Dynamic Shared Computing Resources for Multi-Robot Mars Exploration

Joshua Vander Hook, Tiago Vaquero, Martina Troesch,
Jean-Pierre de la Croix, Joshua Schoolcraft, Saptarshi Bandyopadhyay, Steve Chien
Jet Propulsion Laboratory, California Institute of Technology

Abstract

The NASA roadmap for 2020 and beyond includes several key technologies which will have a game-changing impact on planetary exploration. The first of these is High Performance Spaceflight Computing (HPSC), which will provide orders of magnitude increases in processing power for next-generation rovers and orbiters (Doyle et al. 2013). The second is Delay Tolerant Networking, which overlays the Deep Space Network, providing internet-like abstractions and store-forward to route data through intermittent delays in connectivity. The third is a trend toward small, co-dependent robots included in flagship missions (MarCO, PUFFER, and Mars Heli). Taken together, these imply an increasing amount of communication and computing heterogeneity on Mars in coming decades.

Motivated by these technological trends, we study the concept of Mars on-site shared analysis, information, and communication (MOSAIC) for Mars exploration. The key algorithmic problem associated with MOSAIC networks is simultaneous scheduling of computation, communication, and caching of data, which we illustrate using the three scenarios. We present models, preliminary solutions, and simulation results for two scenarios, showing how mission efficiency relates to communication bandwidth, processing power, geography of the environment, and optimal scheduling of computation, communication, and data caching. The third scenario illustrates future directions of this work.

1 Introduction and Related Work

Three trends are poised to significantly change mission concepts for future NASA planetary exploration. While previous missions involved single robots with limited processing capability, the combination of new networking technology, advanced computation hardware, and small-bodied robot designs is making multi-robot missions more attractive.

In an effort to modernize the flight computing hardware available for NASA missions, the High Performance Spaceflight Computing (HPSC) initiative was announced in 2013 (Doyle et al. 2013; Powell et al. 2011; Mounce et al. 2016). Unlike the current generation of computing, this program aims to keep NASA computing technologies at most one generation behind commercial technologies. HPSC is expected to become a mainstay in post-2020 deployments.

The second key emerging technology is Delay or Disruption Tolerant Networks (DTNs). DTNs span communications links in an overlay architecture, enabling connectivity across network boundaries in a transparent manner, regardless of multiple potentially disparate network link layer protocols. A core principle of this overlay quality is the ability of individual nodes to store network data for possibly long durations before forwarding it to another node. This store-and-forward paradigm is central to DTNs. Many features of Delay Tolerant Networking architectures are of particular utility in the deep space interplanetary communications realm, where a multitude of link layers, bandwidth constraints, and disruptions are expected during end-to-end transfer of mission commands and data (Wyatt et al. 2017).

Finally, an interest in multi-robot systems re-emerged. Currently planetary exploration is limited to benign operating areas due to the inability to land, traverse challenging terrain, or generally too great a risk for the primary mission asset. Unfortunately, the most compelling locations are often in these extreme terrains. Small, low cost, expendable rovers could transport key sensors and instruments to locations considered too risky for the primary lander, rover, or astronaut. Also due to the high communications latencies of deep space missions these expendable rovers must minimize their dependence on ground control and be able to operate primarily autonomously. These small craft can be released from par-

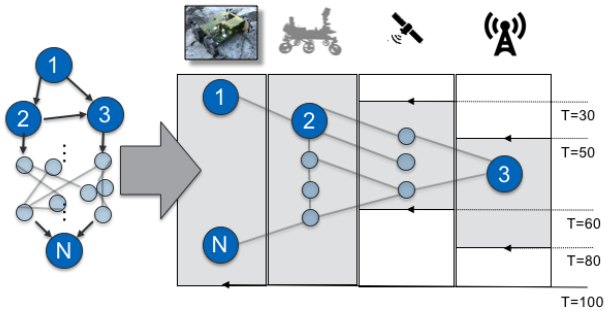


Figure 1: Illustrative MOSAIC scenario. A set of processing and data-driven tasks (left as dependency graph) must be mapped to multiple assets with heterogeneous computing, communication, and energy capacities. Each asset is also available over a fixed time window due to terrain effects or orbital parameters. The goal is to compute all the required tasks as quickly as possible.

ent rovers and guided toward sampling targets which may be out of reach of the main craft, either because of risk, or simply to avoid delays from stopping. The "daughter-craft" do not have advanced processing capabilities due to weight, power, and cost constraints, but are attractive for a number of science targets, such as being left behind to investigate transient detections, risky exploration areas such as Recurring Slope Lineae, or wide-area sampling for In-Situ Resource Utilization. Two examples of such potential future systems now being considered for development are the Mars Helicopter, and the "PUFFER" rover (Pop-Up Flat-Folding Explorer Robots) (Karras et al. 2017).

Combining these three trends, we envision scenarios in which a system containing two or more robotic agents with large discrepancies in processing power, communication bandwidths, data capacities, and energy storage must collaborate to achieve a variety of realistic remote science missions.

The concept of study in all three scenarios is that advanced, software-driven robotic capabilities can be realized on small, resource-constrained, high-risk "edge" devices by optimizing data flows and processing assignments among all the devices. In this paper we formalize this problem and present preliminary results in modelling and analyzing Mars exploration missions. Because data and computation are shared among many devices, we dub a local computation-sharing network a MOSAIC (Multi-robot On-site Shared Analytics Information and Computing) network.

