A Multi-agent Reinforcement Learning Method for Swarm Robots in Space Collaborative Exploration

Yixin Huang
School of Aeronautics and Astronautics
Shanghai JiaoTong University
Shanghai, China
e-mail: huangyxethan@sjtu.edu.cn

Shufan Wu*
School of Aeronautics and Astronautics
Shanghai JiaoTong University
Shanghai, China
e-mail: shufan.wu@sjtu.edu.cn

Zhongcheng Mu
School of Aeronautics and Astronautics
Shanghai JiaoTong University
Shanghai, China
e-mail: muzhongcheng@sjtu.edu.cn

Xiangyu Long
School of Aeronautics and Astronautics
Shanghai JiaoTong University
Shanghai, China
e-mail: xiangyu_long@sjtu.edu.cn

Sunhao Chu
School of Aeronautics and Astronautics
Shanghai JiaoTong University
Shanghai, China
e-mail: Chu1220@sjtu.edu.cn

Guohong Zhao
School of Aeronautics and Astronautics
Shanghai JiaoTong University
Shanghai, China
e-mail: ghzhao@sjtu.edu.cn

***Abstract*—Deep-space exploration missions are known as particularly challenging with high risk and cost, as they operate in environments with high uncertainty. The fault of exploration robot can even cause the whole mission to failure. One of the solutions is to use swarm robots to operate missions collaboratively. Compared with a single capable robot, a swarm of less sophisticated robots can cooperate on multiple and complex tasks. Reinforcement learning (RL) has made a variety of progress in multi-agent system autonomous cooperative control domains. In this paper, we construct a collaborative exploration scenario, where a multi-robot system explores an unknown Mars surface. Tasks are assigned to robots by human scientists and each robot takes optimal policies autonomously. The method used to train policies is a multi-agent deep deterministic policy gradient algorithm (MADDPG) and we design an experience sample optimizer to improve this algorithm. The results show that, with the increase of robots and targets number, this method is more efficient than traditional deep RL algorithm in a multi-agent collaborative exploration environment.**

Keywords-space exploration; swarm robots; multi-agents; reinforcement learning; MADDPG

I. INTRODUCTION

In deep-space exploration missions (e.g. to asteroids, moon, or planets), automatic robots push the development of technology and automation. From the first moon rover launched on the moon surface in 1970s, such exploration missions have acquired valuable data. NASA launched the Sojourner to the Mars in 1997, the Spirit and the Opportunity landed on the Mars in 2004. The Curiosity landed in 2012 is still working and has achieved many valuable information. In 2007, JAXA's Hayabusa explorer landed on an asteroid to sample and return to the Earth 3 years later. In 2019, China's Chang'E-4 landed successfully and the Yutu-II rover toured the Von Karman crater on the Lunar [1]. Up to now, such space exploration missions have been undertaken by a single robotic explorer which not only limits the acquisition of detection data but also introduces an extremely high risk of mission failure. Meanwhile, high reliability and risk control requirements lead to high mission costs. New NASA mission concepts now being studied involve many small rovers operating cooperatively and collaboratively [2]. Compared with the traditional mission, the swarm concept offers several advantages: greater redundancy, protection of assets and reduced costs and risk. Swarm rovers can cooperate on multiple and complex tasks, surpassing even the most capable single rover [3]. Swarm of less sophisticated rovers can even share power resources [4] and communication resources [5]. Previous studies have shown that a large swarm, simple and even less reliable components, can offer better performance and reliability [3].

Nowadays, distributed artificial intelligence (DAI) has attracted tremendous attention due to its ability to settle complex computing problems. Multi-agent system (MAS) consist of autonomous entities known as agents, similarly to computing entities in DAI, agents use their knowledge to act on the environment to solve their allocated task or make a decision [6]. Several complex problems like multi-robot control, air traffic flow management and energy distribution

978-1-7281-6139-6/20/$31.00 ©2020 IEEE

can be modeled as a multi-agent learning problem. In this paper, the swarm space exploration robots' concept is typically a multi-agent problem.

