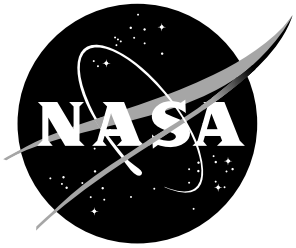
NASA/TM-2020-220439



Machine Learning For Planetary Mining Applications

Joshua Cook
Oregon State University, Corvallis, Oregon

Jamshid Samareh
Langley Research Center, Hampton, Virginia

January 2020

NASA STI Program. . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collection of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

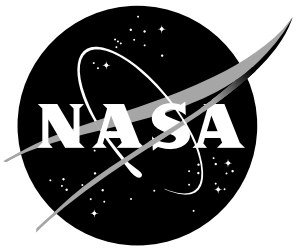
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI Program, see the following:

- Access the NASA STI program home page at `http://www.sti.nasa.gov`
- E-mail your question to `help@sti.nasa.gov`
- Phone the NASA STI Information Desk at 757-864-9658
- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM-2020-220439



Machine Learning For Planetary Mining Applications

Joshua Cook
Oregon State University, Corvallis, Oregon

Jamshid Samareh
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

January 2020

Acknowledgments

The authors would like to thank Charles Liles (NASA Langley) and Pete Lillehei (NASA Langley) for their comments. The NASA Langely Internship Program funded the first author.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

Robotic mining could prove to be an efficient method of mining resources for extended missions on the Moon or Mars. One component of robotic mining is scouting an area for resources to be mined by other robotic systems. Writing controllers for scouting can be difficult due to the need for fault tolerance, inter-agent cooperation, and agent problem solving. Reinforcement learning could solve these problems by enabling the scouts to learn to improve their performance over time. This work is divided into two sections, with each section addressing the use of machine learning in this domain. The first contribution of this work focuses on the application of reinforcement learning to mining mission analysis. Various mission parameters were modified and control policies were learned. Then agent performance was used to assess the effect of the mission parameters on the performance of the mission. The second contribution of this work explores the potential use of reinforcement learning to learn a controller for the scouts. Through learning, these scouts would improve their ability to map their surroundings over time.

1 Introduction

Extended missions on the lunar surface or human missions to Mars will require vast resources and sustain high costs due to the difficulty of transporting resources from Earth's surface. Studies have found that the lunar regolith contains large quantities of resources such as water, oxygen, helium-3, hydrogen, carbon, nitrogen, and other useful elements. Further studies have analyzed the cost effectiveness of in-situ-resource utilization (ISRU) on the Moon in comparison to transporting resources from Earth. While the cost of mining these resources would be higher than that of mining on Earth, the cost of transportation, for the ISRU option, would be significantly lower. Cost savings from transportation outweigh the added cost of mining on the lunar surface [1, 2].

For mining to be efficient on the Moon, new mining technologies must be created. Human based mining would prove to be difficult for a variety of ergonomic, safety, and logistic issues. Robotic mining systems would not suffer from these problems and could potentially provide an efficient solution to mining. Robotic mining of lunar resources could be split into three workloads. One group of robots would scout the terrain for resources, while simultaneously generating a resource density map. The second group of robots would harvest resources at various locations from the scouted map. The last set of robots would collect and process resources [3].

This work focuses largely on the task of scouting. In its simplest form, scouting consists of moving to a location, sampling resource concentration of the regolith, then recording the resource concentration at that location. This task can be easily distributed amongst a team of scouts to increase the search area and increase robustness to system failures. One difficulty associated with this approach is writing controllers for this multiagent system of robots. Examples of difficult behaviors include failure handling, inter-agent cooperation, and handling unforeseen problems. One potential solution is machine learning, specifically reinforcement learning. Reinforcement learning allows the scouts to learn their control policy instead of prescribing one. This work seeks to determine the extent in which reinforcement learning can be applied to robotic scouting and mapping. One application involves using machine learning to assess the impact of various mission parameters on the scouts' performance. Another application of machine learning involves the learning of a controller given few experiences.

2 Background

An agent is an entity which can view its environment via some state representation and then take an action to interact with its environment. The action taken by an agent is determined by its control policy [4]. After taking an action, the agent receives a reward, R, which acts as a metric, describing the agent's performance in completing a desired task. The goal of the agent is to take an action which causes the agent to receive the highest possible reward. Reinforcement learning is a subfield of machine learning in which an agent learns to take better actions to more effectively achieve its goal [5].

A Markov Decision Process (MDP) describes the nature of the transition from one state to the next. An MDP satisfies the Markov property which states the next state only depends upon the previous state and the action taken at the previous state. As a result, the only information needed to make an optimal decision is the state representation at any point in time. This allows for a memoryless policy implementation in which the agent directly maps the state space to an optimal set of actions. This direct mapping from state space to action space greatly simplifies the learning problem for the agent [5].

Sparsely rewarded problems represent a large class of problems in reinforcement learning. In these domains, a majority of the rewards received are constant and uninformative. Problems which can be described by binary task completion or large time delays between rewards, such as robotic exploration, fall into this category. One of the benefits of simplifying a task to a binary completion is that the reward function directly promotes achieving the overarching goal instead of a parallel one. An example would be teaching a bipedal robot to walk from point A to B. Rewarding the robot for reaching point B would almost guarantee that the robot learned to walk to point B. Conversely, if the robot was rewarded at each time step for its velocity towards B, it may not learn to walk to point B. Instead, it may learn to dive in the direction of B, as diving is faster than walking. The trade-off of using a sparse reward is that there is less information to learn from. As a result, the agent may experience difficulty in obtaining a reward or may learn slowly, due to the lack of information provided by the reward signal [6].

