

# Multi-Robot Distributed Computing as a Resource for Mars Exploration

Joshua Vander Hook\*, Tiago Vaquero, Martina Troesch,  
Federico Rossi, Joshua Schoolcraft, Saptarshi Bandyopadhyay,  
Jean-Pierre de la Croix, 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. They are: High Performance Spaceflight Computing (HPSC) Delay Tolerant Networking (DTN) and 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. As such, we study the concept of Mars on-site shared analysis, information, and communication (MOSAIC) for Mars exploration. In this distributed computation regime, the network of heterogeneous robots uses communication to lend computation assistance when required, directing resources to where it is needed. The key algorithmic problem associated with MOSAIC networks is simultaneous scheduling of computation, communication, and caching of data, which we illustrate using two Mars exploration scenarios.

## 1 Introduction

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 [2,13,10]. 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,

---

\* Corresponding Author, [hook@jpl.nasa.gov](mailto:hook@jpl.nasa.gov)

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Government sponsorship acknowledged. Copyright © 2018, California Institute of Technology.

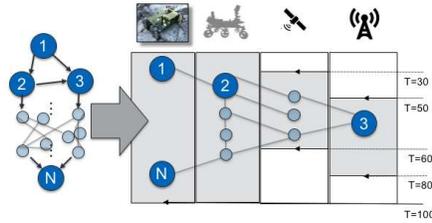


Fig. 1: Illustrative MOSAIC scenario. A set of processing and data-driven tasks (left as dependency graph) must be mapped to multiple assets with heterogeneous computing, communication, and energy capacities. Each asset is also available over a fixed time window due to terrain effects or orbital parameters.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

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. 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 [18].

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 parent 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 potential future systems being considered for development are the Mars Helicopter, and the “PUFFER” rover (Pop-Up Flat-Folding Explorer Robots) [6].

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. We are motivated by

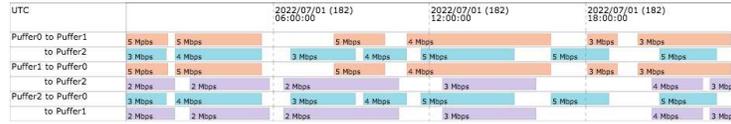


Fig. 2: Contact graph for 3 agents showing times and bandwidths available

the heterogeneity of these systems to study how the choice in computation and communication capabilities affects small, resource-constrained, high-risk “edge” devices. In particular how, by optimizing data flows and processing assignments among all the devices, the mission efficiency can be increased. 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 presents a high-level trade study of the benefit of distributed computation and data sharing for plausible Mars exploration scenarios. In Section 3 we propose an integer-programming approach which can identify how the computational load can be distributed over the network. We next present our problem more formally.

## 2 Problem Description

In this section, we define our data communication and processing work flow to represent the routine tasks that make up mission objectives. We consider robotic agents, which each have an interdependent series of tasks, on-board computing, sensing, and a time-varying communication link to some or all other agents.

**Agents:** Let there be  $N \in \mathbb{Z}^+$  agents in the network, where  $\mathbb{Z}^+$  denotes positive integers. The robot agents are denoted by  $A_1, A_2, \dots, A_N$ . Each agent has known on-board processing, memory, and communication links.

**Tasks and Software Network:** 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. Program outputs are the same irrespective of the agent doing the computing (or are just as useful). If an agent has two or more *dissimilar* processing units, they can be modelled as coincident 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 also can produce data products. Data products for task  $T$  are denoted  $d(T)$ . The size of the data products are known a-priori as  $s(T)$  for task  $T$ . The set of required tasks is denoted  $\mathbb{R} \subseteq \mathbb{T}$ . Optional tasks have a reward score ( $r(T)$ ).

Let  $P_T$  be a set of predecessor tasks for  $T$ . Then  $j \in P_T$  means that task  $T$  depends on the data output of task  $j$ . A task may have multiple prerequisite sets,

one of which must be satisfied entirely. To execute a task, 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.

**Assumptions:** The software network  $SN$  does not have any cycles.

A *solution* is a mapping of tasks to servers (agents) and start-times denoted  $\mathbb{S} : i \rightarrow (A_j, t)$  where  $j \in [1, \dots, N]$  and  $t \geq 0$ . Each agent’s computing schedule in a solution is denoted  $\mathbb{S}_i = j \rightarrow_i(t)$ , and has cost equal to the time required to complete the last task in the agent’s queue,  $C(\mathbb{S}) = \max_i C(\mathbb{S}_i)$  where  $C(\mathbb{S}_i) = \max_j \mathbb{S}_i(j) + C_i^t(j)$ .