Our paper is organized as follows. First, we derive our problem statement in Section 2. We decompose objectives into a set of computing tasks, each of which generates data products which must be fed into subsequent tasks (possibly by transmitting between agents). Each task may be conducted by humans or robots. In Section 3 we discuss a search routine which can identify how the computational load can be distributed over the network.

We describe our study scenarios in Section 4. In the first illustration, (Section 4.1) we consider a single, cpu-bereft asset which can request computation from a nearby base station or visible orbiter. We study both a single PUFFER released from a base station (first scenario), and the Mars 2020 rover assisted by a hypothetical HPSC (second scenario). Planning for a potential Mars Sample Return campaign is dominated by the need for autonomous traversals of increasingly fast speeds, and we show an analysis of impact that computation sharing can have on mission success.

The third scenario (Section 4.3) is a mother / daughter craft design consisting of a large centralized asset (a human or flagship rover) controlling one or more agile, but less-capable "scouts" for an area search task. In this regime we discuss some emergent behavior like data relay and automatic choice of a centralized computing agent.

2 Problem Description

In this section, we describe how we frame the problem of dynamic shared computation for Mars exploration. We define a data communication and processing workflow that represents the mission objectives and intermediate goals at a high level.

Our primary abstraction is that of a Server Graph. Let there be $N \in \mathbb{Z}^+$ agents in the network, where $\mathbb{Z}^+$ denotes positive integers. The robot agents are denoted by $A_1, A_2, \ldots, A_N$. Each agent has on-board processing, memory, and communication links.

2.1 Computation

The agents perform $M \in \mathbb{Z}^+$ data-driven tasks. The set of M tasks is denoted $\mathbb{T}$. We consider heterogeneous processing times, so the time cost of executing task T on agent i is given by: $C_i^t(T)$. The model represents, e.g., the worst-case, expected, or bounded computation time, and so all the times are deterministic. In addition, program outputs are the same irrespective of the agent doing the computing (or are just as useful). Task T performed by robotic agent i may also include an energy cost, $C_i^e(T)$. If an agent has access to two or more *different* processing units, we model those as two co-located agents. If an agent has access to two or more *similar* processing units, we adjust the costs of each task to reflect its level of parallelization, but otherwise consider them the same processor.

Tasks produce data products. Data products for task T are denoted $d(T)$. If a task produces more than one data product, we model it as multiple tasks, one per produced data product. The size of the data products are known a-priori, and labelled as $s(T)$ for task T.

Let P_T be a set of predecessor tasks for T. Then $j \in P_T$ means task T depends on the output of task j. A task may have multiple prerequisite sets, one of which must be satisfied entirely. That is, a task must have only one of its prerequisite sets satisfied.

The static software network SN captures dependencies as the flow of information through various individual programs to solve the complex computing task.

Finally, we allow some of the tasks to be required and some to be optional. Optional tasks have a reward score ($r(T)$). The set of required tasks is denoted $\mathbb{R} \subseteq \mathbb{T}$.

Assumptions: The software network SN does not have any cycles. The mission statement for each problem/scenario can be stated as a software network SN.

A *solution* is a mapping of tasks to servers (agents) and start-times denoted

$$\mathbb{S} : i \rightarrow (A_j, t) \tag{1}$$

where (2)

$$j \in [1, \ldots, N] \tag{3}$$

$$t \geq 0 \tag{4}$$

Each agent's computing schedule in a solution is denoted

$$\mathbb{S}_i = j \rightarrow_i (t) \tag{5}$$

and has cost equal to the time required to complete the last task in the agent's queue,

$$C(\mathbb{S}) = \max_i C(\mathbb{S}_i) \tag{6}$$

where (7)

$$C(\mathbb{S}_i) = \max_j \mathbb{S}_i(j) + C_i^t(j) \tag{8}$$

To execute a specific task in the software network, an agent must have all the data products from one of the tasks predecessor sets, either by computing them directly, or by receiving them by communication from other agents. To communicate, we model each agent as having a Delay Tolerant Networking (DTN) stack to enable communication, as defined next.

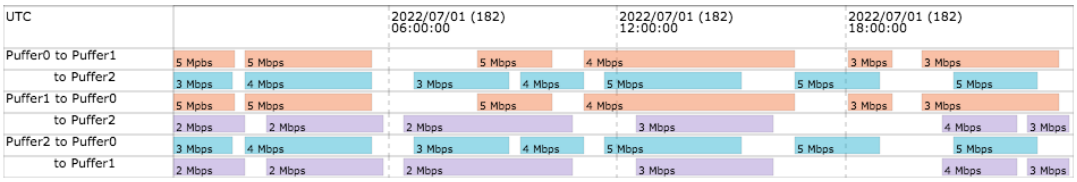


Figure 2: Contact graph for 3 agents showing times and bandwidths available

2.2 Communication

A key feature of DTN-based networking is Contact Graph Routing (CGR) (Wyatt et al. 2017). CGR takes into account predictable link schedules and bandwidth limits to automate data delivery and optimize the use of network resources. The contact graph describing a network's links over time is distributed to participating DTN nodes, allowing each node a clear picture of how to route data in an optimal manner. Each scenario is complicated by the relative geography of the agents which may affect communication rates, their motion plans through the environment, and the nature of long-distance communications such as light delays or degraded signal strengths. The practical effect of incorporating DTN's store-forward mechanism into the scheduling problem is that it is possible to use mobile agents as *robotic routers* to ferry data packets past communication interference.