Multiagent systems are systems in which multiple agents must interact with each other to receive a shared goal or competitive goals. With a shared goal, a shared global reward is necessary to describe the collective performance. If individual rewards are used, then agents may act selfishly and maximize their individual reward. This competition can cause agents to learn selfish policies, which results in less optimal global performance. An example would be a team learning to play soccer. If each player was rewarded individually for scoring a goal, then they would learn to never pass because it would not benefit them individually. Conversely, if each player is awarded points when the team scores a point, they will be incentivized to work together to score points. A major drawback of using a global reward is referred to as the credit assignment problem. The global reward only provides each agent with the team's reward, but not their individual contribution to the reward. As a result it is difficult to determine which agent credit should be assigned due to the vagueness of the global reward. This creates difficulty in learning as each agent needs individual feedback to learn individual control policies [7].

3 Domain

In this domain, a team of scouts is used to generate a discrete grid-map of a simulated resource distribution as shown in figures 1 and 2. It is assumed that the agents have imperfect knowledge of resource locations through satellite data, but need to collect soil samples to confirm. Agents are capable of continuous control over their locomotion and have the ability to drill and sample the soil below them

for the resource concentration. This satellite data is not accurate and will only guide the agents' search. Resource distributions are irregular with varying resource concentration gradients and shapes.

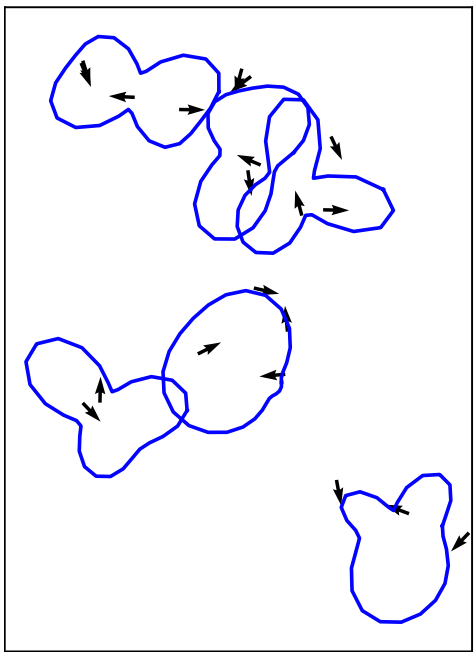


Figure 1. A representation of the scouting domain. Arrows represent rover positions and headings. Blue boundaries represent edges of resource clusters.

Each agent is represented by a tuple of their x position, y position, and orientation angle. The agents receive a state input consisting of distances to the centers of the nearest three resources, distance to nearest agent, location of self, whether or not they have drilled in the current grid location, and distance to nearest frontier. In this scenario, a frontier is defined as an unchecked grid location adjacent to a location that has been found to contain a resource.

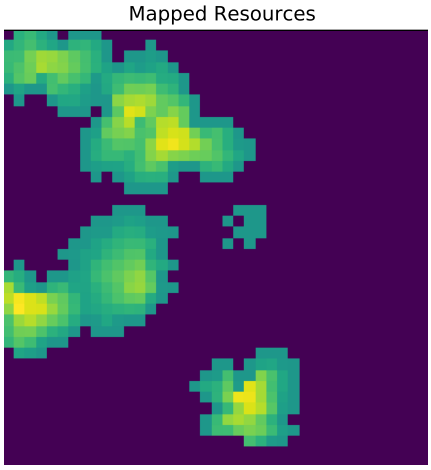


Figure 2. The map generated by the agents of the resource distribution shown in figure 1. Yellow indicates higher resource concentration, green represents areas of low concentration, and blue represents unmapped areas.

The low-level actions the agents can take consist of varying degrees of forward thrust, turning, and the ability to determine resource contents in the regolith below. To reduce the complexity of learning, a set of discrete high-level actions were abstracted from the low-level ones. The high-level action set consisted of moving to one of several nearby suspected resources, moving to a nearby frontier, or drilling at the current location. The global reward function should encourage the agents to discover as many locations containing resources as possible, with higher concentrations of resources given a higher priority. To accomplish this task, the reward function is defined as the sum of the resource concentrations found at each grid-map location, $\hat{C}$. This sum is then divided by the sum of resource concentrations at each location, C, depicted in equation 1. The resulting reward represents an approximate percentage of the resource map to be filled in.

$$R_{global} = \frac{\sum_{i=0}^{n} \hat{C}_i}{\sum_{i=0}^{n} C_i} \tag{1}$$

This reward is received at the end of the episode by each scout on the team, with all other rewards being zero. This ensures that team of agents prioritize discovering as many high density resource locations as possible. This reward function is also designed to encourage agents to work together and discover resources, instead of racing each other to discover each resource location first.

