[SSC21-P1-50]

SmartSat Constellation – A Deep Reinforcement Learning Approach for Decentralized Coordination

Rajesh Aggarwal, Junghwan Kim, Deepanshu Agarwal, Ayushi Tiwari
Avyuct LLC
2832 Thistleberry Court, Herndon, VA 20171, USA
ra@avyuct.com

James Baldo
Volgenau School of Engineering
George Mason University
jbaldo@gmu.edu

ABSTRACT

With rapid advancements in satellite technology, the amount of low earth orbit satellites has grown significantly which are primarily deployed for weather monitoring, earth observation or military purposes. Due to this reason, there has been an increased interest in enhancing the level of autonomy and cognition, onboard satellites to achieve optimal data collection. Optimal data is said to be collected when the satellites in a small sat constellation work together to collect information. This means that even if one of the satellites has missed out on some important information, the others can still collect them. A satellite constellation can be considered as a multi-agent reinforcement learning system. Having these agents coordinate with one another, can reduce the amount of time required to perform a task. The state-of-the-art satellite constellations follow a centralized coordination mechanism in which one primary satellite controls the rest of the satellites. This process is computationally more expensive and requires substantial communication between the satellites. It has a single point of failure and communication might be affected if the primary satellite fails. On the other hand, decentralized coordination allows agents to control their behavior themselves without the command of a supervised master. In this case, there is less inter-satellite communication which reduces the requirement for specialized onboard computational hardware. The proposal constitutes leveraging the Multi-Agent Deep Deterministic Policy Gradient [2] (MADDPG) algorithm to train the agents (satellites) to achieve optimal data collection. There are multiple use cases for the proposed solution such as illegal maritime activity tracking, natural disaster detection and assessing building damage after a natural disaster. The proposed solution focuses on tracking of ships in an extensively simulated environment for which a custom ship environment was created by leveraging OpenAI Gym [12]. By providing on-board autonomy, we aim to reduce frequent Earth Station (ES) communication significantly and enhance data collection capability.

INTRODUCTION

In Multi-Agent Reinforcement Learning (MARL), an agent is trained for maximizing its expected return by interacting with an environment that contains other learning agents. The use of a Centralized Training and a Decentralized Execution (CTDE) procedure [1,2,3,4,5] is a popular framework for MARL. In CTDE procedure, we have centralized critics to approximate the value function of the aggregated observations-actions pairs and train actors restricted to the observation of a single agent. Such critics, have the potential to steer the agents' policies toward highly rewarding behaviors when they are exposed to joint actions in a coordinated manner. This approach might fail in scenarios where such behaviors are unlikely to occur by chance, as these approaches might depend on the agents luckily stumbling on these collective actions in order to grasp their benefit. Thus, we hypothesize that in such scenarios, coordination-promoting inductive biases on the policy search could help discover successful behaviors more efficiently and supersede task-specific reward shaping and curriculum learning [1,2,3,4,5]. To motivate this proposition, we leverage the satellite sensor data and create a simple multi-agent coordination environment for ship tracking.

Before we delve into our experiments, we would like to present an analogy with robots that will help us understand "what is coordination and when do we need it?" Typically, a multi-agent systems (MAS) model of development is pursued when distributed processing and distributed control are required [10]. An issue of

MAS research is to determine how to obtain globally coherent behavior from the system when the agents operate autonomously and asynchronously. In general, when the agents share resources or the tasks being performed by the agents interact, the agents must explicitly work to coordinate their activities. Consider a simple physical example. Suppose there are two maintenance robots, we will call them as agents A1 and A2. These agents are assigned the joint task of moving a large box from one room to another. Both robots also have a set of other independent activities that must be performed, e.g., cleaning the windows. We assume that neither robot can lift the box by him/herself. In order for the robots to move the box together they must coordinate their activities in some manner. This is an example of communication-based coordination that produces a temporal sequencing of activities. This shall enable the robots to interact and carry out the joint task over a shared resource – which is the box in this case. Without the coordination process, it is unlikely that the box would ever be moved as desired unless the robots randomly decided to move the box at the same moment in time. In general, achieving global coherence in a MAS where tasks interact requires coordination [10].

