Orbit Design of an Autonomous Space-based SSA Swarm: Distributed Machine Learning at the Edge

Lorraine Weis,* **Islam Hussein**†

August 29, 2019

ABSTRACT

The goal of machine learning is to extract utility from more data than can be explicitly analyzed. As the Space Situational Awareness (SSA) problem grows with more and more objects in space, more sophistication is needed to understand the available data. Moving from centralized processing to distributed analysis leans in to the distributed nature of the SSA problem - sensors and targets of interest are spread from the earth's surface to cis-lunar space. By pushing more processing to the edge, we can increase the overall level of autonomy of the sensor network. Distributed decision-making allows greater flexibility and resilience to changing local circumstances. Here, we architect a machine learning framework to show how an autonomous sensor swarm might be designed from a scheduling perspective. The simulated swarm uses Long-Short Term memory to provide schedules, which are then used to evaluate the orbit configuration.

1. INTRODUCTION

Distributed Satellite Systems (DSS) are becoming more common, as launching becomes more accessible and hardware costs conitnue to decrease.[1] Both formation flying and more loosely coordinated swarms enable broad types of space missions, with greater coverage and flexibility than monolithic systems. Examples of DSS include the experimental NASA CubeSat Proximity Operations Demonstration (CPOD) mission[2] the synthetic aperture radar (SAR) TanDEM-X mission,[3] the formation flying and rendezvous and docking demonstration PRISMA mission,[4] and the Gravity Recovery and Climate Experiment (GRACE) mission.[5]

The goal of distributed machine learning is to frame problems such that it can be split into smaller problems according to hardware and communication constraints.[8,9] With big data, of all sorts, scalability is a key factor of any machine learning algorithm, and numerous techniques might be useful.[10] These include gradient compression, sparsification, knowledge transfer learning, and federated learning.[11,12] Based on the specific problem, distributing the workload might take many forms, since power, communication bandwidth and reliability, and problem structure might be limiting to different degrees. For a space-based sensor swarm, we need to split the task scheduling problem into very minimal pieces, and limit communication. This means that for the orbit design, it may be easier to pick relative motion that makes it easier to consistently splitting the target object pool in an efficient manner.

Part of what enables new applications of DSS is increasing autonomy. However, much of space operations remains both centralized and human-intensive. Allowing greater autonomy in decision making, at all levels of the mission, will create new behaviors and mission types and drive innovative use of space. Rather than deliberatively pre-planning actions and maneuvers, we can move towards more reactive strategies. Particularly for payload scheduling, this will generate flexible adjustments in a changing environment, without the delays associated with centrally processing the entire decision algorithm. By bringing decisions closer to data collection, the DSS will be far more flexible, and what data is actually needed for general mission oversight can still be centralized. This sort of hybrid approach[1] is a part of current practice, but machine learning can help push more responsibility to the edge nodes of the sensor network. In this paper we explore a reactive on-board approach to payload resource planning.

1.1 At the Edge Decisions

Moving more autonomy to the edges of a distributed system can allow for faster responses, reduce demands on communication links, and enable more versatile behavior. There are many ways to design a distributed learning system and which pieces of teh algorhtm and hardware system are distributed. One might parallelize a model to run on

*Research Scientist, L3 Applied Defense Solutions, Columbia, Maryland, USA
†Senior Scientist, L3 Applied Defense Solutions, Herndon, VA, USA

multiple machines,[13] train the same model on subsets of the data or train parallel models on the same full data,[14] or some combination.[15] These approaches allow machine learning on larger datasets, deeper model networks, and reduce communication needs between system components.

For SSA decision making at the edge, we need algorithms that are lightweight in computation, memory, and communication. An exhaustive search of possible schedules is much too expensive to run on orbit, therefore lighter weight algorithms are needed. Furthermore, tasking strategies should be able to deal with a highly dynamic problem space - relative periodicities can shift substantially, targets may maneuver to new orbit parameters, unexpected events may strongly shift tasking priorities. Rather than having a hand-tuned set of optimization heuristics, an autonomous SSA swarm should be able to use online training to adjust tasking schedules to fit the new environment, with minimal data transfer from the ground.

