# Optimizing Mars Base Resource Management

Anna Hylbert
*Aeronautics and Astronautics*
*Stanford University*
Stanford, CA
anna05@stanford.edu

Anna Sulzer
*Aeronautics and Astronautics*
*Stanford University*
Stanford, CA
asulzer@stanford.edu

Harry Zhao
*Aeronautics and Astronautics*
*Stanford University*
Stanford, CA
harzhao@stanford.edu

***Abstract*—Efficient resource management for Mars exploration can reduce the cost of a human mission to Mars as well as potentially make such missions a reality sooner. Optimizing resource management and in-situ production on Mars can improve the efficiency of human explorers, minimizing the amount of resources that need to be transported from Earth and reducing the amount of unnecessary resources being generated on Mars. We propose a model of a Mars mission as a game-like Markov decision process and a technique to generate an efficient decision-making policy using Monte Carlo tree search, with the aim of balancing resource availability with mission progress and efficiency. Our model simulates the production of power, water, oxygen, and food as well as the continuation of (scientific) mission progress, with each of these modeled as different actions which can be taken at a given time step. Unforeseen circumstances such as accidents are modeled stochastically, with a small probability of occurring also at each time step. Compared to a random policy over 100 simulations, our approach using Monte Carlo tree search had a mission completion rate of 91% versus 75% for the random policy, and our approach also reduced the median mission completion time from 64 steps to 29 steps.**

## I. Introduction

Mission planning and mission management for space exploration missions–both robotic and human–require intensive planning and preparation. Mars is widely considered a prime candidate for human exploration after the Moon, but maintaining an adequate supply of resources is a substantial challenge faced by Mars mission planners. We seek to inform this mission design process by proposing a technique by which space missions can be modeled as game-like problems that can be solved using existing methods. While many factors must be accounted for in a real mission, our focus will be on guiding decisions on resource production, allocation, and consumption.

We simplify Mars base resource management into core needs: power, water, oxygen, and food. The approach taken in this paper models the problem of balancing production of resources versus conducting science or making mission progress as a Markov decision process (MDP). We then apply a planning method to improve mission outcomes compared to a random decision policy. This method used to create a policy is Monte Carlo tree search (MCTS), an online planning method which we select due to the large state space of our chosen problem. The goal of the policy is to reduce time to mission completion by optimizing the generation and consumption of our resources. We believe our proposed technique can be beneficial for at least two applications: 1) providing decision-making guidance to astronauts in an ongoing mission and 2) estimating the allocation of resources necessary for a mission in advance.

## II. Related Work

### A. Mars Mission and Resources

Space mission design is a well-explored topic, particularly by NASA and other space agencies. Much research has gone into the quantity of resources which are expected to be consumed over the course of a mission as well as which kinds of resources may be regenerated or obtained at the destination rather than transported from Earth.

In-situ resource utilization will enable the generation of many resources on Mars over the course of a mission. Power will be crucial to the mission, as it is required for generating nearly every other resource. While power may be generated using solar panels or nuclear reactors brought from Earth, resources which can be produced on Mars include water, oxygen, and food. Mars has substantial deposits of subsurface water ice that can be obtained with mining or drilling as well as polar ice caps consisting of water ice and carbon dioxide ice. This water can be a source of oxygen through electrolysis, although new technologies have been tested to extract oxygen directly from the carbon dioxide in the Martian atmosphere, namely the MOXIE (Mars Oxygen In-Situ Resource Utilization Experiment) carried by the Perseverance rover [1]. While MOXIE's primary goal is to prove technology for generating oxygen as rocket propellant, the oxygen it produces can also be used for breathing.

To supply astronauts with food, studies have been performed on the International Space Station to grow food in space, and it is likely that Mars astronauts will have at least part of their diet consist of food grown on Mars, with food proven to grow in simulated Martian soil [2]. The NASA Life Support Baseline Values and Assumptions Document contains detailed quantitative data regarding resource consumption, but we choose to use simplified integer values to model the problem in this paper for simplicity and faster computation (precise values will require a large, continuous state space) [3].