BACKGROUND

Most Reinforcement Learning methods [9] fall into one of the following two categories: (a) Actor-only methods and b) Critic-only methods. Actor-only methods work with a parameterized family of policies. The gradient of the performance, with respect to the actor parameters, is directly estimated by simulation, and the parameters are updated in a direction of improvement [4, 5, 8, 13]. There is a possible drawback of such methods which is that the gradient estimators may have a large variance. Furthermore, as the policy changes, a new gradient is estimated independently of past estimates. Since, there is no accumulation and consolidation of older information. So, any kind of "learning" is not taking place in this case.

In case of Critic-only methods [9], the agents try to learn how to approximate the value function of a certain state-action pair and aim at learning an approximate solution to the Bellman equation, which will then hopefully prescribe a near-optimal policy. Such methods are indirect as they do not try to optimize directly over a policy space. A method of this type may lack reliable guarantees in terms of near-optimality of the resulting policy. Hence, we need a better approach.

Two main components in policy gradient are the policy model and the value function. It makes a lot of sense to learn the value function in addition to the policy, since knowing the value function can assist the policy update, and that is exactly what the Actor-Critic method does.

Actor-critic methods [9] consist of two models, which may optionally share parameters: Critic updates the value function parameters w and depending on the algorithm it could be action-value $Q_w(a|s)$ or state-value $V_w(s)$. Actor updates the policy parameters θ for $\pi\theta(a|s)$, in the direction suggested by the critic.

Asynchronous Advantage Actor-Critic [11] (A3C), is a classic policy gradient method with a special focus on parallel training. In A3C, the critics learn the value function while multiple actors are trained in parallel and get synced with global parameters from time to time. Hence, A3C is designed to work well for parallel training.

Let's use the state-value function as an example. The loss function for state value is to minimize the mean squared error, $J_v(w)=(G_t-V_w(s))2J_v(w)=(G_t-V_w(s))^2$ and gradient descent can be applied to find the optimal w. This state-value function is used as the baseline in the policy gradient update. A3C enables the parallelism in multiple agent training. The gradient accumulation step can be considered as a parallelized reformation of minibatch-based stochastic gradient update: the values of w or θ get corrected by a little bit in the direction of each training thread independently.

Advantage Actor-Critic (A2C) [11] is a synchronous, deterministic version of A3C; that's why it is named as "A2C". In A3C each agent talks to the global parameters independently, so it is possible sometimes the thread-specific agents would be playing with policies of different versions and therefore the aggregated update would not be optimal. To resolve the inconsistency, a coordinator in A2C waits for all the parallel actors to finish their work before updating the global parameters and then in the next iteration parallel actors starts from the same policy. The synchronized gradient update keeps the training more cohesive and potentially to make convergence faster [11].

MULTI-AGENT DEEP DETERMINISTIC POLICY GRADIENT

MADDPG [2] is an adaptation of the Deep Deterministic Policy Gradient algorithm [7] to the multi-agent setting. Here, multiple agents are coordinate to complete a task using only local information. It allows the training of cooperating and competing decentralized policies through the use of a centralized training procedure. In this framework, each agent i possesses its own deterministic policy μ_i for action selection and critic Q_i for state-action value estimation, which are respectively parametrized by θ_i and φ_i. All parametric models are trained off-policy from previous transitions ζ_t: = (o_t, a_t, r_t, o_{t+1}) uniformly sampled from a replay buffer D. Note that o_t: = [o_{t1}, ...,

o_{tN}] is the joint observation vector and a_t: = [a_{t1}, a_{tN}] is the joint action vector, obtained by concatenating the individual observation vectors o_{ti} and action vectors a_{ti} of all N agents. Each centralized critic is trained to estimate the expected return for a particular agent i from the Q-learning loss [8]:

$$\mathcal{L}^i(\phi^i) = \mathbb{E}_{\zeta_t \sim \mathcal{D}}\left[\frac{1}{2}\left(Q^i(\mathbf{o}_t, \mathbf{a}_t; \phi^i) - y_t^i\right)^2\right]$$
$$y_t^i = r_t^i + \gamma Q^i(\mathbf{o}_{t+1}, \mathbf{a}_{t+1}; \bar{\phi}^i)\Big|_{a_{t+1}^j = \mu^j(o_{t+1}^j; \bar{\theta}^j)\,\forall j} \quad (1)$$

