonomy, nominal, and safe modes. With real (albeit cleanroom-based) thermal measurements and battery state-of-charge calculations, we verified the planner's ability to generate appropriate plans based on thermal and power constraints. The system responded by either gracefully stopping experiments if constraints were violated, or changing task durations when limits were approaching. Additionally, the tests validated correct responses to off-nominal cases, such as sudden drops in battery state or temperature spikes, as well as agents entering safe mode. The tests also confirmed successful planning and state synchronization across all four agents, unlike the DM hardware, which only sports three agents. Overall, not only did this testing campaign give us confidence that the distributed planning, scheduling, and execution system can perform under the scenarios required by the CADRE mission, but also gave us key insights into how to operate autonomous and multi agent spacecraft.

5 CONCLUSION

In this paper, we describe the planning, scheduling, and execution architecture for CADRE, a NASA mission that will demonstrate multi-agent autonomy on the Moon in 2025/2026. Testing of the autonomy stack also provided unique insight into verification and validation of autonomy, and operations of multi-agent systems. Successful completion of CADRE's mission will demonstrate the high technology readiness of multi-agent planning, scheduling, and execution in challenging planetary surface environments (in particular, on the Moon), enabling infusion in future science-driven planetary exploration missions.

ACKNOWLEDGMENTS

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

REFERENCES

[1] Caleb Adams and Jeremy Frank. 2024. The Distributed Spacecraft Autonomy (DSA). (2024). 34th International Conference on Automated Planning and Scheduling (ICAPS) Summer School.

[2] Caleb Adams, Brian Kempa, Michael Iatauro, Jeremy Frank, and Walter Vaughan. 2023. An Overview of Distributed Spacecraft Autonomy at NASA Ames. In *Proceedings of the Small Satellite Conference*. Utah State University.

[3] Keenan Albee, Sriramya Bhamidipati, Federico Rossi, Joshua Vander Hook, and Jean-Pierre de la Croix. 2024. Lunar Leader: Persistent, Optimal Leader Election for Multi-Agent Exploration Teams. In *Int. Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace)*. Auckland, NZ.

[4] Robert Bocchino, Timothy Canham, Garth Watney, Leonard Reder, and Jeffrey Levison. 2018. F Prime: an open-source framework for small-scale flight software systems. In *Small Satellite Conference*. Logan, UT.

[5] Daniel Cellucci, Nick B Cramer, and Jeremy D Frank. 2020. Distributed spacecraft autonomy. In *ASCEND 2020*. 4232.

[6] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandl, S. Frye, B. Trout, S. Shulman, and D. Boyer. 2005. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. , 196–216 pages.

[7] Han-Lim Choi, Luc Brunet, and Jonathan P How. 2009. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics* 25, 4 (2009), 912–926.

[8] Jean-Pierre de la Croix, Federico Rossi, Roland Brockers, Dustin Aguilar, Keenan Albee, Elizabeth Boroson, Abhishek Cauligi, Jeff Delaune, Robert Hewitt, Dima Kogan, Grace Lim, Benjamin Morrell, Yashwanth Nakka, Viet Nguyen, Pedro Proenca, Gregg Rabideau, Joseph Russino, Maira Saboia da Silva, Guy Zohar, and Subha Comandur. 2024. Multi-Agent Autonomy for Space Exploration on the CADRE Lunar Technology Demonstration. In *IEEE Aerospace Conference*. Big Sky, MT, 1–14. https://doi.org/10.1109/AERO58975.2024.10521425

[9] Daniel Gaines, Gary Doran, Heather Justice, Gregg Rabideau, Steve Schaffer, Vandana Verma, Kiri Wagstaff, Ashwin Vasavada, William Huffman, Robert Anderson, Ryan Mackey, and Tara Estlin. 2017. A Case Study of Productivity Challenges in Mars Science Laboratory Operations. In *International Workshop on Planning and Scheduling for Space (IWPSS 2017)*. Pittsburgh, PA.

[10] D. Gaines, G. Doran, H. Justice, G. Rabideau, S. Schaffer, V. Verma, K. Wagstaff, V. Vasavada, W. Huffman, R. Anderson, R. Mackey, and T. Estlin. 2016. *Productivity challenges for Mars rover operations: A case study of Mars Science Laboratory operations*. Technical Report D-97908. Jet Propulsion Laboratory.

[11] Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. 1983. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and systems (TOPLAS)* 5, 1 (1983), 66–77.

[12] Seth Gilbert and Nancy Lynch. 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News* 33, 2 (2002), 51–59.

[13] J. N. Gray. 1978. *Notes on data base operating systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 393–481. https://doi.org/10.1007/3-540-08755-9_9

[14] RM Haberle, Javier Gómez-Elvira, M de la Torre Juárez, A-M Harri, JL Hollingsworth, Henrik Kahanpää, MA Kahre, M Lemmon, FJ Martín-Torres, M Mischna, et al. 2014. Preliminary interpretation of the REMS pressure data from the first 100 sols of the MSL mission. *Journal of Geophysical Research: Planets* 119, 3 (2014), 440–453.