A variety of researches have been focused on coordinating multiple robots in space exploration. For example, the control architecture for multi-robot planetary outposts (CAMPOUT) is an architecture for multiple robots in deeply coupled scenarios [7]. Other areas of multi-robot research for space exploration include multi-robot trajectory and path planning based on dynamic networks, satellites formation flight for cooperative observation and researches on the classic continuous rover problem (CRP) [3].

Reinforcement learning (RL) is a subfield of machine learning to address the problem of automatic learning of optimal decisions, agents select actions and the environment responds by present a reward and a new state [8]. The grand vision of artificial intelligence is to build autonomous agents that can interact with the environment and cooperate with each other. Methods of the learning problem for single agents comes the closest to the vision [9]. Many methods like Q-learning, Deep Q-learning, Policy gradient have made success in single agent domains where the environment stays stationary. However, traditional RL approaches are not suitable for multi-agent problems due to the changing agent's policy in the training progress and the environment becomes non-stationary [10] [11].

In this paper, we construct a collaborative exploration scenario, where a multi-robot system explores an unknown Mars surface. A multi-agent deep deterministic policy gradient algorithm is adopted to train policies and we design an experience sample optimizer to improve this algorithm. The results show that, with the increase of scale of robots' number and complexity of the environment, the MADDPG algorithm with the proposed experience sample optimizer is more efficient than the traditional deep RL algorithm.

II. Preliminaries and Backgriound

A. Collaborative Exploration Scenario

In 2015, NASA proposed to use the Mars Helicopter to explore the surface of Mars [12]. This plan is expected to become the" eye in the sky" and expand the exploration area. Together with remote sensing satellites on the Mars, we can effectively obtain a panoramic image of the detected area. However, limited with communication capability and resolution, we could not observe detailed information. For this reason, we construct such a scenario in which Mars helicopter or remote sensing satellite observe where we are interested firstly, then scientists pick points of interest (POIs) and rugged landscapes traps (e.g. meteor crater, canyon) after image processing. According to the distribution, swarm of robots will be released to explore POIs and keep away from traps. Fig. 1 describes the collaborative exploration scenario on the Mars surface. The global objective of the mission is to observe all POIs within optimal time and routes.

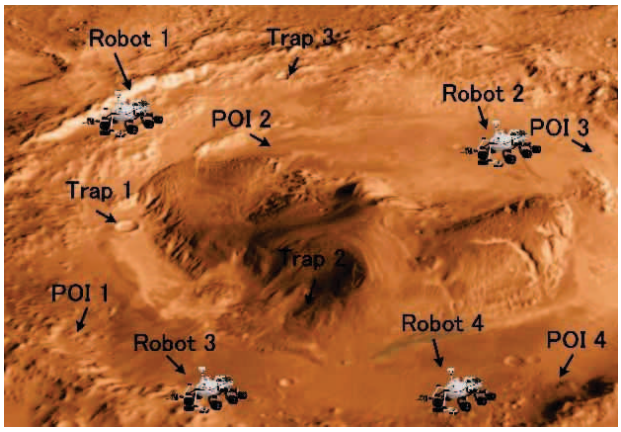

Figure 1. Collaborative exploration scenario on the Mars surface.

B. Deep Reinforcement Learning

1) Markov games

A reinforcement learning problem is modeled as a Markov Decision Process (MDP) and consists of an agent interacting with the environment. Multi-agent extension of MDP can be formalized as a partially observable Markov game [10]. The N-agent Markov game is defined by $\mathcal{S}$, a global state of the multi-agent system, and a set of N observations $\{\mathcal{O}_i\}$ and N actions $\{\mathcal{A}_i\}$. State $\mathcal{S}$ updates at each time step t, each agent obtains its observation O_i^t and takes an action A_i^t using its policy π_i.