4 Mission Parameter Testing

Without sending a robot to another planet or moon, difficulty arises in determining which environmental factors or mission parameters affect mission performance. To test these parameters, a simulation is used in place of a physical model. Some parameter changes may require significant changes in controller design. As a result,

it is difficult to compare changes in parameters with changes in mission performance, without modifying the controller. To overcome this obstacle, reinforcement learning was used to learn control policies for each modified environment. The rewards received by these controllers provided a metric describing the effect of environment parameters on the performance of the robots. These rewards were then analyzed to determine the specific contribution of each environment variable towards the success of the rovers.

Multiple learning algorithms were tested including policy gradient, deep Q-network, and neuroevolution [8, 9]. In each case, agents learned a homogeneous policy, with one policy copied to each agent. A homogeneous policy was used to avoid the problem of credit assignment. With one policy, the global reward directly correlates to the single policy's performance, and learning becomes fairly stable. Also, learning one policy significantly reduces the policy search space. As a result, the team policy is easier to learn, but cannot learn more complex interactions (e.g., agents do not develop roles).

Each of the algorithms tested produced similar quality in learned policy and training time. Typically, neuroevolution learns with a much lower sample efficiency because it learns from one reward over an entire episode, instead of one reward per time step. Due the sparseness of the reward, only one non-zero reward was provided each episode. This sparseness impeded the learning of the Q-network and the policy gradient, causing them to learn with a sample efficiency similar neuroevolution. This would explain the similar training times. Also, the sparse reward encouraged agents to find as many resource bearing locations as possible, instead of converging to a local optimum. This may be the reason these algorithms produced policies of similar quality [4].

Neuroevolution was selected for the final mission parameter search as it was quantitatively found to give results with the least variation. In this implementation of neuroevolution, mutation was performed without crossover. Mutation was represented as the perturbation of the weights using a unit normal distribution with a set standard deviation. Mutated weights were chosen at random using a uniform distribution. The crossover method selected was tournament selection with randomly generated tournament pairs. The winner of the tournament moved to the new population and the loser became a mutated copy of the winner. The population of agents evolved for a set number epochs.

4.1 Parameter Search: Agent Focused

The parameter search began by selecting a set of environment variables which may have an effect on the performance of the scouting rovers. The first five parameters included the number of agents used, the probability of satellite data being accurate, the probability of sensor failure, the probability of an agent experiencing total system failure, and agent differentiation in the controller.

The number of agents was chosen as a parameter, because a larger number of rovers would be able to search an area more quickly than a few rovers. Sensor failure was defined as the probability of any given sensor returning a zero instead of the sensed value. This parameter was chosen to determine if highly accurate sensors

would be needed for the control policy to function effectively. Agent failure was defined as the probability of an agent experiencing a critical failure at the beginning of an episode. This was chosen to determine how well agents could complete the task given a portion of the team could not contribute. Satellite data inaccuracy could hinder scout performance, which is why it was chosen as a parameter to test. An option was included to provide an agent with a binary version of their identification number as part of their state input. The motivation behind this test was to determine if including differentiating information as part of the state input would allow agents to learn slightly different policies. If the policies could be slightly different for each agent, a larger control space could be searched in comparison to purely homogeneous policies. This larger controller space should theoretically contain policies which have a higher performance than the purely homogeneous policy.

A summary of the effects of mission parameters on performance can be seen in figure 3. In this figure, it is clear that the number of agents used has the largest impact on performance, with all other parameters showing minor correlations. The inclusion of incorrect satellite data seems to only have a minor impact on the performance of the agent. This is most likely due to the small performance cost of needing to verify that a location contains no resources.

An unexpected result lies in the slight positive correlation between sensor error and performance. This result is sensible if the sensor failure is viewed as a form of noise. Adding sensor noise has been found to generate more robust policies, which tend to achieve higher performance than their noise free counterparts [10]. Agent failure seemed to cause the largest losses. These losses are fairly small in comparison to the relatively large failure rate of 15 percent.

Including agent identification as part of the state input seems to have a greater effect on the performance of the agents with an increasing number of agents. Theoretically, if there is only one agent, adding an agent identifier to its input should not increase its performance as there are no other agents to differentiate from. This is shown as the performance of one agent does not change with the inclusion of an identifier. Conversely, adding identifier codes to many agents could improve their performance. This is also shown in figure 3 with 12 agents displaying a higher performance with the addition to the state input.