**Communication Graph:** A key feature of DTN-based networking is Contact Graph Routing (CGR) [18]. CGR takes into account predictable link schedules and bandwidth limits to automate data delivery and optimize the use of network resources. 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. An example time line representation for 3 agents with available bandwidths can be seen in Figure 2.

Links have a time-varying data rate from 0 (not connected) to  $\infty$  (communicating to self), denoted by  $r_{ij}(t)$  for the rate from  $A_i$  to  $A_j$  at time  $t$ . At any time  $k$ , let  $\mathcal{G}_k$  be the graph representing the set of agents it can send to or receive from, vertices  $\mathcal{V} = \{1, \dots, N\}$  and the directed edges  $\mathcal{E}_k$  along which communication is possible. The task of communicating the data product  $d(T)$  from  $A_i$  to  $A_j$  at time  $t$  requires time  $C_{ij}^t(T) \propto s(T)/r_{ij}(t)$  for *both* agents.

**Assumptions:** Agents take 0 time to communicate the solution to themselves, except in the case of multiple on-board processors, which are modelled as coincident processors linked by a given data rate.

*Problem 1 (Distributed Computation).* Given a set of tasks modelled as a software network  $SN$ , a list of computational agents  $A_i$   $i \in [1 \dots N]$ , a contact graph  $CG$ , and a maximum schedule length  $C^*$ , 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: (1) item The maximum overall computation time,  $C(\mathbb{S}) = \max_j C(\mathbb{S}_j)$  is no more than  $C^*$ ; (2) All required tasks are scheduled; (3) At least one of the prerequisites for all required tasks are scheduled; (4) The reward due to optional tasks  $\sum_{T \in \mathbb{T} \setminus \mathbb{R}} r(T) 1_{\{T \text{ is scheduled}\}}$  is maximized.

### 3 Scheduler Implementation

To study the role of optimal distributed computing in our mission concepts, we cast Problem 1 as an integer linear program (ILP). We consider a discrete-time approximation of the problem with a time horizon of  $C_d^*$  time steps corresponding to the maximum schedule length  $C^*$ . The optimization variables are:  $X$ , a set

of Boolean variables of size  $N \cdot M \cdot C_d^*$ .  $X(i, T, c)$ , with a true entry iff agent  $A_i$  starts computing task  $T$  at time  $c$ .  $D$ , a set of Boolean variables of size  $N \cdot M \cdot C_d^*$ .  $D(i, T, c)$ , with a true entry iff agent  $A_i$  has stored the data products  $d(T)$  of task  $T$  at time  $c$ .  $C$ , a set of Boolean variables of size  $N^2 \cdot M \cdot C_d^*$ .  $C(i, j, T, c)$ , with a true entry iff agent  $A_i$  communicates the product of task  $T$  to agent  $A_j$  at time  $c$ . The optimization objective is to maximize the sum of the rewards corresponding to completed optional tasks, i.e.

$$\sum_{i=1}^N \sum_{T \in \mathbb{R}} \sum_{c=1}^{C_d^* - C_i^t(T)} r(T) X(i, T, c) \quad (1)$$

The problem constraints are captured by the following set of equations:

$$\sum_{i=1}^N \sum_{c=1}^{C_d^* - C_i^t(T)} X(i, T, c) = 1 \quad \forall T \in \mathbb{T} \setminus \mathbb{R} \quad (2a)$$

$$\sum_{i=1}^N \sum_{c=1}^{C_d^* - C_i^t(m)} X(i, T, c) \leq 1 \quad \forall T \in \mathbb{R} \quad (2b)$$

$$X(i, T, c) \leq \sum_{L \in P_T} D(i, L, c) \quad \forall i \in [1, \dots, N], T \in [1, \dots, M], c \in [1, \dots, C_d^*] \quad (2c)$$

$$\sum_{T=1}^M \left[ \sum_{j=1}^N (C(i, j, T, c) + C(j, i, T, c)) + \sum_{\tau=\max(1, c-C_i^t(T))}^c X(i, T, \tau) \right] \leq 1 \quad (2d)$$