Each agent obtains a reward r_i^t based on the global state $\mathcal{S}^t$ at time t and joint action of the swarm. The system then transforms to the next state $\mathcal{S}_{t+1}$ until reaching a terminal state. $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ is the global reward for agent i with a discount factor $\gamma \in (0,1]$. The objective of each agent is to maximize its global reward. Fig. 2 describes the state transfer diagram of Markov games and Fig. 3 shows the interaction between agent i and the environment.

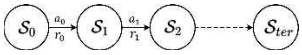

Figure 2. Markov games state transfer diagram.

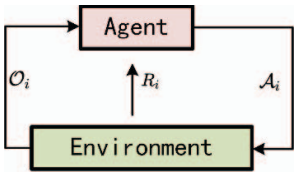

Figure 3. Interaction between agent i and the environment.

2) Policy Gradient (PG) algorithm

Policy Gradient approaches have made success in a variety of RL tasks. The main method is to adjust the parameter θ of the policy to maximize the estimation of

The China State Key Laboratory of Robotics (Grant No. 19Z1240010018).

global reward $J(\theta)=\mathbb{E}_{s\sim p^{\pi},a\sim\pi_\theta}[R]$ by taking steps in the direction of gradient $\nabla_\theta J(\theta)$. Where p^{π} donates the state distribution and π_i is the policy agent i takes. By using the action-state value function $Q^{\pi}(s,a)$, the gradient of policy can be written as [13]:

$$\nabla_\theta J(\theta)=\mathbb{E}_{s\sim p^{\pi},a\sim\pi_\theta}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(a\mid s)Q^{\pi}(s,a)\right] \quad (1)$$

However, in multi-agent settings, policy gradient methods are known to a high variance gradient estimates [10].

3) Deep deterministic PG algorithm

It has been proved that the stochastic policy gradient is equivalent to the deterministic policy gradient (DPG) [14]. In other words, to improve the policy, we just need to calculate the gradient in deterministic policies μ_θ, it can be written as:

$$\nabla_\theta J(\theta)=\mathbb{E}_{s\sim\mathcal{D}}\left[\nabla_\theta \boldsymbol{\mu}_\theta(a\mid s)\nabla_a Q^{\mu}(s,a)\big|_{a=\boldsymbol{\mu}_\theta(s)}\right] \quad (2)$$

Deep deterministic policy gradient (DDPG) is an extension of the DPG algorithm where the policy μ and Q^{μ} are approximated by deep neural networks. In the training process, sample trajectories are stored in a replay buffer of experiences.

4) Actor-critic (A2C) algorithm

The actor-critic algorithm is consist of the policy network and the value network. The policy network is called the actor which returns the probability distribution of actions, the actor uses the deep neural network (DNN) to approximate policy gradient. The value network is called the critic by using DNN to approximate the value function. The critic could evaluate the actor iteratively to optimize the policy. Fig. 4 shows the framework of A2C algorithm.

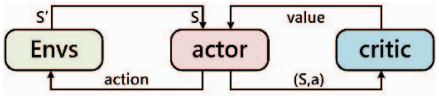

Figure 4. The actor-critic algorithm diagram.

III. METHOD

As discussed above, the traditional reinforcement learning algorithm performs poorly in multi-agent environments because of unstable environment for each agent. In this section, we describe the algorithm used in the space collaborative exploration scenario.

A. Multi-Agent Actor-Critic (MAAC)

The MAAC algorithm framework is centralized training with decentralized execution. So we can allow the policies to use extra information (e.g. global states) to make training, although this information is not used in execution. In another aspect, critics could learn from other agents' policies to optimize state-action value evaluation.

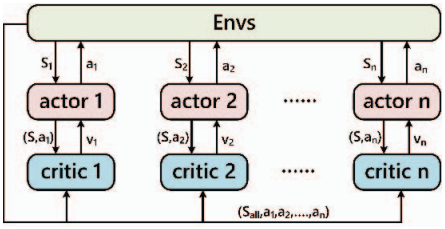

Figure 5. The multi-agent actor-critic algorithm diagram.