### B. RockSample Problem

We take inspiration from the RockSample problem, in which a rover may receive a reward for collecting rock samples and

by reaching an exit area, which are terminal goal states at which the scenario ends [4]. The rover may choose to move in four directions (up, down, left, right), sample a rock, or perform sensing to estimate the state of a rock. There is a negative reward for every step taken, which encourages the rover to balance exploration (sampling or sensing rocks) with traversing toward the exit area.

We adapt this problem to formulate the design of our problem. Instead of traversing in physical dimensions, we use a model with dimensions representing our mission progress and how many resources we currently have, with terminal goal states in the "mission progress" dimension (i.e., our mission terminates at these goal states when we have performed all of our mission objectives). This problem is implemented as RockSample.jl using the POMDPs.jl package, which we will also use to formulate our problem [5].

## III. Markov Decision Process (MDP)

We model our problem as a Markov decision process, where the future states depend only on the present state. Our states represent the level of resources and mission progress that has been made, and we specify actions to represent the consumption and generation of these resources. We also include uncertainty in the form of mishaps that may occur over the course of the mission, encouraging policies which may be resilient in the case of accidents or unforeseen events. We use a discount factor of 0.95 such that an agent is motivated to reach goal states faster–later rewards are worth less this way.

### A. State Space

The state space of the problem is 5-dimensional, with the state being a vector representing the amount of mission progress, power, water, oxygen, and food. Each dimension has a maximum value and minimum value. The minimum value of each dimension was 1, while the maximum value was 50 for all dimensions except mission progress, which had a maximum of 10. While the original intention was to model the dimensions in proportion to actual resource storage capacities and consumption rates, this discrete, integer-based model with simplified values was used to reduce the complexity of the problem. The maximum values of 50 were selected such that there was enough resolution to the problem while also avoiding an extremely large state space. We initialize our problem at $[1, 10, 10, 10, 10]$, representing initial mission progress plus a reserve of resources to begin the mission. This initial state can be varied to examine the benefits and drawbacks of different resources allocations at the beginning of the problem, but we do not explore this feature in this paper.

Goal states are those where the mission progress value is 10, while terminal non-goal states are those where no action can continue the mission within bounds. An example of a terminal non-goal state would be $[3, 8, 1, 1, 10]$, since no action increases both the quantity of water and oxygen, thus any action would cause at least one of the quantities to exceed the lower bound of 1 (see actions in Section III-B).

### B. Action Space

The action space consists of actions which may reduce or increase the amounts of the various dimensions. The actions represent 5 different decisions: conduct science (increase mission progress), generate water, generate oxygen, grow food, or do nothing. While growing food does not occur as fast as the other tasks in real life, the Markov assumption requires future states to be independent of past ones, so we it is difficult to model future growth of plants based on past actions. We still choose to include food production as an action in order to account for consumption of the relevant resources. Many of the resources (components of the state) require consumption of the other resources in order to be generated. For example, in our problem, growing food causes only 1 unit of power to be generated in a day (power goes up by 1) and consumes 2 units of water and oxygen each in return for 5 units of food. Similarly, increasing mission progress decreases the amount of each resource by one unit. At the moment, the exact number of units in each action is not based on exact quantitative data, as our choice of a simplified state space necessitated tweaking of the actions such that they were balanced.

TABLE I
Rows represent actions, columns represent changes in quantity for each action (MP = mission progress)

| | **Change In Quantity** | | | | |
|---|---|---|---|---|---|
| | MP | Power | $H_2O$ | $O_2$ | Food |
| MP | 1 | -1 | -1 | -1 | -1 |
| $H_2O$ | 0 | 1 | 7 | -1 | -1 |
| $O_2$ | 0 | 1 | -2 | 6 | -1 |
| Food | 0 | 1 | -2 | -2 | 5 |
| Nothing | 0 | 2 | -1 | -1 | -1 |

Each row represents the action, either increasing mission progress or making a resource, and the columns represents the impact on the state. Note that power is positive for many of the actions, as we assume power is generated every day automatically, although the decision of which action to take affects how much power is consumed and therefore how much power is remaining after the action is taken.