[15] Sertac Karaman and Emilio Frazzoli. 2011. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30, 7 (2011), 846–894.

[16] John C Mankins. 1995. *Technology readiness levels*. Technical Report. Advanced Concepts Office, Office of Space Access and Technology, NASA.

[17] Sharan Nayak, Federico Rossi, Grace Lim, Michael W. Otte, and Jean-Pierre de la Croix. 2024. Multi-Robot Exploration for the CADRE Mission. (2024). Under review.

[18] NetSAG. 2010. *Mars Network Science Analysis Group (NetSAG) Final Report*. Technical Report. MEPAG. https://mepag.jpl.nasa.gov/meeting/2010-03/NetSAG_MEPAG_Final_v2.pdf

[19] Shreya Parjan and Steve Chien. 2023. Decentralized Observation Allocation for a Large-Scale Constellation. *Journal of Aerospace Information Systems (JAIS)* 20, 8 (August 2023), 447–461. https://doi.org/10.2514/1.I011215

[20] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, Japan, 5.

[21] Gregg* Rabideau, Joseph* Russino, Andrew Branch, Nihal Dhamani, Tiago Stegun Vaquero, Steve Chien, Jean-Pierre de la Croix, and Federico Rossi. 2025. Planning, scheduling, and execution on the Moon: the CADRE technology demonstration mission (Extended Version). (2025). https://arxiv.org/abs/2502.14803

[22] Federico Rossi, Saptarshi Bandyopadhyay, Michael T. Wolf, and Marco Pavone. 2021. Multi-Agent Algorithms for Collective Behavior - A structural and application-focused atlas. (2021). https://arxiv.org/abs/2103.11067

[23] Federico Rossi, Maira Saboia, Siddharth Krishnamoorthy, and Joshua Vander Hook. 2023. Proximal Exploration of Venus Volcanism with Teams of Autonomous Buoyancy-Controlled Balloons. *Acta Astronautica* 208 (2023), 389–406. https://doi.org/10.1016/j.actaastro.2023.03.003

[24] Maira Saboia, Federico Rossi, Viet Nguyen, Grace Lim, Dustin Aguilar, and Jean-Pierre de la Croix. 2024. CADRE MoonDB: Distributed Database for Multi-Robot Information-Sharing and Map-Merging for Lunar Exploration. In *Int. Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace)*. Auckland, NZ.

[25] Florian Sellmaier, Thomas Uhlig, and Michael Schmidhuber. 2022. *Spacecraft Operations*. Springer.

[26] M. Troesch, F. Mirza, K. Hughes, A. Rothstein-Dowden, R. Bocchino, A. Donner, M. Feather, B. Smith, L. Fesq, B. Barker, and B. Campuzano. 2020. MEXEC: An Onboard Integrated Planning and Execution Approach for Spacecraft Commanding. In *Workshop on Integrated Execution (IntEx) / Goal Reasoning (GR), International Conference on Automated Planning and Scheduling (ICAPS IntEx/GP 2020)*. Also presented at International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS 2020) and appears as an abstract.

[27] M. Troesch, F. Mirza, G. Rabideau, and S. Chien. 2019. Onboard re-planning for robust mapping using pre-compiled backup observations. In *11th International Workshop on Planning and Scheduling for Space (IWPSS)*. Berkeley, California, USA, 168–175.

[28] Steven D. Vance, Sharon Kedar, Mark P. Panning, Simon C. Stähler, Bruce G. Bills, Ralph D. Lorenz, Hsin-Hua Huang, W.T. Pike, Julie C. Castillo, Philippe Lognonné, Victor C. Tsai, and Alyssa R. Rhoden. 2018. Vital Signs: Seismology of Icy Ocean Worlds. *Astrobiology* 18, 1 (Jan. 2018), 37–53. https://doi.org/10.1089/ast.2016.1612

[29] Vandi Verma, Mark W Maimone, Daniel M Gaines, Raymond Francis, Tara A Estlin, Stephen R Kuhn, Gregg R Rabideau, Steve A Chien, Michael M McHenry, Evan J Graser, et al. 2023. Autonomous robotics is driving Perseverance rover's progress on Mars. *Science Robotics* 8, 80 (2023).

[30] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. 2014. Designing Collective Behavior in a Termite-Inspired Robot Construction Team. *Science* 343, 6172 (2014), 754–758. https://doi.org/10.1126/science.1245842

[31] Michael T Wolf, Amir Rahmani, Jean-Pierre de la Croix, Gail Woodward, Joshua Vander Hook, David Brown, Steve Schaffer, Christopher Lim, Philip Bailey, Scott Tepsuporn, et al. 2017. CARACaS multi-agent maritime autonomy for unmanned surface vehicles in the Swarm II harbor patrol demonstration. In *Unmanned systems technology XIX*, Vol. 10195. SPIE, 218–228.

[32] Brian Yamauchi. 1997. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 146–151.

[33] Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. 2024. Decentralized, Decomposition-Based Observation Scheduling for a Large-Scale Satellite Constellation. In *International Conference on Automated Planning and Scheduling*. Banff, Canada.