The time-varying contact graph CG captures the communication network topology between agents. For each agent, the graph provides a list of all the time intervals during which it can establish a directed communication link with another agent. An example timeline representation of a contact graph for 3 agents showing available bandwidths can be seen in Figure 2.

Links have a time varying data rate from 0 (not connected) to ∞ (communicating to self), denoted by $r_{ij}(t)$ for the rate from A_i to A_j at time t. Thus, communication links are directed.

At any time k, let $\mathcal{G}_k$ be the graph representing the set of agents it can send to or receive from. Vertices $\mathcal{V} = \{1, \ldots, N\}$ and the directed edges $\mathcal{E}_k$ along which communication is possible. That is, if information can flow from the i^{th} agent to the j^{th} agent at the k^{th} time instant (where $i, j \in \mathcal{V}$), then the edge $\overrightarrow{ij} \in \mathcal{E}_k$.

Communication between agents is a task with cost determined by the size of the data product and the current data rate between agents. The task of communicating the data product $d(T)$ from A_i to A_j at time t requires time $C^t_{ij}(T) \propto s(T)/r_{ij}(t)$ for *both* agents and energy equal to $C^e_{ij}(T)$ on the *sending* agent.

Assumptions: Agents take 0 time to communicate the solution to themselves. Intervals with non-zero data rates are sufficiently long to transmit any data product (or they would be "effectively zero").

Problem 1 (Distributed Computation)**.** *Given a set of tasks modelled as a software network SN, a list of computational agents A_i $i \in [1 \ldots N]$, a contact graph CG, and a maximum schedule length $C^\star$, find a solution which is a mapping of tasks to servers (agents) and start times, $\mathbb{S} = f(i) :\rightarrow (A_j, t)$, such that:*

- *The maximum server cost, $C(\mathbb{S}) = \max_j C(\mathbb{S}_j)$ is no more than $C^\star$;*
- *All required tasks are scheduled;*
- *At least one of the prerequisites for all required tasks are scheduled.*

3 Scheduler Implementation

To study the role of optimal distributed computing in our mission concepts, we implemented a scheduler which uses a simple state space search to satisfy Problem 1.

We use a simple solution-space search. Conceptually, a priority queue of solutions is maintained, sorted by cost. The set of acceptable end states are those which contain all the required tasks. The starting state is an empty schedule. At each iteration, a new, partial solution is constructed by adding a new task to one of the servers. If the requisite data products were not previously calculated on that server, then the solution is first augmented with communication tasks to gather the missing pre-requisites. The cost of the communication task depends on the transmission rate between the agents, and the earliest time that the agents can communicate. The agent which ensures earliest arrival time is chosen for each prerequisite data product. Thus, at each iteration either the solution is augmented by one task, or by a set of transmissions to retrieve missing data followed by the task itself. During search, a list of feasible solutions is kept for each reward value. If the resulting solution contains all required tasks, it is stored, indexed by reward. The search continues until the priority queue contains only solutions exceeding the maximum cost. Then, the maximum reward solution is returned.

The performance of the implemented scheduler is suitable for trade studies and ground-side assignment of computing duties. However, since we can solve the optimal distribution of tasks a-priori given a communications regime, it is simple to provide an onboard scheduler as a lookup table of pre-verified assignments.

In what follows we describe three scenarios where scheduling shared computing resources is key and would have impact. We use agent's tasks taken from literature and from future exploration missions to Mars.

4 Scenario Descriptions

Given the problem description and scheduler implementation from previous sections, we now describe the mission scenarios we consider. The scenarios were chosen to be realistic enough for meaningful analysis, and to stress different aspects of the computation and communication scheduling.

The costs for transmissions vary by data product size and transmission speeds (e.g., the contact graph data rates). Thus, we vary the data rates and maximum time to explore the trade space of solutions. The resulting set of solutions could be pre-calculated for quick look-up in a real mission if the device was too resource-constrained to run a full scheduler. However, we leave the onboard scheduler implementation to future work and instead explore the "tipping points"

between schedules and data rate thresholds at which interesting transitions between scheduler regimes occur.

In what follows we describe three scenarios. For each scenario we determine which set of agent capabilities is relevant, and compose them into a software network. For two of them we provide initial results of simulations, which illuminate the benefits of a MOSAIC-like architecture. The last scenarios illustrate more complex missions in which the architecture would provide a promising impact in future work.

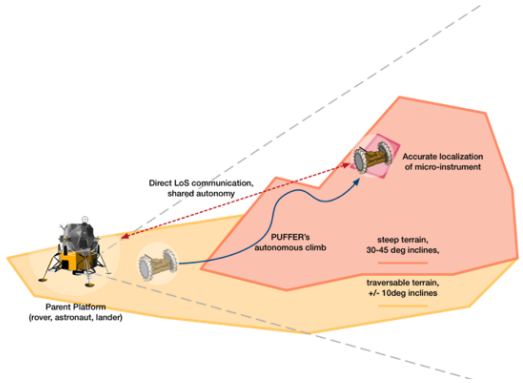


Figure 3: The "assisted drive" scenario.