For a given set of weights w, we define its target counterpart $\bar{w}$, updated from $\bar{w} \leftarrow we + (1 - \tau)\,\bar{w}$ where τ is a hyper-parameter. The algorithm discounts the rewards based on its step and updates the policies to maximize the expected discounted rewards.

$$J_{PG}^i(\theta^i) = \mathbb{E}_{\mathbf{o}_t \sim \mathcal{D}}\left[Q^i(\mathbf{o}_t, \mathbf{a}_t)\Big|_{\substack{a_t^i = \mu^i(o_t^i; \theta^i),\\ a_t^j = \mu^j(o_t^j; \bar{\theta}^j)\,\forall j \neq i}}\right] \quad (2)$$

The non-stationary multi-agent environment can be stationary by sharing each agents' actions and policies and that shared information was used in a centralized training. This centralized training provides cooperation between agents even though they do decentralized execution. In the centralized training, the MADDPG doesn't specify any particular conversation format between agents, the cooperation can be learned based on the rewards, and this allows diverse applications.

COORDINATION OF SATELLITES IN A CONSTELLATION

A satellite constellation is a group of satellites working together as a system. Unlike a single satellite, a constellation can provide global or near global coverage of Earth. Satellites are typically placed in sets of complementary orbital planes and connect to globally distributed ground stations. They may also use inter-satellite communication.

Low Earth orbiting satellites (LEOs) often use satellite constellation because the coverage area provided by a single LEO satellite only covers a small area that moves as the satellite travels at the high angular velocity needed to maintain its orbit. Many LEO satellites are needed to maintain continuous coverage over an area.

This contrasts with geostationary satellites, where a single satellite, moving at the same angular velocity as the rotation of the Earth's surface, provides permanent coverage over a large area. A constellation of satellites can be regarded as a multi-agent system. Coordination of these agents can result in performing a set of tasks that would require greater time for each of the agent working individually. The agents can also share resources that is sometimes required to perform a task.

However, coordination of these agents is a challenging task. The coordination is generally distributed and can be centralized or decentralized.

Centralized Coordination:

This is a master/worker approach where a single centralized planner (master) is used to coordinate other agents (workers). The selection of master can be manual or dynamic. This is conceptually a simple planning approach where all commands are sequenced, and master can perform checks and direct the worker agents. However, this approach tends to be computationally expensive and might require substantial communication. This approach is also a single point of failure if the master becomes inoperable for any reason.

Decentralized Coordination:

Decentralized coordination strategy allows agents to control their behavior based only on individual decisions without a supervised specific master for operation. This is generally used for a constellation with large number of autonomous agents. Recall the robot/box example stated previously, where the coordination episode was peer-to-peer. Imagine now a room full of robots, each having multiple joint tasks with other agents and all sharing physical resources such as tools and floorspace or X/Y coordinates [10]. Without coordination, the said room full of robots would have much in common with a preschool “free play session” with robots moving about, unable to perform tasks due to obstacle avoidance systems always diverting them from their desired directions or due to the lack of a required tool.

There are two primary ways to coordinate this room full of robots – either in a distributed peer-to-peer (or group to group) fashion or in a centralized fashion. When coordination is distributed each agent is responsible for determining when to interact with another agent and then having a dialog to determine how they should sequence their activities to achieve coherence. When coordination is centralized generally one agent plans for the others or manages a shared resource. Note that in the example above coordination focuses on when to perform a given task. Coordination can also be about which tasks to perform, what resources to use, how to perform a task, and so forth. While the robot domain is good for illustrating conceptually the coordination problem, the need for coordination is not limited to robots. Software agents, humans, and systems composed of mixes of agents, humans, and robots [10] all have a need for some kind of coordination. When the tasks or activities of different parties interact, in a

setting where control is distributed (parties are autonomous), coordination is needed. We attempt to emulate the coordination of satellites in a constellation to achieve optimal target coverage.