2. DESIGNING AN AUTONOMOUS SSA SWARM

The effectiveness of a space-based SSA swarm will depend greatly on orbit design. This can be evaluated on coverage and resiliency for a target population through a distributed deep machine learning tasking algorithm. Orbit diversity, number of swarm members, and other factors can greatly affect swarm performance in SSA. The distributed deep learning algorithm uses recursive neural nets to assign subsets of the target population to individual swarm members and then calculate near-optimal tasking schedules. While training may be initiated on ground-based processors, the intent is to allow continued online training on the spacecraft themselves, and use peer-to-peer communication to allow global optimization of the many-to-many tasking problem. The use of Recursive Neural Networks (RNNs) allows the space-based resources to maintain a simplified model of the nonlinear dynamics that can still adjust to changing environmental parameters. Long-Short Term Memory cells allow online distributed training as the target population evolves through time. Aggregation among swarm members allows flexible target assignments and benefit from broader datasets. By using this underlying framework, we can evaluate the effectiveness swarm orbital designs directly, as well as gain intuition on which regimes such a swarm might be the most effective.

Consider a swarm of space-based sensors for space situational awareness. Autonomous sensor tasking needs to deal with a highly nonlinear dynamic problems space, as well as severe constraints in power, computation, action, and geometry. By moving the tasking algorithm from the ground to the swarm, it is no longer limited by the up/down link, and can use more in-situ information for its optimization. While communication among swarm members allows much better data transfer than with the ground, it is still distributed, and the processing power is more limited. Here, we design a Long Short Term Memory (LSTM) recursive neural network (RNN) in order to learn how to optimally task a small swarm on a population of targets. After training, the neural net is used to create a tasking schedule for the entire swarm, which is evaluated based on an information-gain metric. These schedule metrics can then be used to evaluate swarm orbit designs, and optimize the strategic control of the entire sensor swarm.

As a simple example problem, we use a target set of eleven geosynchronous satellites, with a swarm size of three. We train the LSTM network using simplfied orbital dynamics, and evaluate the resulting schedules. While this is not a particularly evocative example, it shows how designing the autonomous swarm could work practically - by progressively replacing the simulations with higher fidelity models, and the schedule evaluation with anticipated impact on an existing space catalog.

2.1 Long-Short Term Memory Recurrent Neural Net

The goal of the RNN is to implicitly encode the real, nonlinear orbital dynamics in a parameter space that is more naturally parallelizable in low power processing devices, such as FPGAs. Rather than explicitly calculating and predicting the uncertain perturbation forces, the onboard neural net will model the individual rewards - based on feedback from intermittent communication from the full SSA processing chain. Since the LSTM architecture allows for online training, the scheduling priorities will be able to adapt.

The LSTM cell uses internal neural networks to “gate” information. Each time step, the internal cell state is updated from the previous output, the new input, and the previous state using both sigmoid and tanh activation functions. Stacking LSTM cells allows for “hidden” layers, which can better model more complex problem topologies. For this work, we use two layers of LSTM cells in the recursive neural net.

LSTMs are commonly used in wide variety of applications, such as handwriting interpretation and text prediction. These use the deep learning layers to predict the likelihood of a particular output being correct, in the context of a