4.1 Mars Drives

The first conceptual mission (Figure 3) is based on a single PUFFER combined with a parent platform (e.g., base station or flagship rover) to accurately place a PUFFER's instrument (microimager) on a terrain feature. This operation occurs within the parent platform's direct communication and sensing line-of-sight (LoS). PUFFER must be capable of autonomously navigating the environment homing in on the feature. It may leverage the better computation capabilities of the parent platform, as well as its sensors that offer a more advantageous perspective of the drive to improve its placement accuracy.

Each PUFFER is equipped with two STM32F4 microprocessors clocked at 180MHz and 168MHz with 256KB and 192KB of SRAM (Static Random Access Memory). Current versions of PUFFER utilize a Bluetooth radio with up to 2.1 Mbit/s data rates at approximately 1 W. Future versions of PUFFER may use a mesh radio, such as ZigBee, with data rates up to 250 kbits/s with approximately 100 mW power draw.

The parent platform, representing either a lander or rover, would include more significant computational resources, such as the HPSC. It would also have more power (e.g., MSL's radioisotope thermoelectric generator produces 2.5 kWh), and the communication equipment to communicate with an orbiter or directly to Earth (e.g., MSL has X-band for direct communication with Earth at 32kbit/s with 15 W, and UHF for communication to the orbiter at 2Mbit/s with 9 W) (Edwards et al. 2014).

We assume the parent platform can image the surrounding environment and locate the puffer to provide terrain-relative localization. We also assume the PUFFER can estimate it's own position using visual odometery (VO) and inertial measurements. We assume onboard state estimation using VO requires an image from the PUFFER's onboard camera system.

Since the PUFFER is equipped with a small scientific instrument, we assume that the puffer can acquire measurements from the instrument during its drive, but that this requires time, such as focusing, deploying, and pre-processing an image from a microscope. Thus, in process of navigating to its destination, a PUFFER has to sense the environment, plan its path and act (dispatch and execute low level tasks). During that process, a PUFFER might choose to perform science (microimager) and transmit the science data product to the lander for further use.

Figure 4a shows a data flow diagram to represent the software network associated with the aforementioned processes. The diagrams model the options available to the PUFFER to execute and share tasks in this scenario. Depending on the bandwidth and contact graph, the vehicle might choose to request the lander to take a long range image, localize the vehicle, perform path planning and then send the plan back to the PUFFER to execute the plan. That would potentially allow the PUFFER to use the spare time to take a microscope image and send it to the lander to archive it for further data fusion. The PUFFER might choose, as an alternative, to take an image from its camera system, use visual odometry to localize, and perform path planning all onboard; however, given that VO is less accurate than the terrain-relative localization from the lander, the PUFFER would have a higher uncertainty level about its position which would be carried to path planning.

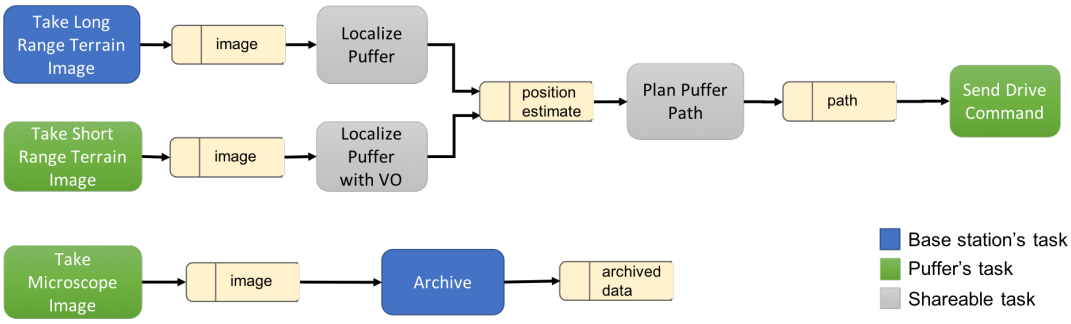
If both images (from lander and PUFFER) are taken and both localization processed are performed, the resulting position estimate is more accurate and so is the resulting trajectory from the path planning process. Figure 4a illustrates the tasks that can be shared and executed either onboard the lander or the PUFFER. Specifically, localization tasks and path planning are example of computational capabilities that can be scheduled and placed either onboard the lander or the PUFFER itself. Colored tasks mark the required vehicle for those capabilities.

We analyzed this software network for a variety of time limits and bandwidths between parent and PUFFER. The analysis is summarized in Table 4b. Figure 4c shows an example activity timeline for the puffer and base station from one of the resulting regimes. We find that high bandwidths are required to show preference towards off-board computing, at least for this scenario.

As shown in Table 4b, the long delays of taking microscopic images can be offset by requesting computational aid from the base state for planning paths. Alternatively, optimizations to program runtimes could have greater impact than bandwidth increases. Analyzing the *sensitivity* of these scheduler regimes with respect to runtimes, environment such as bandwidth distributions, and hardware choices is a key future direction to enable quick hardware and mission trade studies for distributed systems.