The biggest obstacle for Reinforcement learning with multi-agents is due to their nonstationary environment. No matter how good policies are for each agent, it's unable to cooperate if an agent does not know other agents' actions and behavior. This might happen because it's infeasible for satellites to communicate with each other in real-time. However, by leveraging the capabilities of the MADDPG algorithm, we can resolve this issue. At first, after training, the MADDPG algorithm uses a decentralized execution framework. So, we don't need to worry about the disconnection between satellites while they are in LEO, MEO or higher orbits. Secondly, even though the algorithm uses a decentralized execution framework, it still uses a centralized training approach. So, by providing actions taken by other agents and the policy alterations, the environment can be stationery and cooperative during the tracking target (ship) task in the training process.

TRAINING ENVIRONMENT

Our continuous control tasks are built on OpenAI's multi-agent particle environment [6]. We aimed to enhance performance of a multi-agent simulation by developing and integrating Reinforcement Learning based on coordination. Pattern-of-life discovery would be obtained by observing typical and anomalous activity of targets. All satellites in a constellation try to track the object (ship) as much as possible. But they are still moving in their orbit and the only action they are allowed is altering their on-board camera angle. As long as satellites can't see the object, they don't have another option except tilting their cameras. But we believe that if one of them finds the object, the other satellites can bring optimization of their camera control.

Our physical world for this task would be initially guided by sensors, environmental conditions, availability of a resource, trafficability, potential final destinations, and target patterns of life. We have presented two experiments with their results in this paper. The simulation environment is a 18x18 grid where the blue portion denotes the sea area and the green portion denotes the land area or ports. There are four satellites forming a constellation and a single target which is a ship that they are trying to track. The satellites with camera sensors are the reinforcement learning agents and the ship is the target. The objective is to keep track of the ship as much as possible. One important point to note here is that our training environment is less complex and it can be extended to larger and more complex environments. We basically want to compare the difference between the percentage of target coverage by using Deep Reinforcement Learning techniques and without using them.

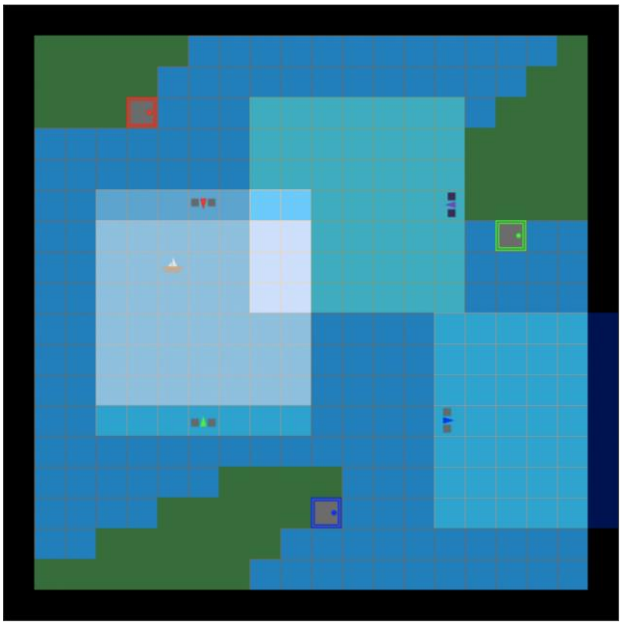

Figure 1: Simulation environment using OpenAI Gym

EXPERIMENT 1

In Experiment 1, we assume that the four satellites are in the low earth orbit (LEO). So, it seems like the satellites are moving because they are faster than the earth's rotation.
The four satellites are moving in a specific orbit, and they try to track the ship as much as possible by tilting the angle of their camera. In this scenario, the ship moves only between port green and blue to prevent too complicated environments because the orbit of satellites already brings complexity to the environment. 1 epoch has 50 steps which mean each satellite can tilt their angle 50 times in the location of their orbit while the ship moves 50 steps.