$$\forall i \in [1, \dots, N], c \in [1, \dots, C_d^*]$$

$$D(i, T, c+1) - D(i, T, c) \leq \sum_{\tau=1}^c \sum_{j=1}^N \frac{r_{ji}(c)}{s(T)} C(j, i, T, c) + \sum_{\tau=1}^{c-C_i^t(T)} X(i, T, \tau) \quad (2e)$$

$$\forall i \in [1, \dots, N], T \in [1, \dots, M], c \in [1, \dots, C_d^* - 1]$$

$$C(i, j, T, c) \leq D(i, T, c) \quad \forall i, j \in [1, \dots, N], T \in [1, \dots, M], c \in [1, \dots, T] \quad (2f)$$

$$D(i, T, 1) = 0 \quad \forall i \in [1, \dots, N], T \in [1, \dots, M] \quad (2g)$$

Equations (2a) and (2b) ensure that all required tasks are performed and optional tasks are performed at most once. Equation (2c) requires that agents only complete a task if they have access to the data products of its predecessor tasks. Equation (2d) captures the agents' limited computation resources by ensuring that each agent either performs a task or communicates at any given time. Equation (2e) ensures that agents learn the content of a task's data products only if they (i) receive such information from other agents or (ii) complete the task themselves. Equation (2f) ensures that agent only communicate a data product if they have the data product themselves. Finally, Equation (2g) models the fact that data products are initially unknown to all agents.

The ILP has  $N^2MT + 2NMT$  Boolean variables and  $M(N(3T-1) + N) + NT$  constraints; instances with dozens of agents and tasks can be readily solved by state-of-the-art ILP solvers. We do not propose this as a scheduler to be used by a robotic heterogeneous swarm. It is intended to be a reliable, heuristic-free way to compute optimal computing distribution for a given scenario so that we

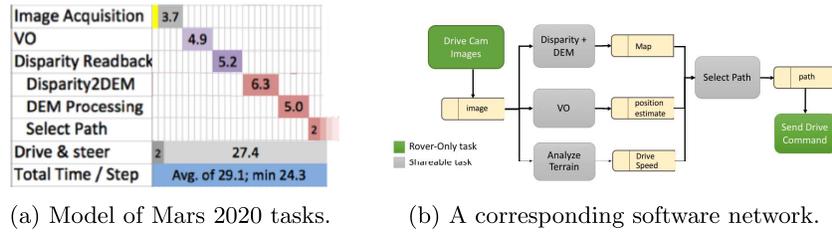


Fig. 3: A model for the timing of Mars 2020 rover path planning, as discussed.

can do a cost-benefit analysis. However, we will see that the results of this study could be used by agents as a “playbook”, as discussed next.

## 4 Scenario Descriptions

The scenarios now presented were chosen to be realistic enough for meaningful analysis, and to stress different aspects of the computation and communication scheduling. For both scenarios, we derive optimal computing schedules given a time-varying contact graph (connectivity and data rates for the agents). We vary the contact graph to find how the schedules change. The resulting set of pre-calculated solutions could be used for quick look-up in a real mission, as long as the robot has knowledge of the communication network topology and throughput. However, we leave a precise description and analysis of a distributed-swarm on-board scheduler to future work.

**Mars 2020 Assisted Drive** One defining feature of proposed Mars Sample Return mission concepts is the likelihood re-visiting the same area with subsequent launches to fetch, retrieve, and eventually launch soil samples for return to earth[9]. If an on-site computing asset were available to multiple rovers in the area, they could make use of it for 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<sup>1</sup> The asset could be embedded in a CubeSat network, and “piggy back” on the 2020 launch, similar to the MarCO CubeSats [5], be embedded in the “sky crane” lander and dropped during the “flyaway” phase [8,16], or could be a tethered balloon configuration [7].

Thus, we consider a strategic drive campaign by a Mars 2020 rover. We used information about the intended Mars 2020 drive planning pipeline from a talk given by Richard Rieber [14]. It is simplified for our use in Figure 3a, which shows the timings for each task, and waterfall dependencies on prior tasks. The key point of the pipeline is that all tasks must complete in less than 30 seconds to accommodate the strategic rate of progress for the mission. The randomized

<sup>1</sup> An interesting direction for future research would be to identify the requirements of such an asset.

time associated with Select Path is understandable given the mission analysis from [11]. From this, we created the model software network for Mars 2020 illustrated in Figure 3b. We also included an additional capability which has been evaluated for Mars 2020, Analyze Terrain. This capability, described in [12], allows estimation of drive speeds from forward-looking imagery.