4.2 Mars 2020 Assisted Drive

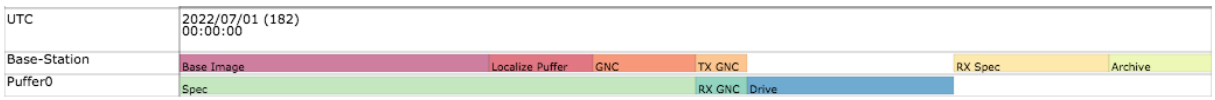
Note, the single-PUFFER scenario closely mimics the Mars 2020 mission with only minor changes. One defining feature of Mars Sample Return mission concepts is the likliihood of re-visiting the same area with subsequent launches to fetch, retrieve, and launch the samples (Mattingly and May 2011). If an on-site computing asset were available to multiple rovers in the area, they could make use of it for



(a) Data flow diagram representing software network for the single PUFFER scenario.

b/w (Mbps)	Base Cam	Puffer Cam	VO	Locate Puffer	GNC	Drive	Microscope	Store Img
≤ 0.05	Base	PUFFER	PUFFER	Base	PUFFER	PUFFER	N/A	N/A
$(0.05, .5]$	Base	PUFFER	PUFFER	Base	Base	PUFFER	N/A	N/A
$(0.5, 2.5)$	Base	N/A	N/A	Base	Base	PUFFER	PUFFER	Base
≥ 2.5	Base	PUFFER	Base	Base	Base	PUFFER	PUFFER	Base

(b) Distributed processing regimes for a single PUFFER and base station.



(c) Example activity timeline for the base station and puffer for a resulting regime.

Figure 4: Single-PUFFER scenario. The software network (4a) was analyzed as a function of bandwidth between the base station and rover to produce different processing regimes (4b. The rate of data transfer between the two uniquely determines what processes are possible, and where they are executed. Each data point is a timeline as shown in (4c. The roll-up shows aggregation of thousands of timelines produced by a solution-space search routine.

off-loading their required engineering tasks, in order to take advantage of opportunistic science processing and sensing. Thus, the assisting asset(s) could provide an "infrastructure upgrade" and could remain on-site, providing communication, computation, and data analysis services for all subsequent phases of the campaign. An interesting direction for future research would be to identify the requirements of such an asset. The asset could be embedded in a CubeSat network, and "piggy back" on the 2020 launch, similar to the MarCO CubeSats (Hodges et al. 2016). Alternatively, it could be embedded in the "skycrane" lander and dropped during the "flyaway" phase (Korzun et al. 2010; Sell et al. 2013). Finally, it could be a tethered balloon configuration (Kerzhanovich et al. 2004).

To explore any potential benefit, we next consider a strategic drive campaign by a Mars 2020 rover. In this case, we used information about the intended Mars 2020 drive pipeline from a talk given by Richard Rieber (Rieber 2017). The Mars 2020 conceptual path-planning pipeline, presented in (Rieber 2017) is simplified for our use in Figure 5a. The randomized time associated with Select Path is understandable given the mission analysis from (Ono et al. 2015). This data used in simulation is adapted from (Ono et al. 2016).

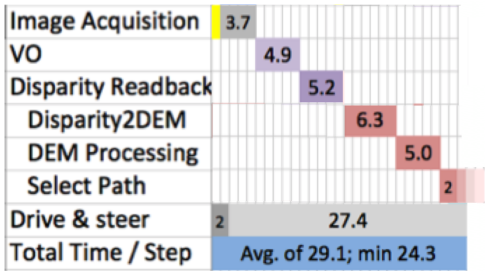
We created the model software network for Mars 2020 illustrated in Figure 5b. The required tasks are constructed to model the timings given in Figure 5a. From (Ono et al. 2016), we also included the ability for the rover to use imagers to classify the terrain, but only as an optional algorithm, since the current Mars 2020 pipeline does not include it.

To model the terrain in our simulations, we use terrain data classified from HiRISE imagery from (Ono et al. 2016). Multiple terrain types are grouped into different classes or as obstacles (terrain that cannot be traversed). We do not currently take slope into account, therefore we model the velocity of a rover in a given terrain class based on the average speed over multiple slopes for that classification.

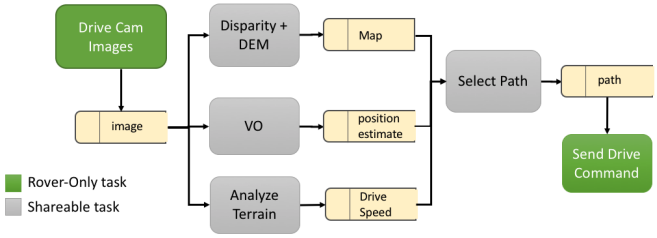
In order to model the different fidelity of data obtained in orbit and on the ground by the rover, we assume certain terrain types as unknown. When a rover is in an unknown terrain type, it will move at the velocity of the real terrain class; however, it will plan a path assuming a terrain with the fastest traverse velocity. Nevertheless, if a rover is able to perform terrain classification, we assume it will be able to correctly classify the terrain within a given radius.

Repeating the analysis of the software network produced the data shown in Table 6a. From this analysis, we isolated four operating regimes for the rover. In the first regime, the rover has no access to the assisting resource (regime 0). Regimes 1-4 represent increasing bandwidth, and therefore increasing savings from assisted computation. To reveal the *strategic* benefits of these computational regimes, we simulate the four rover regimes across a Mars-like strategic drive.

To test the different communication and computation regimes, simulations for 4 different regimes were run on 3 different terrain subsections 10 times each (resulting in 30 total runs) using stochastic durations for the path planning and terrain analysis activities. The assumed stochastic activity times are shown in Table 1. It is further assumed that the



(a) Simplified model of Mars 2020 path planning.