### C. Uncertainty

We introduce uncertainty in the problem as various unforeseen setbacks that may occur during the duration of a Mars mission. These are primarily accidents that could happen while astronauts are living on Mars. For example, the Mars base may experience a loss of water supply due to leaking in a storage tank, or the base may experience a loss of power due to an electrical mishap. These events have a small chance of occurring on every step. Other forms of uncertainty include unsuccessful mission progress or even backwards mission progress–experiments could have failed, or data may have been lost. These are modeled in our transition function, which is discussed next.

In our model of the problem, the probability of the base losing its water and electricity were 0.02 each, and the

probability of lost mission progress was set to 0.05. We also have a "do nothing" event, simulating the possibility that weather, communications windows, or other conditions make other actions infeasible. In the event of a power or water loss, the quantity was set to half of the previous value, and in the event of lost mission progress, the amount of mission progress was reduced from the current value by 1 unit.

### D. Transition Function

Our transition function ingests a state-action pair and returns a distribution over possible next states. Within our transition function, we incorporate uncertainty to model the previously-mentioned setbacks. We assign a probability to each possible next state based on the probability of the setbacks, with the probability of "nominal" transition where the next state is generated without setbacks being the difference between 1 and the sum of the setback probabilities. An algorithm representing the transition function is shown.

**Algorithm 1** Transition Function

**function** TRANSITION(s, a)
  **if** not INBOUNDS(s, a) **then**
    $a' \leftarrow a$(do nothing)
    **if** INBOUNDS(s, a') **then**
      $s' \leftarrow s + a'$
    **else**
      $a''$ = VALID ACTIONS(s, a)
      $s' \leftarrow s + a''$
    **end if**
    **return** s' $\sim \mathbb{P}(s' \mid s, a) = 1$
  **else**
    $\mathbb{P}(s' \leftarrow s + a(\text{do nothing}) \mid s, a) = 0.05$
    $\mathbb{P}(s' \leftarrow (s + a) - 0.5s[\text{water}] \mid s, a) = 0.02$
    $\mathbb{P}(s' \leftarrow (s + a) - 0.5s[\text{power}] \mid s, a) = 0.02$
    $\mathbb{P}(s' \leftarrow (s + a) - s[\text{mission}] \mid s, a) = 0.05$
    $\mathbb{P}(s' \leftarrow (s + a) \mid s, a) = 0.86$
    **return** s' $\sim \mathbb{P}(s' \mid s, a)$

In this transition algorithm, a check is performed on whether the state-action pair results in an solution within our dimensional bounds. If it is not, we assign the "do nothing" action to represent a rejection by the astronauts of the decision. If the state with "do nothing" is also not in our bounds, we choose a random valid action. Otherwise, when the original state-action pair is valid, we return a distribution of possible next states, where there is a 0.86 probability that the next state is the expected next state based on the current state and action (sum of current state and action vectors), while the remaining probability is reserved for accidents that can also occur stochastically.

While there are states which are terminal because there is no action for which a valid state-action pair exists for that state, these states cause the game to terminate before another action can be selected, so there will always be at least one valid action when the transition function is called.

### E. Reward Function

The reward function rewards reaching the mission completion goal states with a large reward of $10^{12}$ to encourage reaching these states despite their distance from the initial state. A small negative reward is assigned to most states, while state-action pairs which take the next state out of dimensional bounds results in a larger penalty to discourage the agent from attempting to visit inaccessible states.