Results - To calculate the accuracy of testing, we test the environment's 100 epochs and calculated the average coverage proportion within 50 steps. Target coverage was found to be 63.5%. The coverage with random camera sensor motion was found to be 50.1%.

EXPERIMENT 2

In Experiment 2, we have four satellites and one target ship. We assumed that the four satellites are in the low

earth orbit (LEO). They are moving in a specific orbit, and they try to track the ship as much as possible by tilting the angle of their camera. In this scenario, the ship starts from red port and it can move randomly between all the ports. A single epoch has 50 steps which means that each satellite can tilt their angle 50 times in the location of their orbit while the ship moves 50 steps.

Results - To calculate the accuracy of testing, we tested the environment's 100 epochs and calculated the average coverage proportion within 50 steps. Target coverage was found to be ~73%. The coverage with random camera motion was found to be 60.4%.

CONCLUSION

We were able to successfully emulate the maritime environment where a ship is moving in order to reach its destination (port). We tested the two scenarios, one using the MADDPG algorithm and the other without it. We found that there was a higher target coverage with Deep Reinforcement Learning approach as compared to coverage achieved from random camera motion. There was 13.4% more target coverage in case of Experiment 1 and 2 where we are using Deep RL approach as compared to the case 1 when the camera sensors are randomly trying to track the ship. The two scenarios can be helpful in different uses cases and through our future work which is proposed next, we plan to explore those spaces.

FUTURE WORK

Markov process modeling can be used to estimate state of the ship at the sea. This state estimation can complement the satellite coordination technique in the following way. We understand that we may not know the state X(t) perfectly. We would explore stochastic noise models as Hidden Markov model (HMM). We also understand that we may not know all the parameters required to define state and we would explore Parameter Estimation techniques. We shall also compare results from MADDPG using other multi-agent environment algorithms. Lastly, we would like to explore how we can integrate our simulation with Computer Vision techniques which can enhance the decision-making capabilities of the agents.

We would like to map the simulation to real world situation and geography, particularly we would like to emulate the maritime environment in South China sea. We would also like to apply these techniques to other experiments such as optimal collections for natural disasters, we plan to test our experiments onboard for actual small satellite constellation.

REFERENCES

1. Leal, P.H., Kartal, B. and Taylor, M.E. Is multiagent deep reinforcement learning the answer or the question? a brief survey. arXiv preprint arXiv:1810.05587, 2018.
2. Lowe, R., Wu, Y., Tamar, A., Harb, J. et all. Multi-agent actor-critic for mixed cooperative-competitive environments. In Advances in Neural Information Processing Systems, pages 6379–6390, 2017.
3. Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N. et all. Counterfactual multi-agent policy gradients. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
4. Iqbal, S., and Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In International Conference on Machine Learning, pages 2961–2970, 2019
5. Foerster, J., Song, F., Hughes, E., Burch, N., et all. Bayesian action decoder for deep multi-agent reinforcement learning. International Conference on Machine Learning, 2019.
6. Abbeel. Emergence of grounded compositional language in multi-agent populations. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
7. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., et all. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.
8. Watkins, C.J.C.H., and Dayan, P. Q-learning. Machine learning, 8(3-4):279–292, 1992
9. Konda, V.R., Tsitsiklis, J.N. Actor-Critic Algorithms. Laboratory for Information and Decision Systems, MIT
10. Wagner, T.P., Guralnik, V. (2004) Centralized VS. Decentralized Coordination: Two Application Case Studies. In: Wagner T.A. (eds) An Application Science for Multi-Agent Systems. Multiagent Systems, Artificial Societies, and Simulated Organizations, vol 10. Springer, Boston, MA. https://doi.org/10.1007/1-4020-7868-4_4
11. Mnih, V. et all. (2016). Asynchronous methods for Deep Reinforcement Learning. arXiv preprint arXiv:1602.01783
12. Brockman. G, et all. (2016). OpenAI Gym. arXiv preprint arXiv:1606.01540v1 [cs. LG] 5 Jun 2016