(b) A corresponding software network.

Figure 5: A model for the timing of Mars 2020 as discussed. The Select-path task is modelled as a random process taking a minimum of 2 seconds, but widely varying. The over-runs associated with any runtime longer than 30 seconds is the primary contributor to lost drive distance. The secondary contributor was a lack of terrain awareness, caused by insufficient processing power to run onboard terrain analysis.

duration to communicate the data to the balloon is approximately 3 seconds, and that the duration to communicate a response back to the rover is approximately 1 second. The distance between the start and end points for each traversal was approximately 93 meters.

Table 1: Duration of activities in seconds on-board the rover and on the balloon.

	On-Board	Balloon
Path Planning	$\mathcal{N}(8,4)$	$\mathcal{N}(0.5,0.0001)$
Terrain Analysis	$\mathcal{N}(4,4)$	$\mathcal{N}(0.5,0)$

The baseline regime is Regime 1, where the rover performs all path planning on-board and does not perform any terrain analysis. In Regime 2, the rover sends data to a balloon where the path planning algorithm is performed and the results sent back to the rover. Regime 3 is the same as Regime 2, except that with the extra time, the rover performs terrain analysis on-board, which can be used for the next planning cycle. In Regime 4, terrain analysis is also performed on the balloon and the results communicated back to the rover.

Figure 6b shows an example of the different paths that are taken for the different regimes when some of the terrain is unknown without terrain analysis. The yellow terrain requires terrain analysis to be identified and is also slower to traverse. From this example, it is shown that with the terrain identification knowledge, Regime 3 and Regime 4 are able to come up with more efficient paths.

Since it is assumed that the rover must operate on a fixed 30 second cycle, if the path planning and/or terrain analysis are not completed within the allotted 8 seconds, an overrun will occur, causing the rover to stop until computation is completed. The distribution of percentage of overruns are shown as box plots in Figure 6d. As expected, Regime 2 and Regime 4 result in no overruns, whereas Regime 1 and Regime 3 have overruns around 50% of the time.

Another metric for the improvement of the rover performance is in the time it takes to traverse a terrain. Figure 6c shows the time to traverse a terrain for each regime compared to the baseline (Regime 1). From these results, it is shown that being able to perform terrain analysis, and therefore being able to plan a path with better terrain knowledge, improves the time to travel between two points.

We note a measurable increase in strategic drive efficiency using this limited study technique. Future work can focus on a more realistic terrain model, including that of the intended landing site. In addition, we can more realistically model the communication network. Intermittent loss of connectivity and varying data rates are significant impediments to this approach over long dries. Finally, modelling multiple assets would involve not only competing for the computational resources, but forwarding terrain classifications and drive rates between rovers.

4.3 Cooperative Exploration

The next conceptual mission (Figure 7) is based on multiple PUFFERs cooperatively (i.e., their autonomous operations are coordinated by sharing information) expanding science and exploration footprints into areas not within direct line-of-sight of the parent platform. The team of PUFFERs will maintain a communication network while exploring an environment with limited direct line of site (e.g., rubble fields, caves, lava tubes).

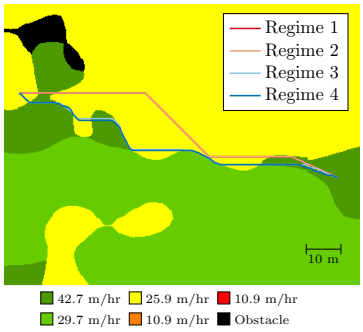
We assume the PUFFERS are exploring a distributed, but spatially-correlated phenomena, such as water moisture levels. We model the sampling and estimation on a similar terrestrial process used in farms(Tokekar et al. 2016). The point samples of moisture levels are gathered by spectroscopy or dipole measurements, and are incorporated into a spatial-estimation technique called Kriging (Brdossy and Lehmann 1998). Kriging is computationally expensive, and requires storage of all measurements. Therefore, it is not suitable for computationally-constrained devices like PUFFERS, but can be performed on the base station, orbiter, or on Earth.

In this scenario, each PUFFER operates under the same condition and software network as those used in the single vehicle scenario (Figure 4a), except that herein the lander becomes a shared resource for computation requests. Moreover, the team of PUFFERs provides a larger mesh-based communication network, allowing data to be sent across vehicles to reach the lander.

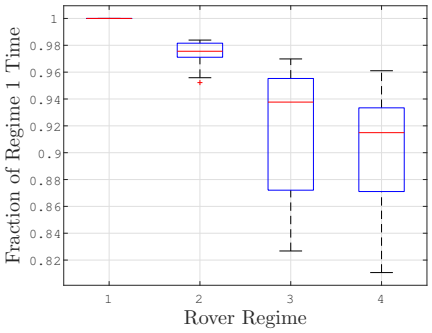
As before, a PUFFER can perform localization and path planning onboard, request the lander for support on those tasks, or even both while navigating the environment. In those cases, the computation sharing has to be coordinated among the vehicles since the lander has limited computational resource. In such coordination, PUFFERs can reason

b/w (Mbps)	Time	Image	Mapping	Extra Observations	Plan Path	Confirm / Drive	SPOC-lite
$(0-.1]$	27	Rover	Rover	N/A	Rover	Rover	N/A
$(0.1-.3]$	29.3	Rover	Rover	N/A	Assist	Rover	Rover
$(0.3-1]$	$(29.7-28.2]$	Rover	Rover	Rover	Assist	Rover	Rover
$(1,100]$	$(27.3-15.3]$	Rover	Assist	Assist	Assist	Rover	Rover