**Algorithm 2** Reward Function

**function** REWARD(s, a, s')
  **if** $s'$ is Goal State **then**
    $r \leftarrow 10^{12}$
  **else if** not INBOUNDS(s, a) **then**
    $r \leftarrow -100$
  **else**
    $r \leftarrow -1$
  **end if**
  **return** $r$

Particularly, we want to discourage negative resource quantities, as this is equivalent mission failure due to running out of resources. Our implementation includes terminal behavior such that if a state is reached such that no action results in a valid next state, the game ends. This same terminal behavior is used as well for the goal states such that the game also ends when the goal state is reached.

## IV. Policy

### A. Random Policy Simulation

We use a random policy simulation as the baseline for our problem. We created a solver which randomly selects valid actions from the action space based on the current state. This sometimes results in failure of the mission, either by exceeding the mission step limit (100) or due to reaching a terminal state prematurely (one where no choice of action can continue the mission). The results of the random policy are later compared to our Monte Carlo tree search results in Section V.

### B. Monte Carlo Tree Search

We implement a Monte Carlo tree search (MCTS) to perform online planning and to determine the best action to take for managing the Mars base. While Monte Carlo tree search does not guarantee an exact solution to the problem, it allows us to solve our problem, which has a large state space, with relative efficiency compared to an exact solution method, avoiding exponential complexity through the use of simulations [6].

In our implementation of Monte Carlo tree search, we perform rollouts and replanning at every step. This means that we run rollouts to a specific depth from our current state, take the best action based on the values found by the rollouts, and then transition to a next state. We then repeat this process at that next state, performing a new rollout from the new state and taking yet another action based on the new rollout.

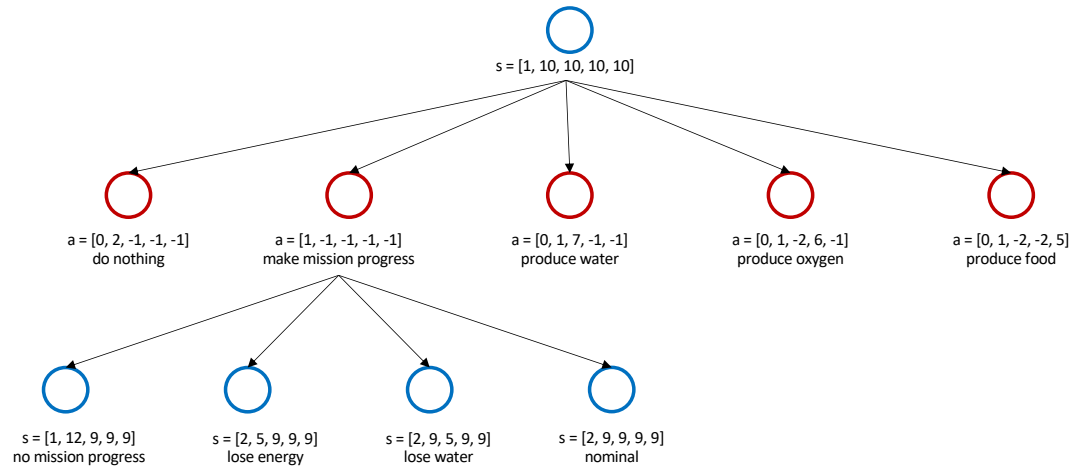


Fig. 1. Example tree with root nodes, possible actions, and possible next states

**Algorithm 3** Monte Carlo Tree Search Policy

$s \leftarrow s_{\text{initial}}$
$i \leftarrow 1$
**while** $i \leq$ steps **do**
  $\pi$ = ROLLOUT(number rollouts, depth)
  $a = \pi(s)$
  $\mathbb{P}(s' \mid s, a)$ = TRANSITION($s$, $a$)
  $s' \leftarrow$ SAMPLE($\mathbb{P}(s' \mid s, a)$)
  $s \leftarrow s'$
  **if** $s$ is terminal state **then**
    **break**
  **end if**
  $i \leftarrow i + 1$

We briefly describe the behavior of this Monte Carlo tree search method. Our method with replanning involves an "outer" loop over which our current state is updated and actual actions are taken as well as an "inner" loop in which rollouts are performed to inform the decision made in the outer loop.