Fig. 5 shows the framework of MAAC algorithm. In the training process, each actor could obtain the global state s_{all} and action other agents take $\{a_1,a_2,\cdots,a_n\}$. In execution, agents observe partial states $\{s_1,s_2,\cdots,s_n\}$ and interact with the environment independently.

Similarly to (1), the gradient of expected reward for agent i can be written as:

$$\nabla_{\theta_i} J(\theta_i)=\mathbb{E}\left[\nabla_{\theta_i}\log \boldsymbol{\pi}_i(a_i\mid s_i)Q_i^{\pi}(\mathbf{x},a_1,\ldots,a_n)\right] \quad (3)$$

The $Q_i^{\pi}(\mathbf{x},a_1,\ldots,a_n)$ is the centralized state-action value function which take the state information $\mathbf{x}$ and actions of all agents $\{a_1,a_2,\cdots,a_n\}$ as input. $\mathbf{x}$ is consist of all observations, $\mathbf{x}=(s_1,s_2,\cdots,s_n)$. Since state-action value for each agent Q_i^{π} is learned separately, agents can have respective reward to be assessment index instead of global reward merely.

B. Multi-Agent Deep Deterministic PG (MADDPG)

Referring to this method, consider deterministic policies μ_{θ_i} (abbreviated as μ_i), the gradient of expected reward can be written as:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i)=\mathbb{E}\left[\nabla_{\theta_i}\boldsymbol{\mu}_i(a_i\mid s_i)\nabla_{a_i}Q_i^{\boldsymbol{\mu}}(x,a_1,\cdots,a_n)\right] \quad (4)$$

The experience replay buffer $\mathcal{D}$ is where to record experiences of all agents' trajectories, it is consist of the set $\{\mathbf{x},\mathbf{x}',a_1,a_2,\cdots,a_n,r_1,r_2,\cdots,r_n\}$. To learn the state-action value function $Q_i^{\boldsymbol{\mu}}$ corresponding to the optimal policy by minimizing the loss function:

$$\mathcal{L}(\theta_i)=\mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}\left[\left(Q_i^{\boldsymbol{\mu}}(\mathbf{x},a_1,\ldots,a_n)-y\right)^2\right] \quad (5)$$

where:

$$y=r_i+\gamma Q_i^{\boldsymbol{\mu}'}\left(\mathbf{x}',a_1',\ldots,a_n'\right)\big|_{a_j'=\boldsymbol{\mu}_j'(s_j)} \quad (6)$$

Here the target policies with delayed parameter θ_i' is $\boldsymbol{\mu}'=\left\{\boldsymbol{\mu}_{\theta_1'},\cdots,\boldsymbol{\mu}_{\theta_n'}\right\}$. In (5), it takes all agents' policies as critics' input. In a simulation environment, it is possible to obtain the policies set, but in a complex communicative scenario, this information is not always available. To make this approach feasible, other agents' policies can be approximated by $\bar{\mu}_{\phi_i}^{j}$ (abbreviated as $\bar{\mu}_i^{j}$), which represents

agent i's approximation of agent j's policy, where ϕ is the parameter. The cost function can be written as:

$$\mathcal{L}\left(\phi_i^j\right) = -\mathbb{E}_{s_j,a_j}\left[\log \bar{\boldsymbol{\mu}}_i^j\left(a_j \mid s_j\right) + \lambda H\left(\bar{\boldsymbol{\mu}}_i^j\right)\right] \tag{7}$$

where H is the entropy of the policy distribution. Similar to (6), the approximation value $\bar{y}$ can be written as follows:

$$\bar{y} = r_i + \gamma Q_i^{\boldsymbol{\mu}'}\left(\mathbf{x}', \bar{\boldsymbol{\mu}}_i'^1\left(s_1\right), \ldots, \bar{\boldsymbol{\mu}}_i'\left(s_i\right), \ldots, \bar{\boldsymbol{\mu}}_i'^n\left(s_n\right)\right) \tag{8}$$

Before updating the centralized Q-function $Q_i^{\boldsymbol{\mu}}$, we can sample a batch in the experience replay buffer to update ϕ_i^j.