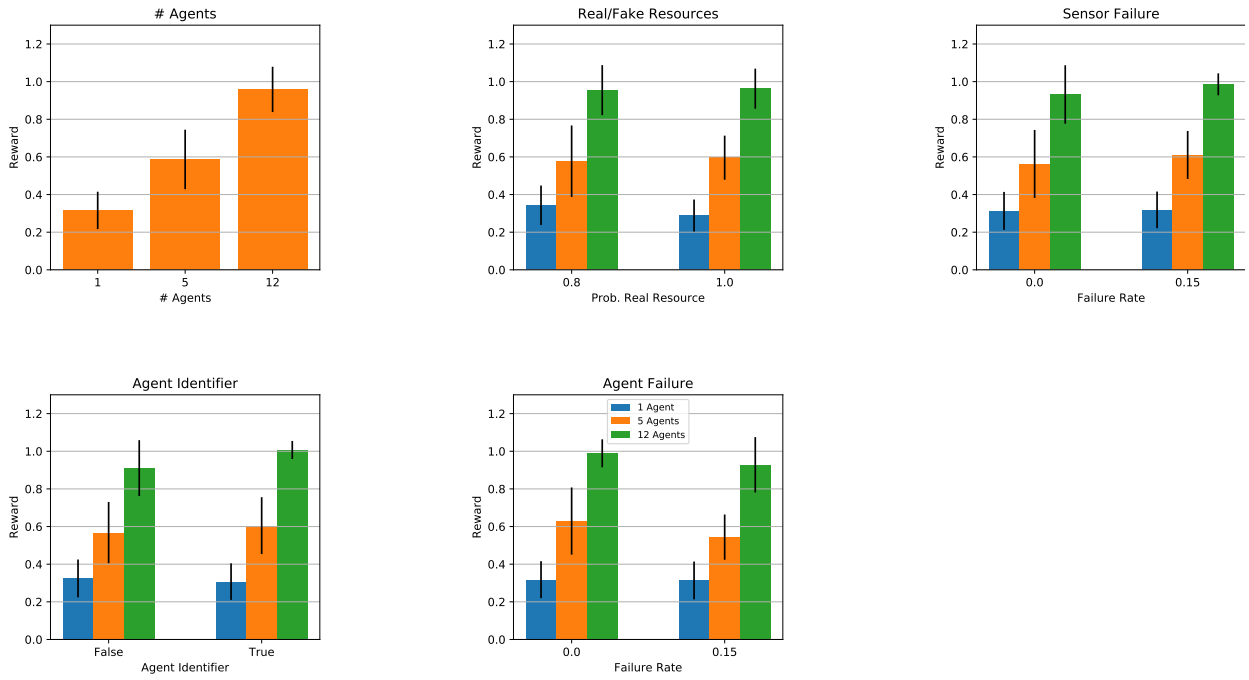


Figure 3. Five plots present the effect of each of the five parameters on mission performance. A $3 * 2^4$ full factorial design was tested with 5 trials per test. Each plot presents the results, grouped by parameter and number of agents. It should be noted that the failure rate was not included for the single agent cases. This explains the similarity in the agent failure plot in the single agent cases.

A linear regression model was used to predict agent performance given the mission parameters. The coefficients of this model are included in figure 4. The purpose of this figure is to quantify the effects of the relationships presented in figure 3. Similarly to figure 3, this figure shows that the number of agents have the largest effect on mission performance.

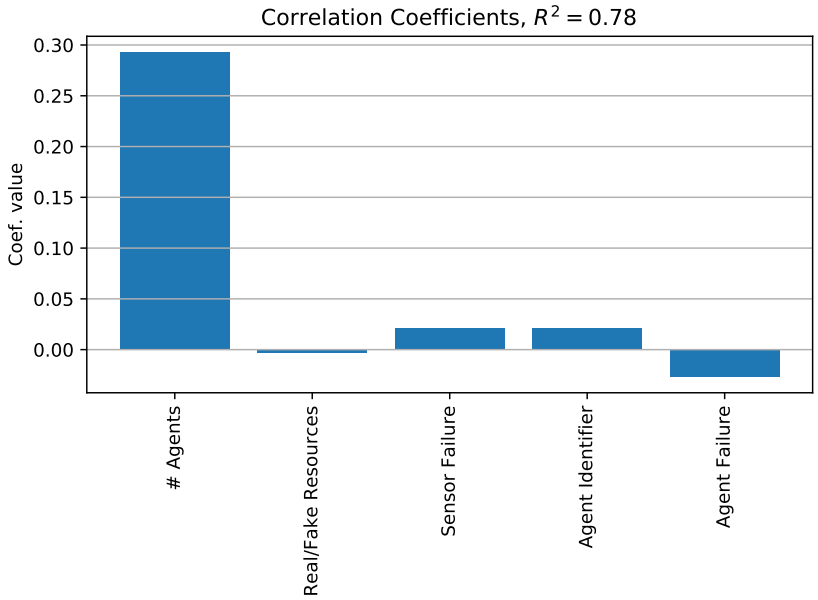


Figure 4. Correlation coefficients of linear regression plotted. This model described the relationship between the five parameters and reward received. Note that number of agents contributed the most by a large margin.

4.2 Parameter Search: Agent Identification

Another study continued the exploration of the effect of agent identification on agent performance. A comparison was performed, with and without agent identification, for a varying number of agents from one to twenty. The results are presented in figure 5. These results generally confirm previous results, showing a general trend in improved performance at higher agent population sizes. From this, it is clear that the agents with identification perform better than those without. While homogeneous policies are easier and more stable to train, these results support that heterogeneous polices may be necessary to increase agent performance.