Each rollout begins at a root node, which corresponds to the current state in the outer loop. Depending on the exploration strategy, an action will be selected and there will be a transition to a next state. The UCB1 exploration heuristic is shown in Eq. 1, where $Q(s,a)$ is the action value function, $N(s,a)$ is the state-action pair visit count, and $N(s)$ is the total state visit count. This can be used to encourage taking unexplored actions through an *exploration bonus*, parameterized by $c$ [6]. This continues, sampling into more states until a specified depth that we choose or until a terminal state is reached. In our case, the exploration parameter used is 5 and the depth is 100. We perform 1,000 rollout simulations as part of our tree search each time in the inner loop. At each state, the value of the state (reward) is used to update the action value function. Fig. 1 shows how the tree can expand by taking an action and how a stochastic transition could lead to any of the possible next states.

$$Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \tag{1}$$

Upon completion of the rollout, the results are propagated up the tree, updating the action value function such that we can determine the best action to take in our current state. The update to the action value function can be seen in Eq. 2, which is based on Algorithm 9.6 from *Algorithms for Decision Making* [6]. The updated values can be used as a policy in the outer loop to choose the best action. This action is taken, and then the process repeats in order to replan at the next state. This outer loop continues until we reach the maximum number of steps in the outer loop or until we reach a terminal state.

$$Q(s,a) \leftarrow Q(s,a) + \frac{q - Q(s,a)}{N(s,a)} \tag{2}$$

Our implementation of Monte Carlo tree search and model of the problem as a MDP use the MCTS framework and the QuickMDP interface from the POMDPs.jl package, respectively [5].

## V. Results and Analysis

Running our Mars mission problem over 100 trials for each the random policy and the MCTS policy, we find that MCTS performs far better in terms of mission success rate as well as number of steps to mission completion.

TABLE II
COMPARISON OF POLICY RESULTS

| | MCTS | Random |
|---|---|---|
| Success Rate (%) | 91 | 69 |
| *Completion Time (steps, excluding failed missions)* | | |
| Mean | 30.06 | 65.28 |
| $\sigma$ | 7.79 | 18.12 |
| Median | 29 | 64 |
| 1st Quartile | 24 | 51 |
| 3rd Quartile | 35 | 79 |
| Maximum | 48 | 99 |
| Minimum | 16 | 26 |

MCTS has a success rate of 91% versus 69% for random. MCTS also results in much lower completion times in every statistic, including a mean completion time of 30.06 steps versus 65.28 for the random policy.

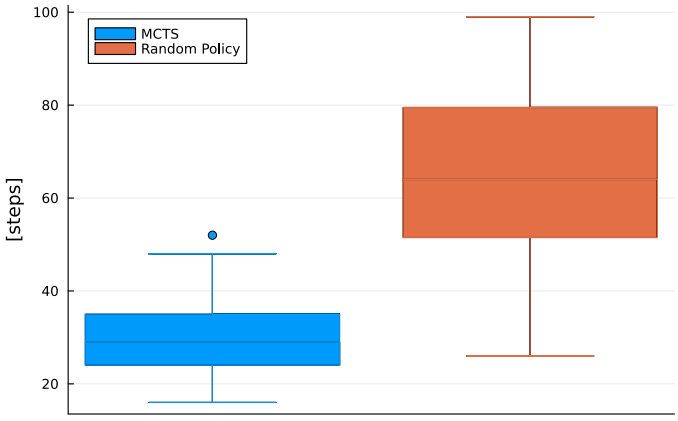


Fig. 2. Statistics of MCTS versus random policy

We also consider the mission history profiles of a few example simulations, illustrating the change in mission progress and resources over the course of the simulation. We first show the mission history of the random policy running our Mars mission problem in Fig. 3.