As mentioned before, the environment is not stationary due to the agents' dynamic policies. To obtain more robustness in multi-agent collaborative environments, a policy ensemble method is adopted. For each agent, a collection of K different sub-policies is trained. At each episode, one particular sub-policy $\boldsymbol{\mu}_{\theta_i}^{(k)}$ (abbreviated as $\boldsymbol{\mu}_i^{(k)}$) is selected randomly to execute. For each agent, the objective is to maximize policy ensemble reward, which can be written as:

$$J_e\left(\boldsymbol{\mu}_i\right) = \mathbb{E}_{k\sim\mathrm{unif}(1,K),\mathbf{s}\sim p^{\boldsymbol{\mu}},a\sim\boldsymbol{\mu}_i^{(k)}}\left[R_i(s,a)\right] \tag{9}$$

For sub-policy $\boldsymbol{\mu}_i^{(k)}$, an experience replay buffer $\mathcal{D}_i^{(k)}$ is maintained to store different sub-policies in different episodes.

Accordingly, the gradient of the ensemble objective can be written as follows:

$$\nabla_{\theta_i^{(k)}} J_e\left(\boldsymbol{\mu}_i\right) = \frac{1}{K}\mathbb{E}_{\mathbf{x},a\sim\mathcal{D}_i^{(k)}}\left[\xi\right] \tag{10}$$

where ξ equals to:

$$\nabla_{\theta_i^{(k)}} \boldsymbol{\mu}_i^{(k)}\left(a_i \mid s_i\right) \nabla_{a_i} Q^{\boldsymbol{\mu}_i}\left(\mathbf{x}, a_1, \ldots, a_n\right)\Big|_{a_i=\boldsymbol{\mu}_i^{(k)}(s_i)} \tag{11}$$

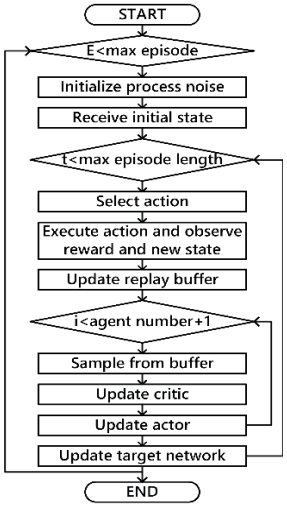

Figure 6. Flow chart of MADDPG algorithm.

The specific steps of the MADDPG algorithm are illustrated in Fig. 6.

C. Experience Sample Optimization

In the MADDPG, sub-policies for each agent to execute is selected randomly in each episode and stored in an experience replay buffer $\mathcal{D}_i^{(k)}$. We design an experience sample optimizer relying on the priority of current experience. The priority $Pr(i)$ is the probability of current experience being sampled, it can be calculated as follows:

$$Pr(i) = \frac{p_i^{\alpha}}{\sum_{j=1}^{M} p_j^{\alpha}} + \beta \tag{12}$$

where α is the amplification factor of priority, $\beta \in (0,1)$ is the bias factor of probability. p_i can be calculated as:

$$p_i = rank[Loss(i)] \tag{13}$$

$Loss$ is the target network loss function:

$$Loss = \left(Q^{\boldsymbol{\mu}}\left(s, a_1, a_2 \cdots a_n\right) - y\right)^2 \tag{14}$$

y is the evaluation Q-value and can be calculated as follows:

$$y = r + \gamma Q^{\boldsymbol{\mu}'}\left(s', a_1', a_2' \cdots a_n'\right)\Big|_{a_i'=\boldsymbol{\mu}_i'(s_i)} \tag{15}$$

In the MADDPG algorithm, we can sample experience from replay buffer according to the probability distribution calculated in (12).