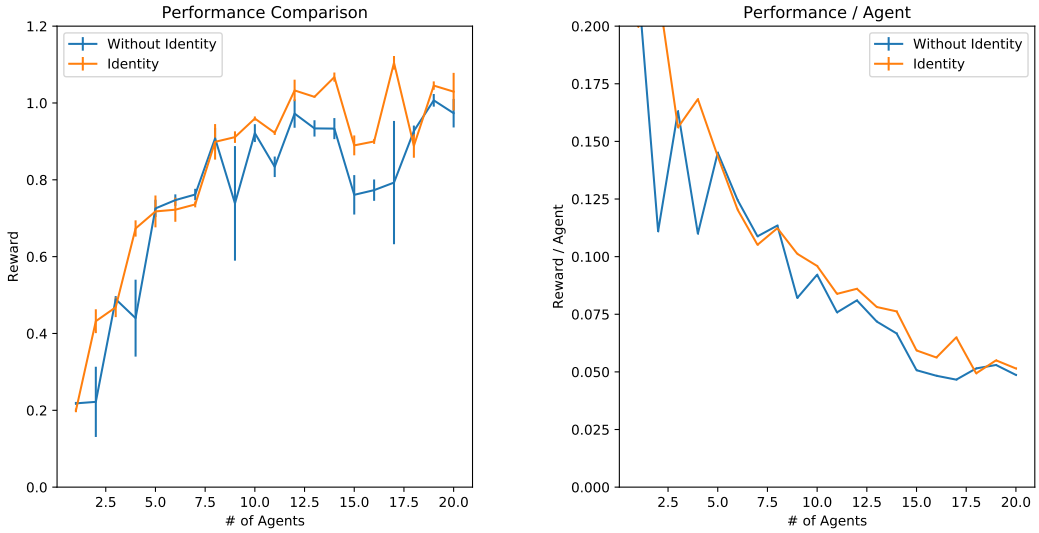


Figure 5. Agent performance (left) and performance per agent (right) as a function of the number of agents. Both plots compare the effects of the inclusion of agent identification on performance. Note the diminishing returns as the number of agents increases.

4.3 Parameter Search: Satellite Data Inaccuracy

A more in-depth study was performed on the effect of erroneous satellite data on agent performance. The goal of this study was to determine the accuracy of the satellite data needed to maintain high agent performance. The intuition behind this test was that if a significant portion of the satellite data was inaccurate, then the scouts would have to spend more time verifying the data instead of finding resource locations. If the data provided was too inaccurate, the scouts' search would no longer be guided and should break down into a form of brute-force search. To test this, five real reasource groups were included in the sattelite data along with a varying number of fake resource groups.

Figure 6 shows the relationship between the number of agents and fake resources and the performance of the team. The plot shows a defined increase in performance as the number of agents increases, with a very minor decrease in performance as the number of fake resources increases. Even as the fake resources outnumber the real resources by a factor of four, the agents seem to map the resources without obstruction. These results show that satellite data might not need to be accurate for agents to successfully map the resource distribution. Even with a data accuracy of 5 real resources groups out of 25 total groups, the scouts are provided enough information to effectively complete the task.

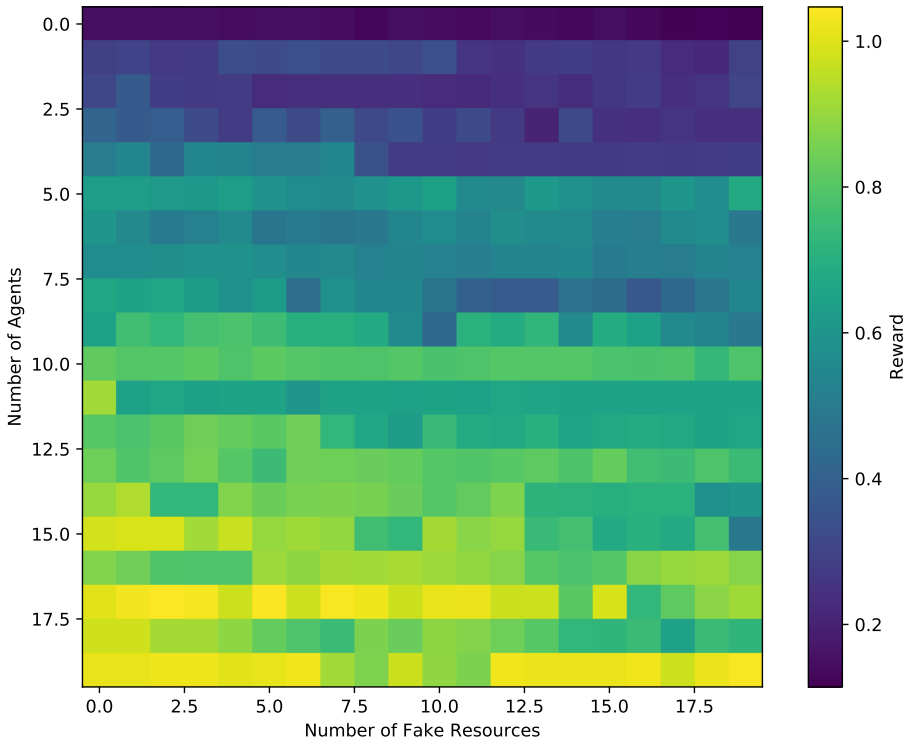


Figure 6. The relationship between number of agents and number of fake resource groups on the reward received is displayed. During these tests, five real resource groups were to be mapped, surrounded by fake resource groups. Yellow indicates high performance while blue indicates low performance.

4.4 Parameter Search: Resource Focused

The last parameter search focused on the effects of the resource distribution on agent performance. The parameters tested include number of agents, the number of time steps in an episode, the spread of the resources, and the size of each resource. In these experiments, spread is defined as the maximum distance from the center of the map to a resource, while size refers to the average diameter of each resource cluster. Theoretically, increasing resource spread should cause the agents to spend more time moving from one resource site to another, thus lowering the total available time for sampling soil resources and increasing the difficulty of the task. If agents are tasked with mapping resources in a given area, then increasing the size of the resource sites should increase the difficulty of the task. As the size of the resources increase, then the total area of land the agents must check also increases. Having a larger area to check, means that the team of agents may run out of time before a majority of resource locations can be found.