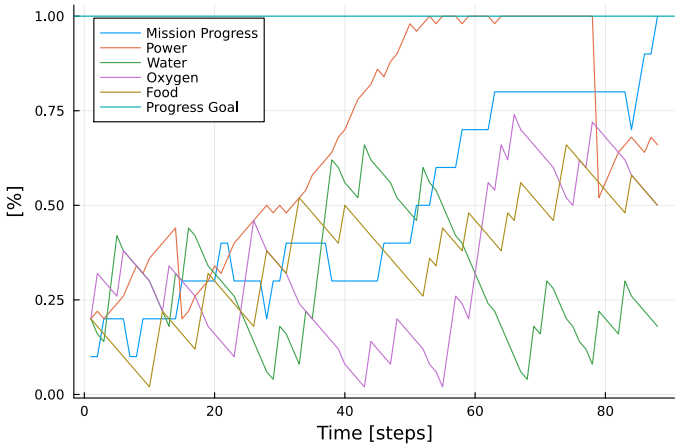


Fig. 3. Random policy mission profile example

The random policy shown takes approximately 90 steps to reach the terminal state for mission progress. Furthermore, at certain times, the policy leads to an accumulation of resources and extended stagnation in mission progress, such as around step 70. While in this instance the policy does manage to reach a terminal state, this is not the case in many other simulations, where the policy fails to either complete the simulation in 100 steps or causes the mission to end up in a state that is no longer valid.

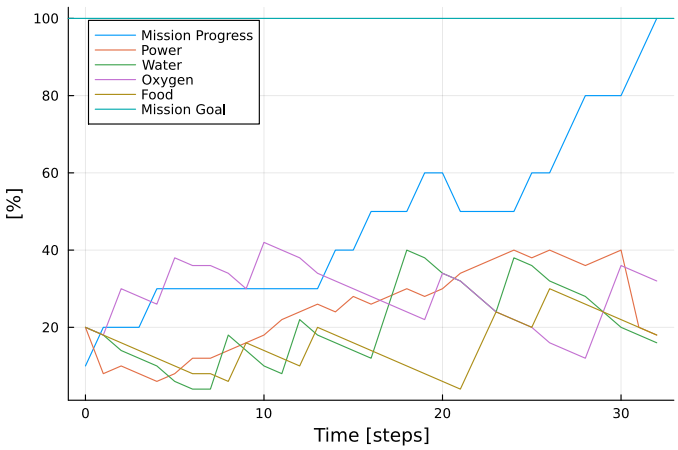


Fig. 4. MCTS policy mission profile example

By comparison, we observe the results of Monte Carlo tree search on the mission problem in Fig. 4. The simulation reaches mission completion in just over 30 steps, and the amount of excess resources of each category is kept to a relative minimum. We can also observe one of the previously-mentioned stochastic events: there is loss of mission progress at step 20. Even with this setback, the results using MCTS manage to complete the mission in less than half the steps of the random policy in Fig. 3.

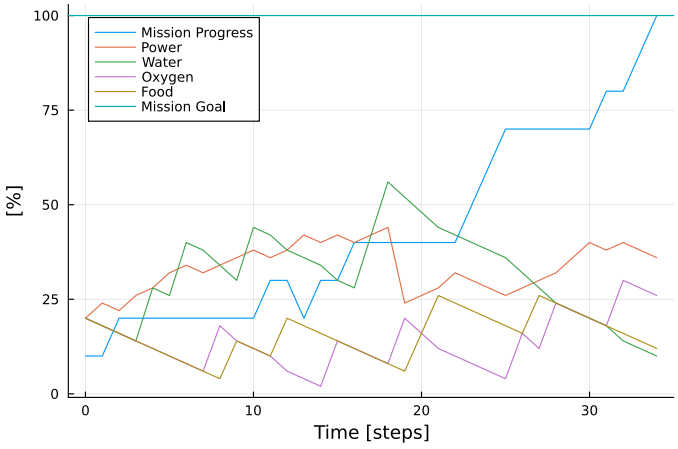


Fig. 5. Another MCTS policy mission profile example

Fig. 5 shows another example of the result of Monte Carlo tree search. In this instance, we observe a different type of setback event, in this case of power loss at step 18, where the power is halved. We also observe mission progress loss at step 12. This example reaches the mission completion terminal state in 34 steps.