IV. Experiment

In this section, we construct a collaborative exploration scenario to simulate the Mar surface environment firstly. To confirm the effectiveness of the experience sample optimizer proposed in section III, four comparison experiments on the simulation scenario are operated.

A. Scenario Construction

Section II introduces the emulational collaborative exploration scenario on Mars, where Mars helicopters or remote sensing satellites observe where we are interested. In this environment, human scientists pick a set of potential POIs and traps as the primary task, then a swarm of rover robots nearby will be released. Rovers observe the relative positions of other rovers, traps and POIs. In other word, the rovers have to explore all of the POIs. Further, the rovers are penalized when colliding with each other or falling into traps.

Four comparison experiments are operated as is shown in Fig. 7. In the square scenario, three colors represent different objects (Rovers, Traps and POIs). We set up different numbers (from 2 to 5) of entities to demonstrate the proposed algorithm's performance under different experiment complex conditions and scale of the multi-agent system environment.

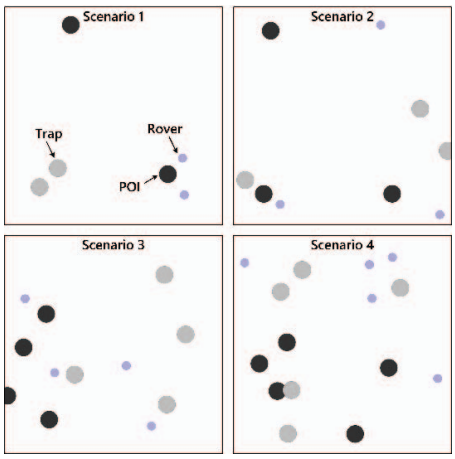

Figure 7. Comparision experiments with different number of entities.

In the exploration process, reward for each agent r_i is calculated as follows:

$$r_i = -\min_{j \leqslant N}(D(i,j)) - \Sigma C_i^{rr} - \Sigma C_i^{rt} \tag{16}$$

where $D(i,j)$ is the distance between rovers and POIs, ΣC_i^{rr} is collision counts between rovers, ΣC_i^{rt} is collision counts between rovers and traps.

$$C_i = \begin{cases} 1, \text{ collision} \\ 0, \text{ no collision} \end{cases} \tag{17}$$

The actor, critic and corresponding target network are parameterized by a two-layer ReLU multi-layer perceptron with 64 units per layer.

B. *Result and Analysis*

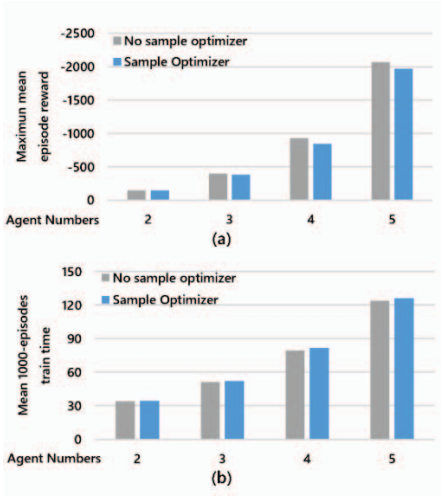

Figure 8. Comparision of MADDPG with/without texperience sample optimizer. (a): Sample optimizer's effect on maximum mean episode rewards. (b): Sample optimizer's effect on mean 1000-episodes train time.