The summary of the results of the experiment is shown in figure 7. This plot displays the coefficients of linear regression performed to model the effect of the list

of parameters on mission performance. In this figure, it is clear that both resource spread and size have a strong negative correlation to mission performance. It should be noted that the negative impact of size is much larger than spread. This outcome is intuitive as increasing size increases an area the agents must probe, which scales quadratically, while increasing spread increases travel distance, which scales linearly. Due the quadratic scaling of resource size should generally increase the difficulty of the task more than the linear scaling of resource spread.

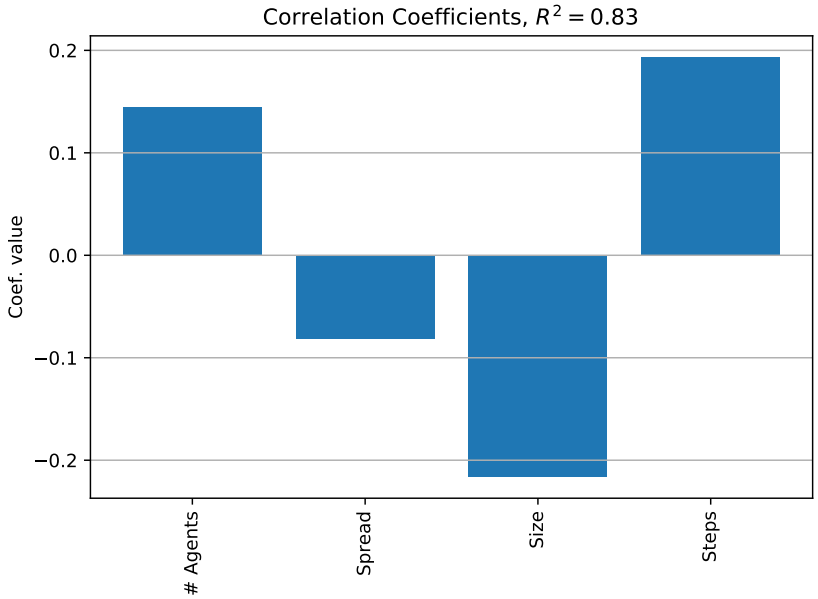


Figure 7. Linear regression was used to determine the relationship between parameters and reward received. Correlation coefficients of the linear regression are plotted to show the correlations between parameters and mission performance.

Figure 8 presents a side by side comparison of the experimental results. From these plots it is clear that increasing the resource size decreases the reward received by the agent. The left portion of the figure shows diminishing returns after around 6 agents. This shows that the problem is fairly easy to solve at a size of 0.5, only requiring six agents to solve. Adding more than 6 agents would provide no additional benefit, but instead increased cost. When the size is set to 2.0 and the spread to 30.0, the agents experience difficulty in solving the problem. It should be noted that in this case, the relationship between reward received and number of agents is roughly linear. As a result, each agent added roughly adds the same value as the agent before it. From this it can be concluded that there are not enough agents used to solve this task and multiple agents could be added past the 20 in this plot before experiencing diminishing returns.

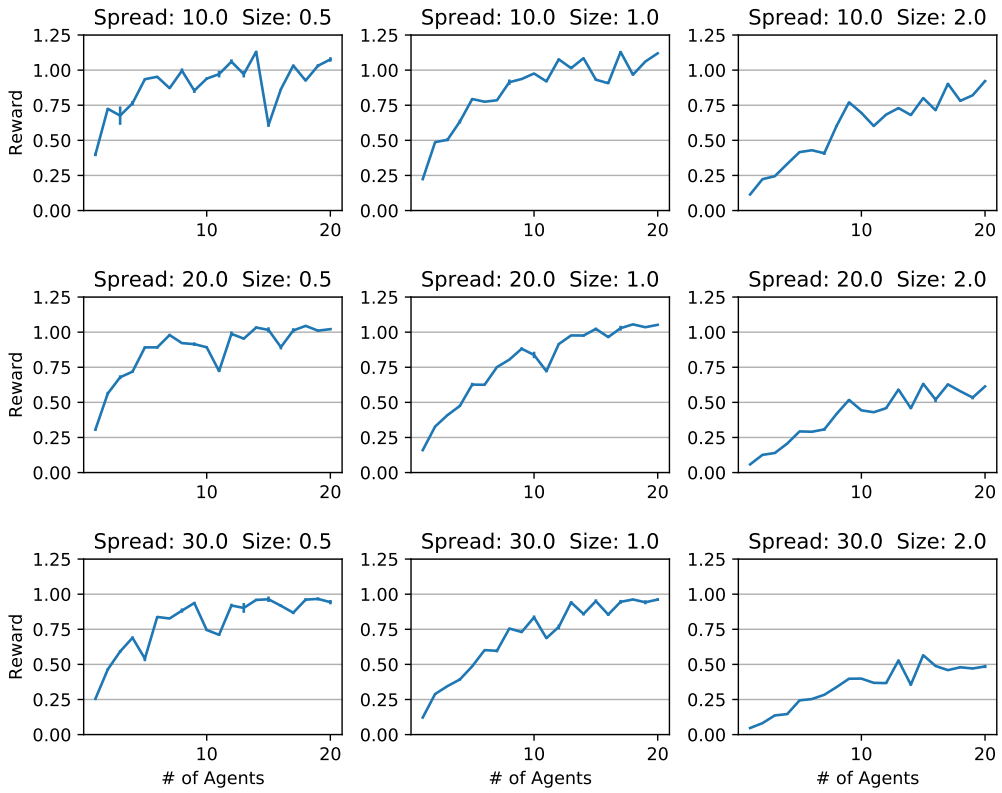


Figure 8. The effect of resource size, resource spread, and number of agents on agent performance is displayed. A $20 * 3^3$ full factorial design was tested to gather this data, with each data point representing the mean of 5 trials.