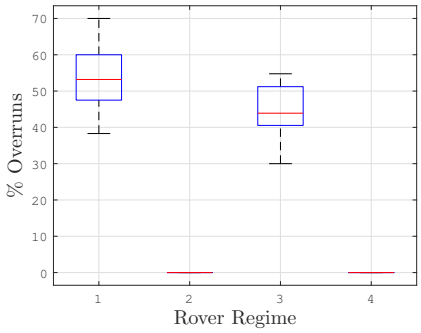
(a) A Mars 2020 rover adaptation of assisted drive. The adaptation was made using the pipeline information given in Figure 5a.



(b) Example paths



(c) Time to traverse waypoints per regime



(d) Overruns per traverse for each regime

Figure 6: Effect of computing regimes on a Mars 2020-like mission. 6b shows the path choices. The main effects of addition computation assistance is reduced planner overrun and better terrain classification, resulting in more efficient paths, as shown in 6c and 6d. Terrain types are designated as different colors and the darker terrain (darkest except for black) can only be identified using terrain analysis.

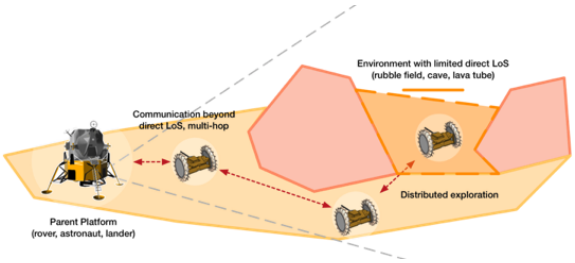


Figure 7: Multi-Robot Scenario 2

about routing their data products to the lander. For example, it might detect that a direct communication link to the lander is poor due to the current terrain features, but routing data through one of the other PUFFERS would work better. That would allow the scheduler to potentially add both microscope image and archive tasks to the regime, along with transferring the data through the vehicles network.

This scenario presents significant challenges to distributed computation because of the combination of roles a PUFFER may take. First, it may be purely sensory, taking images and then moving while sending those images to the base station. Second, it may be able to position itself as a relay node, spending all its time ferrying data between other assets. Alternatively, it could be a combination of the two, depending on its location, the plans of assets around it, and the motion around intervening terrain which may affect bandwidth. Finally, the motion planning problem in this context is critical. How are sample locations chosen for the PUFFERs? How does the motion and location of the PUFFERs affect data rate, and can paths be chosen to maximize information flow? These questions are good directions for future work.

5 Conclusion

In this paper we described the MOSAIC concept for Mars exploration in which simultaneous scheduling of computation, communication, and caching of data across different networked assets becomes increasingly essential. We presented a series of scenarios to illustrate how MOSAIC networks can impact science utility, vehicle performance and would enable an optimal distribution of computational loads, specially in multi-asset scenarios - a natural progression of future missions to Mars and other planets.

The cooperative exploration scenario in Section 4.3 represents our major next hurdle. A comprehensive solution would include role assignment (relays versus sensors), position and path assignments to maintain connectivity, and response to changing communication networks, including momentary breakage of links to gain greater sensing data. We will proceed first with role assignments for data routing.

The preliminary study of optimal processing distribution is useful as feedback into hardware design. The methods of this paper can be used to optimize the hardware of the PUFFER design, or design communication networks for future Mars exploration missions. In this direction, determining the "tipping points" between different processing regimes is most important. The differences in efficiency between regimes can be very large. A schedule sensitivity analysis is required to determine the optimal schedule's response to perturbations to e.g., bandwidth. We have conducted this analysis by using a "brute-force" search routine, but producing analytical and algorithmic results which are quick are more capable are a primary next step for research. We expect this analysis will fold nicely into a framework similar to (Herzig et al. 2017) which provides a hardware-space expansion for designing multi-asset missions.

The initial results and envisioned scenarios described in this paper brings interesting next steps and promising re-

search efforts in the MOSAIC project. We will study in more depth the multi-vehicle scenarios presented in this paper and identify the key algorithmic requirements for those cases. In these cases we will investigate on different scheduling techniques and formalisms that could be utilized onboard the assets to allocate computation load, considering vehicle with both low and high CPU capabilities, and manage connectivity fluctuations. Our framework is designed to be responsive to loss of connectivity by re-scheduling tasks based on a new communications graph using a set of pre-verified distribution of tasks. In particular, we have studied the change in optimal computing distribution due to bandwidth fluctuations, but more research is necessary to fully evaluate risk of connectivity variations and provide an onboard scheduler which can accommodate unlikely but impactful changes. Moreover, we will also incorporate the multi-agent coordination aspect to the target scenarios, in which agents have to negotiate the distribution of computation, data flow and utilization of resources. Agents might have different utility functions and goals that will add an interesting element to our network problem.