With the scale of a multi-agent system becomes larger, the maximum mean episode rewards decreases and mean 1000-episodes train increases, as is shown in Fig. 8 and Fig. 9 illustrates that, when the MAS is of a small scale, the proposed experience sample optimizer has limited advantages. As the scenario becomes more and more complex, preferentially sampled experience helps to improve the training results. Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

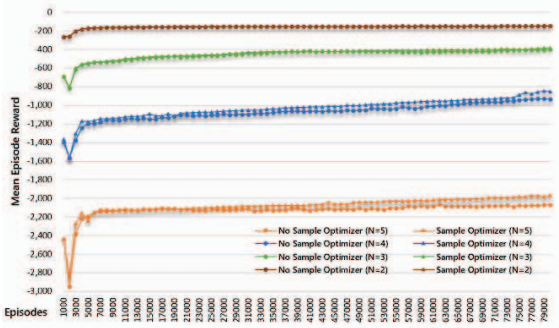

Figure 9. Mean episode rewards in collaborative exploration scenario.

TABLE I. EFFECTS OF EXPERIENCE SAMPLE OPTIMIZER

Agent's number	*Episode rewards*	*Training time*
2	1.42%	2.26%
3	3.08%	2.50%
4	4.71%	2.69%
5	8.78%	1.79%

Table I demonstrates the effects of proposed experience sample optimizer, with a little loss of mean training time, this method could increase episode rewards compared with the original MADDPG algorithm, which means a better collaborative exploration strategy.

V. CONCLUSION

Swarm robots' application in space collaborative exploration scenario is an alternative solution compared with a risky and costly single explorer. In this paper, we construct a collaborative exploration scenario, where human scientists assign primary task target to robots and the multi-robot system explore an unknown Mars surface taking optimal policies autonomously.

The method adopted in this scenario is a multi-agent deep deterministic policy gradient algorithm, with a proposed experience sample optimizer. The experiment results demonstrate that with the scale of the multi-agent system increase, the proposed optimizer can be of a better performance in mean episode reward.

ACKNOWLEDGMENT

This work is supported by the China State Key Laboratory of Robotics (Grant No. 19Z1240010018).

REFERENCES

[1] C. Li et al., "Chang'E-4 initial spectroscopic identification of lunar far-side mantle-derived materials," Nature, vol. 569, no. 7756, pp. 378–382, May 2019.

[2] W. Truszkowski, Ed., Autonomous and autonomic systems: with applications to NASA intelligent spacecraft operations and exploration systems. London ; New York: Springer, 2009.

[3] M. Colby, L. Yliniemi, and K. Tumer, "Autonomous Multiagent Space Exploration with High-Level Human Feedback," J. Aerosp. Inf. Syst., vol. 13, no. 8, pp. 301–315, Aug. 2016.

[4] K. Schreiner, "NASA's JPL Nanorover Outposts Project develops colony of solar-powered nanorovers," IEEE Intell. Syst., vol. 16, no. 2, pp. 80–82, Mar. 2001.

[5] S. Chien, A. Barrett, T. Estlin, and G. Rabideau, "A comparison of coordinated planning methods for cooperating rovers," in Proceedings of the fourth international conference on Autonomous agents - AGENTS '00, Barcelona, Spain, 2000, pp. 100–101.

[6] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," IEEE Access, vol. 6, pp. 28573–28593, 2018.

[7] T. Huntsberger et al., "CAMPOUT: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration," IEEE Trans. Syst. Man Cybern. - Part Syst. Hum., vol. 33, no. 5, pp. 550–559, Sep. 2003.

[8] R. S. Sutton and A. G. Barto, Reinforcement learning: an introduction, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.

[9] S. Kapoor, "Multi-Agent Reinforcement Learning: A Report on Challenges and Approaches," ArXiv180709427 Cs Stat, Jul. 2018.

[10] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 6379–6390.

[11] J. Foerster et al., "Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning," 2017, p. 10.

[12] B. Balaram et al., "Mars Helicopter Technology Demonstrator," in 2018 AIAA Atmospheric Flight Mechanics Conference, Kissimmee, Florida, 2018.

[13] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in Advances in neural information processing systems, 2000, p. 7.

[14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in the 31th International Conference on Machine Learning, Beijing, China, 2014, p. 9.