The set of tasks are:

- **Image**- Take images of the surrounding terrain.
- **VO**- Visual Odometry to estimate location.
- **DEM**- Build from disparity (or update) a digital elevation map (DEM) to plan with. (This is shown as two steps in Figure 3a).
- **Select Path** - Plan a safe path using the localization estimate and DEM
- **Analyze Terrain** - Not shown in Figure 3a, yet included in [12] as a way to identify high-slip / high-slowdown terrains (plan paths around them).

We assumed that the three data products which are derived from the drive cam imagery (VO, DEM, and Drive Speeds), can be computed in parallel, and that all are useful for the path planner to determine optimal paths. We continued the assumption that all tasks must to operate within the thirty second time window, as that was dictated by strategic drive goals, safe speed constraints, and hardware limits.

To model the terrain in our simulations, we use terrain data classified from HiRISE imagery from [12]. 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; but it will plan a path assuming a terrain with the fastest traverse velocity. If a rover is able to perform **Analyze Terrain**, we assume it will be able to correctly classify the terrain within a given radius.

**Study Results:** We used the MILP scheduler to derive optimal operating regimes based on the available bandwidth between the Co-processor and rover Table 4a. 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. Simulations for 4 different regimes were run on 3 different HiRISE terrain subsections of 100 meters length, 10 times each (resulting in 30 total runs) using stochastic durations for the path planning and terrain analysis activities. Stochastic times for Path Planning and Terrain Analysis were assumed to be Gaussian. On-board Path Planning was  $\mathcal{N}(8, 4)$  versus HPSC-enabled parallel search with  $\mathcal{N}(0.5, 0.0001)$ . Similarly, Terrain Analysis was  $\mathcal{N}(4, 4)$  and  $\mathcal{N}(0.5, 0)$ .

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

b/w (Mbps)	Avg Time	Image	Mapping	Analyze Terrain	Plan Path
(0 - .1]	27	Rover	Rover	N/A	Rover
(0.1 - .3]	29.3	Rover	Rover	N/A	Assist
(0.3 - 1]	(29.7 - 28.2]	Rover	Rover	Rover	Assist
(1, 100]	(27.3 - 15.3]	Rover	Assist	Assist	Assist

(a) A Mars 2020 rover adaptation of assisted drive.

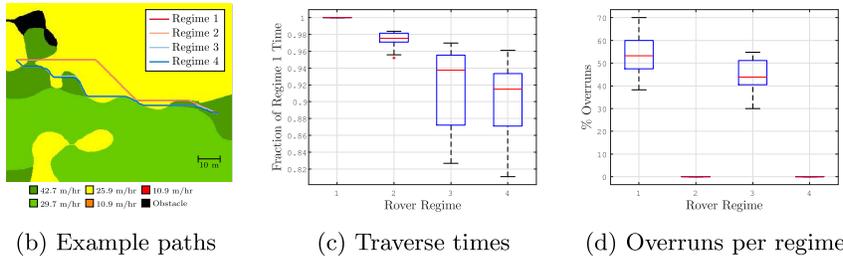


Fig. 4: Effect of assisted computing, as a function of bandwidth on a Mars 2020-like mission.

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 4b shows an example of the 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 4d. As expected, Regime 2 and Regime 4 result in no overruns, whereas Regime 1 and Regime 3 have (minor) overruns around 50% of the time.

Figure 4b 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 4c and 4d. Terrain types are designated as different colors and the darker terrain (darkest except for black) can only be identified using terrain analysis. Figure 4c shows the time to traverse a terrain for each regime compared to the baseline (Regime 1). 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. Intermittent loss of connectivity and varying data rates are interesting avenues for future research.

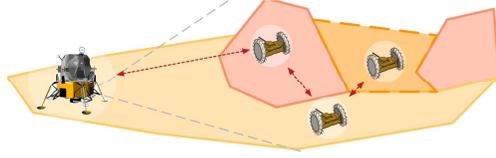


Fig. 5: Multi-Robot Scenario 2

**Cooperative Exploration** The second conceptual mission (Figure 5) 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 a parent platform (e.g., base station or flagship rover). 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).

Details of the current PUFFER design are given in [6]. We assume that the drive pipeline is similar to that of Mars 2020 (at a high level). 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.