## VI. Conclusion

We demonstrated that our application of Monte Carlo tree search to a game-like Mars mission problem is able to provide substantially improved results compared to a random policy, with faster mission completion times. We also demonstrated

that such an algorithm can be more resilient than the random policy when handling stochastic transitions such as the mission setbacks in our problem–our algorithm has a higher mission completion rate (91%) than the random policy (75%).

Opportunities for future work include the application of offline policies such as Q-learning, which may obtain interesting results for this problem. This analysis can also benefit from Monte Carlo tree search using a continuous state space such that more precise values for resource consumption can be modeled. When using a continuous state space for MCTS, variations with progressive widening may be used so that the larger action and state spaces remain manageable. The problem could also be extended to include additional uncertainties, including risks introduced by resupply missions. Resupply missions could be delayed by weather, launch windows, and in-flight failures. Finally, an analysis of different starting conditions for the problem (i.e., different amounts of resources that the mission starts with) may reveal useful results for the planning of Mars missions.

## VII. Contributions

Anna Sulzer modeled the transitions and rewards function as well as the general simulation and plotting environment for the problem. She also simulated and processed the data and created the plots and graphs. Anna also wrote the algorithms for the transition and reward functions.

Anna Hylbert contributed to the abstract, introduction, Mars Mission and Resources, Markov Decision Process (MDP), Monte Carlo Tree Search and Conclusion sections of the report. Anna researched theoretical values to use for the action and state spaces that would make sense for an actual Mars base and also helped troubleshoot the transition model among other parts of the code.

Harry Zhao contributed the RockSample, Transition Function, Uncertainty, MCTS Algorithm, and Results and Analysis sections of the paper. He also performed formatting and final editing of the paper. Harry modeled the problem using POMDPs.jl and implemented the Monte Carlo tree search using MCTS.jl.

## References


[1] J. A. Hoffman, M. H. Hecht, D. Rapp, J. J. Hartvigsen, J. G. SooHoo, A. M. Aboobaker, J. B. McClean, A. M. Liu, E. D. Hinterman, M. Nasr, S. Hariharan, K. J. Horn, F. E. Meyen, H. Okkels, P. Steen, S. Elangovan, C. R. Graves, P. Khopkar, M. B. Madsen, G. E. Voecks, P. H. Smith, T. L. Skafte, K. R. Araghi, and D. J. Eisenman, "Mars oxygen isru experiment (moxie)—preparing for human mars exploration," *Science Advances*, vol. 8, no. 35, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1126/sciadv.abp8636

[2] G. W. W. Wamelink, J. Y. Frissel, W. H. J. Krijnen, M. R. Verwoert, and P. W. Goedhart, "Can plants grow on mars and the moon: A growth experiment on mars and moon soil simulants," *PLoS ONE*, vol. 9, no. 8, p. e103138, Aug. 2014. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0103138

[3] M. S. Anderson, M. K. Ewert, and J. F. Keener, "Life support baseline values and assumptions document," Tech. Rep., 2018.

[4] T. Smith and R. Simmons, "Heuristic search value iteration for pomdps," 2012. [Online]. Available: https://arxiv.org/abs/1207.4166

[5] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, "Pomdps. jl: A framework for sequential decision making under uncertainty," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 831–835, 2017.

[6] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for decision making*. Massachusetts Institute of Technology, 2022.