5 Application of Reinforcement Learning to Physical Scouts

Hand tuning controllers presents a set of challenges in its application to lunar resource scouting. The first issue is the difficulty of programming a controller for an unfamiliar environment. Differences between the testing environment and the physical environment can cause the scouts to perform suboptimally or produce fatal errors. Another potential issue is the handling of unforeseen experiences. A scout may move forward and lodge a wheel between a few rocks. The scout may need to reverse its movement to free its wheel, but its programming continues to attempt to move the scout forward. In this scenario, the robot is clearly taking a poor action, but cannot update the controller to improve the action it takes. Reinforcement learning would allow the scout to learn from its mistakes and perform a low-level of problem solving. This type of learning would also allow the scout to learn to better interact with its environment, so that differences in its simulation do not hurt its performance. The idea of reinforcement is appealing, but its application can be challenging.

5.1 Sample Efficiency

Humans have the extraordinary capability of learning to complete a task given only a few experiences. An example of this would be the game of Pong. After only playing a few rounds, a human understands the general goal of the game and can play at a reasonable skill level. Current reinforcement learning algorithms can also successfully learn to play a game of Pong, but will require thousands of games of experience [8,11,12]. In reinforcement learning, the amount of information an agent learns from an experience is referred to as sample efficiency. Low sample efficiency is not problematic for learning to play games such as Pong because thousands of games can be simulated every second, thus generating thousands of samples to learn from. In robotics this is usually not the case. For example, a robot learning to map resources may take a few hours to a few days to find a new resource. Using these sample inefficient algorithms, it may take the rover weeks to learn a good control policy. The sparsely rewarded nature of this domain requires high sample efficiency in order to learn a control policy in a reasonable amount of time.

Sparsely rewarded domains present their own challenge to sample efficiency. In these domains, the majority of the rewards an agent receives are constant and uninformative. The agent experiences difficulty learning from these experiences because the reward does not provide much information on its performance. In an episode of learning, only a handful of informative rewards may be provided to the agents. With so few informative samples, sample efficiency becomes important if the agent is to learn a control policy quickly.

Reinforcement learning algorithms can be divided into two categories: model-free and model-based. Model-free algorithms perform a direct policy search, essentially learning via trial and error. These algorithms make small changes to the policy and determine how the changes affect the reward received. Due to the trial and error nature of these algorithms, they tend to converge slowly to a local minimum. They also do not perform well in sparsely rewarded domains, such as our scouting domain, as they require frequent feedback. Model-based algorithms learn to model their environment, then learn to take optimal actions from information from their model. These algorithms tend to be more sample efficient, but not efficient enough due the large number of samples needed to model the environment. However, with this model, search becomes easier. Thus, they are better at learning out of local minima and tend to produce higher performing policies. A model based algorithm would be the best approach for the scouts, but it would be difficult for the scouts to learn an accurate model of their environment quickly.

One recently proposed method of improving sample efficiency is the use of a hindsight experience replay buffer. This algorithm allows an agent to learn from experiences which provide no reward, by setting incremental goals from which to learn. This approach proved to be significantly more sample efficient than previous deep reinforcement learning algorithms, but two issues arise. The first issue is for an agent to learn successfully, clear goal states must be present. A bipedal robot tasked with walking from one point to another has a clear state which it must be in to receive a reward. In the case of the mapping rovers, there is no clear goal state which can be defined. The second issue is that while this algorithm is more sample

efficient, it is not sample efficient enough. This algorithm still requires hundreds of experiences to learn from, which the scouts do not have.

Soft actor critic was presented as a more sample efficient actor critic method, and it proved to be significantly more sample efficient than previous deep reinforcement learning algorithms. One example of this improvement in sample efficiency was shown in a quadrupedal robotic locomotion. In this work, the robot was trained to walk in a span of 2 hours. This significant improvement is not enough for the rover domain however. While the training of the quadruped took only 2 hours, 400 episodes passed. An episode for a rover could take hours to days. Using this algorithm, 400 episodes would cost too much of the agents' time [13].

5.2 Simulation-to-Real Learning

One promising area of applied reinforcement learning to robotics is simulation-to-real learning. This method involves generating a fairly accurate simulation of a robotics system, then training the robot in the simulation. The simulation allows the robot to experience a learning episode in a much shorter time frame. With this, thousands of simulations can be run in a short time. With thousands of examples to learn from, sample efficiency no longer becomes an issue. The largest issue that arises with this method is the transfer of the policy to a real robot. Discrepancies between the simulated model and the actual physics of the robot can cause the agent to learn a policy which performs well in the simulated system, but performs poorly in the physical system. One popular method to compensate for this is to add noise to the simulation to encourage the agent to learn a more general policy [14, 15].