Finally, uncertainty and risk management is a key aspect of realistic assets networks for planetary exploration. Several aspects of exploration mission have uncertainty and can potentially be represented with stochastic models, such as task outcome and duration, vehicle failure, connectivity, bandwidth variations, and others. One promising research avenue is to incorporate probabilistic planning and scheduling approaches (Santana et al. 2016) to the computation sharing problem, as well risk-bounded techniques to provide guarantees that the network and the vehicles are able to operate within user specified bounds.

Acknowledgements

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

References

[Brdossy and Lehmann 1998] Brdossy, A., and Lehmann, W. 1998. Spatial distribution of soil moisture in a small catchment. part 1: geostatistical analysis. *Journal of Hydrology* 206(1):1 – 15.

[Doyle et al. 2013] Doyle, R.; Some, R.; Powell, W.; Mounce, G.; Goforth, M.; Horan, S.; and Lowry, M. 2013. High performance spaceflight computing (hpsc) next-generation space processor (ngsp): a joint investment of nasa and afrl. In *Proceedings of the Workshop on Spacecraft Flight Software*.

[Edwards et al. 2014] Edwards, C. D.; Barela, P. R.; Gladden, R. E.; Lee, C. H.; and Paula, R. D. 2014. Replenishing the mars relay network. In *2014 IEEE Aerospace Conference*, 1–13.

[Herzig et al. 2017] Herzig, S. J. I.; Mandutianu, S.; Kim, H.; Hernandez, S.; and Imken, T. 2017. Model-transformation-based computational design synthesis for mission architecture optimization. In *2017 IEEE Aerospace Conference*, 1–15.

[Hodges et al. 2016] Hodges, R. E.; Chahat, N. E.; Hoppe, D. J.; and Vacchione, J. D. 2016. The mars cube one deployable high gain antenna. In *2016 IEEE International Symposium on Antennas and Propagation (APSURSI)*, 1533–1534.

[Karras et al. 2017] Karras, J. T.; Fuller, C. L.; Carpenter, K. C.; Buscicchio, A.; McKeeby, D.; Norman, C. J.; Parcheta, C. E.; Davydychev, I.; and Fearing, R. S. 2017. Pop-up mars rover with textile-enhanced rigid-flex pcb body. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 5459–5466. IEEE.

[Kerzhanovich et al. 2004] Kerzhanovich, V.; Cutts, J.; Cooper, H.; Hall, J.; McDonald, B.; Pauken, M.; White, C.; Yavrouian, A.; Castano, A.; Cathey, H.; Fairbrother, D.; Smith, I.; Shreves, C.; Lachenmeier, T.; Rainwater, E.; and Smith, M. 2004. Breakthrough in mars balloon technology. *Advances in Space Research* 33(10):1836 – 1841. The Next Generation in Scientific Ballooning.

[Korzun et al. 2010] Korzun, A. M.; Dubos, G. F.; Iwata, C. K.; Stahl, B. A.; and Quicksall, J. J. 2010. A concept for the entry, descent, and landing of high-mass payloads at mars. *Acta Astronautica* 66(7):1146 – 1159.

[Mattingly and May 2011] Mattingly, R., and May, L. 2011. Mars sample return as a campaign. In *2011 Aerospace Conference*, 1–13.

[Mounce et al. 2016] Mounce, G.; Lyke, J.; Horan, S.; Powell, W.; Doyle, R.; and Some, R. 2016. Chiplet based approach for heterogeneous processing and packaging architectures. In *2016 IEEE Aerospace Conference*, 1–12.

[Ono et al. 2015] Ono, M.; Fuchs, T. J.; Steffy, A.; Maimone, M.; and Yen, J. 2015. Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In *Aerospace Conference, 2015 IEEE*, 1–10. IEEE.

[Ono et al. 2016] Ono, M.; Rothrock, B.; Almeida, E.; Ansar, A.; Otero, R.; Huertas, A.; and Heverly, M. 2016. Data-driven surface traversability analysis for mars 2020 landing site selection. In *Aerospace Conference, 2016 IEEE*, 1–12. IEEE.

[Powell et al. 2011] Powell, W.; Johnson, M.; Some, R.; Wilmot, J.; Gostelow, K.; Reeves, G.; and Doyle, R. 2011. Enabling future robotic missions with multicore processors. In *Infotech@ Aerospace 2011*. 1447.

[Rieber 2017] Rieber, R. R. 2017. Designing for a martian road trip: The mobility system for mars-2020. Keynote Talk: Mars Forum (URS: URS270204, CL17-5707).

[Santana et al. 2016] Santana, P.; Vaquero, T.; Toledo, C.; Wang, A.; Fang, C.; and Williams, B. 2016. Paris: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

[Sell et al. 2013] Sell, S.; Chen, A.; Davis, J.; San Martin, M.; Serricchio, F.; and Singh, G. 2013. Powered flight design and reconstructed performance summary for the mars science laboratory mission. Technical report, Jet Propulsion Laboratory, National Aeronautics and Space Administration.

[Tokekar et al. 2016] Tokekar, P.; Hook, J. V.; Mulla, D.; and Isler, V. 2016. Sensor planning for a symbiotic uav and ugv system for precision agriculture. *IEEE Transactions on Robotics* 32(6):1498–1511.

[Wyatt et al. 2017] Wyatt, E. J.; Belov, K.; Burleigh, S.; Castillo-Rogez, J.; Chien, S.; Clare, L.; and Lazio, J. 2017. New capabilities for deep space robotic exploration enabled by disruption tolerant networking. In *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 1–6.