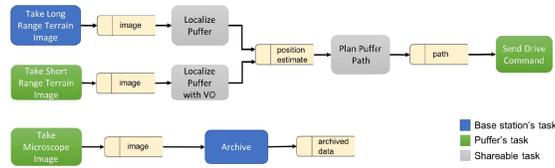
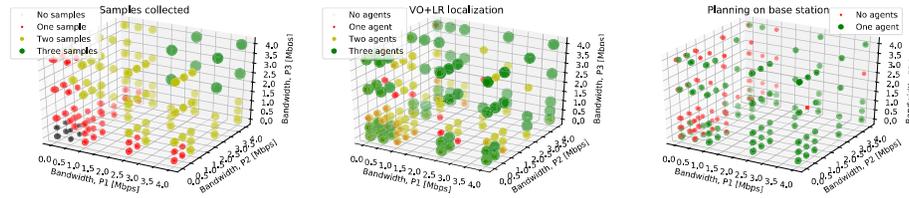


Fig. 6: Data flow diagram representing software network for each PUFFER in the cooperative exploration. Identical to Figure 3b except **Analyze Terrain** is now the science objective (**Microscope Image**), and must be archived on the base.

The PUFFERS are deployed from a lander or rover. As a stationary resource, it would include more significant computational resources, such as the HPSC. It would also be large enough to have more power generating capacity. To gain useful data from the PUFFERS, it must also have communication equipment to communicate with an orbiter or directly to Earth (See [3] for details).

The parent platform can image the surrounding environment to locate the puffer to provide terrain-relative localization. We assume on-board state estimation using VO requires an image from the PUFFER's onboard camera, and fusing of estimates from the parent (or other PUFFERS) is encouraged. If both images (from lander and PUFFER) are taken and both localization processed



(a) Overall number of samples collected by the PUFFERs. (b) PUFFERs using both odometry and terrain relative localization. (c) Number of PUFFERs who rely on the base station for path planning.

Fig. 7: Cooperative exploration scenario. The software network (6) was analyzed as a function of the bandwidths available between each pair of agents to produce different processing regimes, producing different distributions of capabilities, and increasing ability to perform high-reward activities, such as multi-sensor localization or additional science activities.

are performed, the resulting position estimate is more accurate and so is the resulting trajectory from the path planning process. 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 [17]. The point samples of moisture levels are gathered by spectroscopy or dipole measurements, and are incorporated into a spatial-estimation technique called Kriging [1]. Kriging is computationally expensive, and requires storage of all measurements — not suitable for computationally-constrained devices like PUFFERs. Figure 6 shows a data flow diagram to represent the software network associated with the aforementioned processes. 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.

**Study Results:** We analyzed this software network for a variety of time limits and bandwidths between parent and PUFFER. Unlike the Assisted Drive scenario with 4 notable regimes, we found the operating regimes were more sensitive to bandwidth given more agents, and therefore omit the full presentation in the interest of brevity. In the first, low bandwidth precludes cooperation and no additional science is gained. In the second, moderate bandwidth between base and rovers allows some images and improvements to localization by offloading planning and VO to the base station. In the third, the base station takes on most of the responsibility. The full set of regimes was calculated for finely-discretized bandwidth steps. As discussed, the lookup table could be used by the rovers to have an  $\mathcal{O}(1)$  distributed scheduler, but is intended in this study to inform hardware design points going forward.

The total benefit of this scenario is measured not by drive distance (as was the case with the Assisted Drive scenario), but by the number of science samples

taken, and the total localization accuracy of the robots during sampling. Figures 7a-7c show the number of samples collected, the number of agents who have access to accurate localization, and the number of agents who rely on the base station for planning in the case where PUFFERS can only communicate with the base station. This compact representation compresses the complexity of the 4-robot scheduling problem. The space of solutions (regimes) was too large to show as we did in Figure 4a.

## 5 Conclusion

We described the MOSAIC concept for Mars exploration in which scheduling of computation, communication, and caching of data across networked assets is shown to be beneficial. 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.

Based on the improved mission metrics, the methods of this paper can be used to optimize the hardware of the distributed missions, or design communication networks for future Mars exploration missions. Thus, determining the “tipping points” between different processing regimes is most important since the differences in efficiency between regimes can be very large. We expect this analysis will fold nicely into a framework similar to [4] which provides a hardware-space expansion for designing multi-asset missions.

A primary next step is to investigate different scheduling techniques that could be utilized onboard the assets to allocate computation load. Agents might have different utility functions and goals that will add an interesting element to our network problem. 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 [15] 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.