The largest drawback associated with these methods is that very little training is performed after the policy leaves the simulation. This is problematic in domains where an accurate simulation of the robot cannot be constructed. Differences between the physical and simulated environments would lead to poor policy transfer to the physical robot. The policies also do not perform well when they encounter new obstacles for the first time in the real world. These are issues the scouts would face, due to the difficulty of generating a realistic simulation of their environment. This problem could be solved if the agent learned to overcome the discrepancies between the simulated and physical environments.

5.3 Future Work

A potential solution to this problem would be a form of transfer learning. A large portion of a human's ability to learn with such a high sample efficiency is due to transfer learning. A human often quickly learns to complete a task by comparing it to a similar task or a task with similar system dynamics. This type of learning would be directly applicable to the scouts. The scouts could learn to model the environment in a simulation, then learn a control policy from this model. The model and policy would then be transferred to the robot, where the robot would continually update the model to better reflect the physical system. From this model, the agent could learn a control policy which better performs in the physical system. This solves the problem of sample efficiency by performing a majority of the learning

in the simulation. Outside of the simulation, the model and policy should only need minor adjustments to reflect the real world dynamics.

6 Conclusions

Robotic mining of lunar resources has the potential to reduce the cost of extended missions on the Moon and to Mars. This robotic mining would be accomplished by teams of robots, grouped into three work types. This work focused on the resource scouting robots and the application of reinforcement learning to their mission success.

The mission parameter search began as a series of mission parameters. A series of simulations were generated with these modified parameters. Due to the differences in environments, a single control policy could not be used for each case. As a result, neuroevolution was used to a control policy for each environment. The performances of these policies were used to then determine the impact of the parameter change on mission performance.

At the time of this writing, there are very few algorithms which would allow the scouts to learn to map the resources in a reasonable time span. This is largely due to the issue of low sample efficiency in many modern deep reinforcement learning algorithms. Learning in a simulated environment, then transferring the policy to a physical robot, presents a promising solution, but is not entirely practical. One possible solution to this problem would be to have an agent learn to model the system in a simulation, then use transfer learning to accelerate the agent's ability to model the real system. This model could then be used to quickly learn a viable control policy.

References

1. Mendell, W. W.: Lunar bases and space activities of the 21st century. 1985.

2. Blair, B. R.; Diaz, J.; Duke, M.; Lamassoure, E.; Easter, R.; Oderman, M.; and Vaucher, M.: Space resource economic analysis toolkit: The case for commercial lunar ice mining. *Final report to the NASA Exploration Team*, 2002.

3. Freitas, Jr., R.; and Zachary, W.: A self-replicating, growing lunar factory. *4th Space manufacturing; Proceedings of the Fifth Conference*, 1981, p. 3226.

4. Russell, S. J.; and Norvig, P.: *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

5. Sutton, R. S.; and Barto, A. G.: *Reinforcement learning: An introduction*. MIT press, 2018.

6. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, O. P.; and Zaremba, W.: Hindsight experience replay. *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

7. Wooldridge, M.: *An introduction to multiagent systems*. John Wiley & Sons, 2009.

8. Volodymyr, M.; Kavukcuoglu, K.; Silver, D.; Graves, A.; and Antonoglou, I.: Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.

9. Such, F. P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K. O.; and Clune, J.: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

10. Peng, X. B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P.: Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

11. Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D.: Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

12. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P.: Trust region policy optimization. *International conference on machine learning*, 2015, pp. 1889–1897.

13. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al.: Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

14. Tan, J.; Zhang, T.; Coumans, E.; Iscen, A.; Bai, Y.; Hafner, D.; Bohez, S.; and Vanhoucke, V.: Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

15. Lee, J.; Hwangbo, J.; and Hutter, M.: Robust Recovery Controller for a Quadrupedal Robot using Deep Reinforcement Learning. *arXiv preprint arXiv:1901.07517*, 2019.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-01-2020	Technical Memorandum	

4. TITLE AND SUBTITLE

Machine Learning For Planetary Mining Applications

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Joshua Cook, Jamshid Samareh

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

954879.02.01.23.01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, Virginia 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L–21094

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM-2020-220439

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category- 91
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Robotic mining could prove to be an efficient method of mining resources for extended missions on the Moon or Mars. One component of robotic mining is scouting an area for resources to be mined by other robotic systems. Writing controllers for scouting can be difficult due to the need for fault tolerance, inter-agent cooperation, and agent problem solving. Reinforcement learning could solve these problems by enabling the scouts to learn to improve their performance over time. This work is divided into two sections, with each section addressing the use of machine learning in this domain. The first contribution of this work focuses on the application of reinforcement learning to mining mission analysis. Various mission parameters were modified and control policies were learned. Then agent performance was used to assess the effect of the mission parameters on the performance of the mission. The second contribution of this work explores the potential use of reinforcement learning to learn a controller for the scouts. Through learning, these scouts would improve their ability to map their surroundings over time.

15. SUBJECT TERMS

Lunar Mining, Reinforcement Learning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Information Desk (help@sti.nasa.gov)
U	U	U	UU	23	19b. TELEPHONE NUMBER *(Include area code)* (757) 864-9658