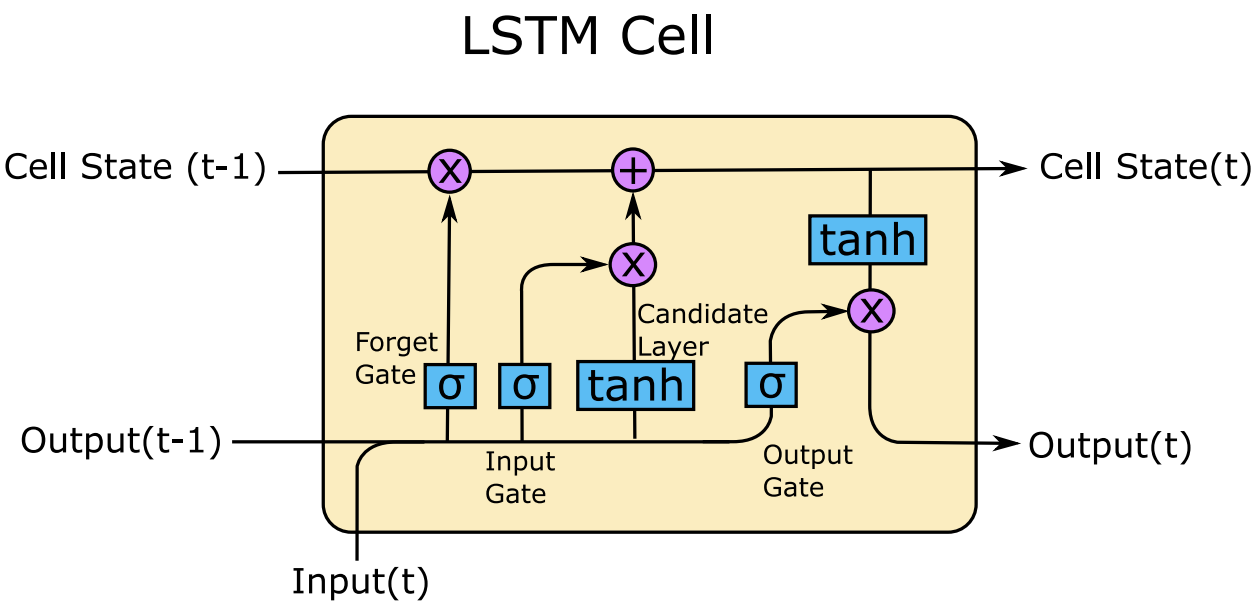


Fig. 1: Diagram of an LSTM cell with internal sigmoid and hyperbolic tangent neural networks acting as gates.

series of inputs. A text prediction RNN trained on a reference text can embed the relationship of words in sentence structure using a high-dimensional parameter space. Similarly, we are using an LSTM to embed the highly nonlinear dynamics of relative orbit trajectories and low dimensional measurements in a form that is more efficiently processed in an on-board setting.

Here, in the absence of labeled training data, we use a performance metric based on the information gain expected from a given tasking geometry. Since both targets and sensors are in orbital motion, the trained model incorporates the dynamic relationship in its training space implicitly. In a real sensor swarm system, the online training could have feedback directly from the SSA processing, either onboard, or intermittently from central aggregation. The ability to "forget" information, a key feature of LSTM-based RNNs, prevents overfitting to short term dynamics, while still allowing online adaptation to changing orbital relationships.

For the loss function, we use a proxy for informational entropy: a function of the relative distance to each target from each candidate sensor. This is a simplified model of rating each tasking schedule based on the resulting state uncertainties. In the next section, we compare results from a swarm of sensor spacecrafts with diverse orbits, with the results from an all LEO swarm.

To enable the RNN to learn from long term dynamics, we simulate high cadence data over many months. After selecting a randomized batch, we iterate over short term scheduling windows. This parallels how the system might perform online learning in the practical system. Figure 2 shows how the training and evaluation relate to the simulated ephemerides.

For training, we explicitly calculate the loss function for each sensor-target pair for each time step in the schedule - creating simulated labeled data. The "best" target for each sensor at each time is thus formulated as a one-hot matrix. The weights and biases in the RNN are then trained to make a similar selection. After training through the batch, we evaluate how the trained network would perform in that batch. This is analogous to how the scheduling network would impact a real catalog.

2.2 Results

The first step is to show that the dynamics captured by the trained RNN can adequately produce swarm tasking schedules for an SSA mission. Here, we define a small population of target objects, as well as a small sensor swarm. The loss function uses relative distance between the sensor and its target as a measure of efficacy for a particular tasking. The LSTM is then trained to maximize this efficacy in the schedule window. Sample results of this training are shown in Figure 3.