## References

1. Brdossy, A., Lehmann, W.: Spatial distribution of soil moisture in a small catchment. part 1: geostatistical analysis. *Journal of Hydrology* 206(1), 1 – 15 (1998)
2. Doyle, R., Some, R., Powell, W., Mounce, G., Goforth, M., Horan, S., Lowry, M.: High performance spaceflight computing (hpsc) next-generation space processor (ngsp): a joint investment of nasa and aflr. In: *Proceedings of the Workshop on Spacecraft Flight Software* (2013)
3. Edwards, C.D., Barela, P.R., Gladden, R.E., Lee, C.H., Paula, R.D.: Replenishing the mars relay network. In: *2014 IEEE Aerospace Conference*. pp. 1–13 (March 2014)

4. Herzig, S.J.I., Mandutianu, S., Kim, H., Hernandez, S., Imken, T.: Model-transformation-based computational design synthesis for mission architecture optimization. In: 2017 IEEE Aerospace Conference. pp. 1–15 (March 2017)
5. Hodges, R.E., Chahat, N.E., Hoppe, D.J., Vacchione, J.D.: The mars cube one deployable high gain antenna. In: 2016 IEEE International Symposium on Antennas and Propagation (APSURSI). pp. 1533–1534 (June 2016)
6. Karras, J.T., Fuller, C.L., Carpenter, K.C., Buscicchio, A., McKeeby, D., Norman, C.J., Parcheta, C.E., Davydychev, I., Fearing, R.S.: Pop-up mars rover with textile-enhanced rigid-flex pcb body. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on. pp. 5459–5466. IEEE (2017)
7. 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., Smith, M.: Breakthrough in mars balloon technology. *Advances in Space Research* 33(10), 1836 – 1841 (2004), <http://www.sciencedirect.com/science/article/pii/S0273117703011682>, the Next Generation in Scientific Ballooning
8. Korzun, A.M., Dubos, G.F., Iwata, C.K., Stahl, B.A., Quicksall, J.J.: A concept for the entry, descent, and landing of high-mass payloads at mars. *Acta Astronautica* 66(7), 1146 – 1159 (2010), <http://www.sciencedirect.com/science/article/pii/S0094576509004895>
9. Mattingly, R., May, L.: Mars sample return as a campaign. In: 2011 Aerospace Conference. pp. 1–13 (March 2011)
10. Mounce, G., Lyke, J., Horan, S., Powell, W., Doyle, R., Some, R.: Chiplet based approach for heterogeneous processing and packaging architectures. In: 2016 IEEE Aerospace Conference. pp. 1–12 (March 2016)
11. Ono, M., Fuchs, T.J., Steffy, A., Maimone, M., Yen, J.: Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In: Aerospace Conference, 2015 IEEE. pp. 1–10. IEEE (2015)
12. Ono, M., Rothrock, B., Almeida, E., Ansar, A., Otero, R., Huertas, A., Heverly, M.: Data-driven surface traversability analysis for mars 2020 landing site selection. In: Aerospace Conference, 2016 IEEE. pp. 1–12. IEEE (2016)
13. Powell, W., Johnson, M., Some, R., Wilmot, J., Gostelow, K., Reeves, G., Doyle, R.: Enabling future robotic missions with multicore processors. In: Infotech@ Aerospace 2011, p. 1447 (2011)
14. Rieber, R.R.: Designing for a martian road trip: The mobility system for mars-2020 (2017), keynote Talk: Mars Forum (URS: URS270204, CL17-5707)
15. Santana, P., Vaquero, T., Toledo, C., Wang, A., Fang, C., Williams, B.: Paris: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty. In: International Conference on Automated Planning and Scheduling (ICAPS) (2016)
16. Sell, S., Chen, A., Davis, J., San Martin, M., Serricchio, F., Singh, G.: Powered flight design and reconstructed performance summary for the mars science laboratory mission. Tech. rep., Jet Propulsion Laboratory, National Aeronautics and Space Administration (2013)
17. Tokekar, P., Hook, J.V., Mulla, D., Isler, V.: Sensor planning for a symbiotic uav and ugv system for precision agriculture. *IEEE Transactions on Robotics* 32(6), 1498–1511 (Dec 2016)
18. Wyatt, E.J., Belov, K., Burleigh, S., Castillo-Rogez, J., Chien, S., Clare, L., Lazio, J.: 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). pp. 1–6 (Sept 2017)