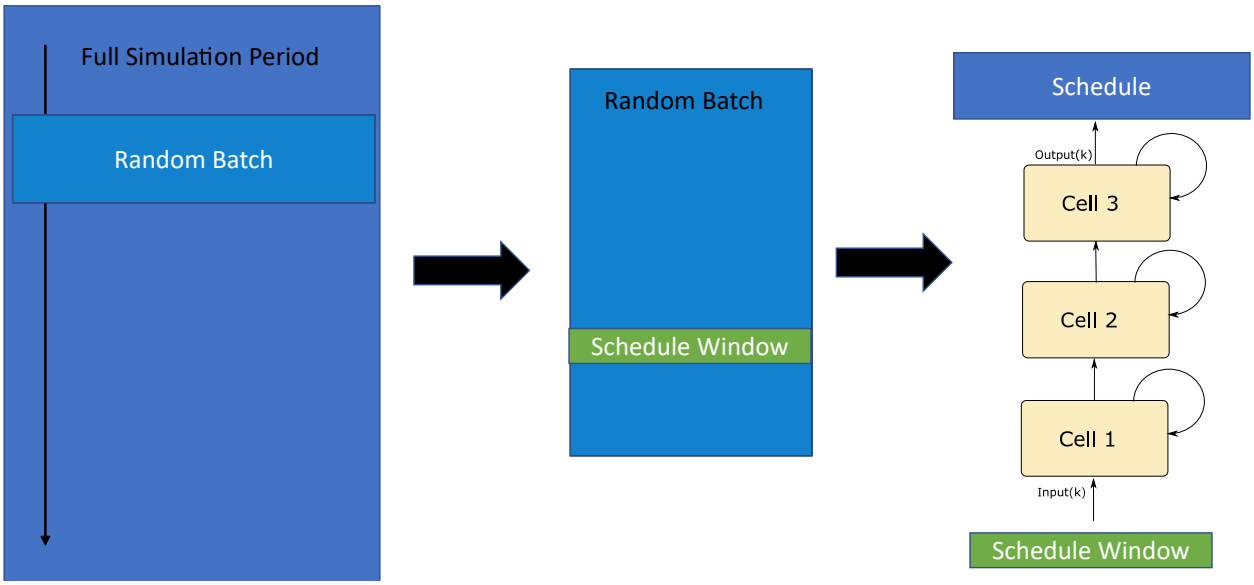


Fig. 2: Architecture for applying an RNN to the SSA scheduling problem.

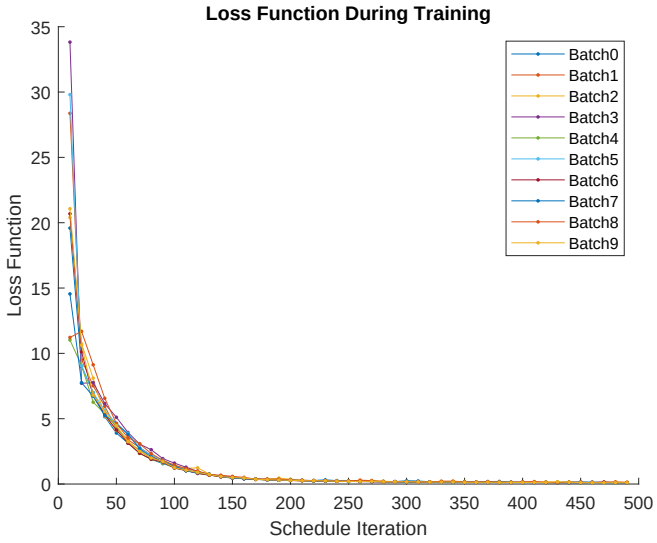
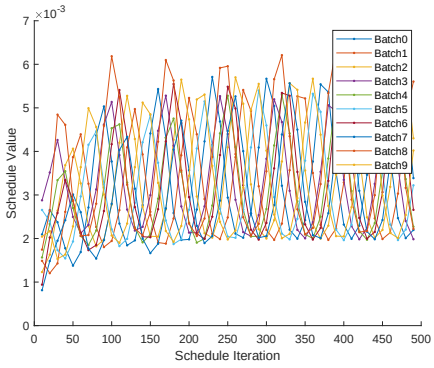
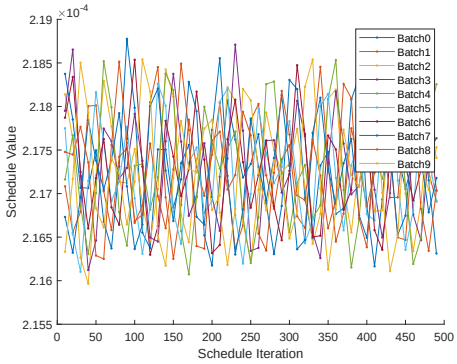


Fig. 3: The performance metric shows improvement as the RNN is trained.

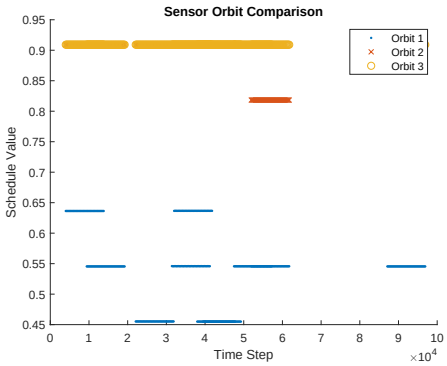


(a) For the diverse swarm, the schedule value tends to increase somewhat in the early iterations, but mostly governed by dynamics.

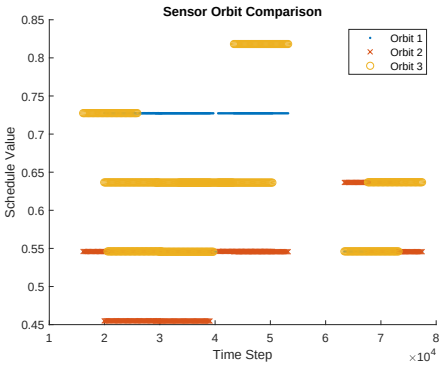


(b) The all LEO swarm shows no improvement over the training period.

Fig. 4: Comparison of orbit design performance.



(a) The variation in the schedule rating can be used for higher level analysis.



(b) An all LEO swarm does not have much differentiation between individual members for this population of targets

Fig. 5: Comparison of orbit design performance.

One of the benefits of LSTM is its ability to allow online training - the system will continue to weigh new inputs and external validation with its existing trained model. Here, we continue training on a randomly chosen batch to show that the RNN can adapt to a new configuration of the objects in orbital space. The lack of differentiation between batches means that the simplified problem chosen was too simple - the RNN cannot learn anything new in the new batch and thus has similar performance. We can also compare the tasking performance of each candidate sensor orbit on the same population of targets. This will allow the tuning of orbit parameters. Figure 5a shows the relative performance of three sensor orbit choices through the entire scheduling window.

3. SUMMARY

Developing an orbit plan for a large space-based sensor swarm is immensely challenging. By formulating the problem in a manner that allows for multiple levels of machine learning, this work aims to show how missions designed for autonomy can succeed. Individual sensors can intelligently task themselves in order to improve overall situational awareness and flexibly adjust to new tasking priorities or other changing circumstances. To truly take advantage of this, the swarm must be designed with autonomy in mind. By spreading out the load of difficult or high priority objects intelligently among the swarm, the overall performance of the sensor network will be less impacted. For a space-based sensor swarm, we need to split the task scheduling problem into very minimal pieces, and limit communication. This means that for the orbit design, it may be easier to pick relative motion that makes it easier to consistently splitting the

target object pool in an efficient manner.

REFERENCES

[1] M. D'Errico, *Distributed Space Missions for Earth System Monitoring*. New York, NY: Springer and Microcosm Press, 2013.

[2] C. W. Roscoe, J. J. Westphal, and E. Mosleh, "Overview and gnc design of the cubesat proximity operations demonstration (cpod) mission," *Acta Astronautica*, vol. 153, pp. 410 – 421, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094576517314042

[3] J.-S. Ardaens and D. Fischer, "Tandem-x autonomous formation flying system: Flight results," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 709 – 714, 2011, 18th IFAC World Congress.

[4] S. D'Amico, J. S. Ardaens, and R. Larsson, "Spaceborne autonomous formation-flying experiment on the prisma mission," *Journal of Guidance Control and Dynamics*, vol. 35, no. 3, pp. 834–850, May–June 2012.

[5] M. Kirschner, F.-H. Massmann, and M. Steinhoff, "GRACE," in *Distributed Space Missions for Earth System Monitoring*, M. D'Errico, Ed. New York, NY: Springer and Microcosm Press, 2013, ch. 19, pp. 547–574.

[6] O. Brown and P. Eremenko, "The value proposition for fractionated space architectures," *2006 AIAA Space*, 2006.

[7] T. Risen. (2017) Disaggregation.

[8] T. Tuor, S. Wang, T. Salonidis, B. J. Ko, and K. K. Leung, "Demo abstract: Distributed machine learning at resource-limited edge nodes," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018, pp. 1–2.

[9] T. Tuor, S. Wang, K. K. Leung, and K. S. Chan, "Distributed machine learning in coalition environments: Overview of techniques," *2018 21st International Conference on Information Fusion (FUSION)*, pp. 814–821, 2018.

[10] A. Sapio, M. Canini, C.-Y. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. R. K. Ports, and P. Richtarik, "Scaling distributed machine learning with in-network aggregation," KAUST, Tech. Rep. MSR-TR-2019-9, February 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/scaling-distributed-machine-learning-with-in-network-aggregation/

[11] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *ArXiv*, vol. abs/1901.00844, 2019.

[12] Z. Y. Zhou, X. Chen, E. Z. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *ArXiv*, vol. abs/1905.10083, 2019.

[13] M. Teng and F. Wood, "High throughput synchronous distributed stochastic gradient descent," *CoRR*, vol. abs/1803.04209, 2018. [Online]. Available: http://arxiv.org/abs/1803.04209

[14] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1223–1231. [Online]. Available: http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf

[15] Z. Zhou, H. Liao, B. Gu, K. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," *IEEE Network*, vol. 32, no. 4, pp. 54–60, August 2018.

[16] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 26–35. [Online]. Available: http://doi.acm.org/10.1145/2847263.2847265

[17] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "Dlau: A scalable deep learning accelerator unit on fpga," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 36, no. 3, pp. 513–517, Mar. 2017. [Online]. Available: https://doi.org/10.1109/TCAD.2016.2587683

[18] F. Ortega-Zamorano, J. M. Jerez, I. Gómez, and L. Franco, "Layer multiplexing fpga implementation for deep back-propagation learning," *Integrated Computer-Aided Engineering*, vol. 24, pp. 171–185, 2017.