Designing Low-Thrust Transfers near Earth-Moon L_2 via Multi-Objective Reinforcement Learning

Christopher J. Sullivan*, Natasha Bosanac†
University of Colorado Boulder, Boulder, CO 80303

Rodney L. Anderson‡
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

Multi-objective reinforcement learning is used to uncover a subset of the multi-objective solution space for low-thrust transfers between two L_2 southern halo orbits in the Earth-Moon circular restricted three-body problem. Multiple policies are trained in this scenario to recover transfers while maximizing their reward functions that reflect distinct relative weightings of two competing objectives: minimizing both the time-of-flight and propellant mass usage. These policies are simultaneously trained using an algorithm that is designated as Multi-Reward Proximal Policy Optimization. Evaluating these policies successfully produces transfers between the two orbits with various geometries, propellant mass requirements, and flight times that span a subset of the trade space. These results are also compared to nearby solutions to a constrained optimization problem. A hyperparameter exploration is also performed to determine their influence on the behavior of the trained policies.

Nomenclature

$\hat{A}$	=	estimated advantage function
C_J	=	Jacobi constant
$\boldsymbol{c}$	=	constraints applied to the state at the end of an arc
c_1	=	value error coefficient
c_2	=	entropy coefficient
c_m	=	propellant mass usage coefficient
$\boldsymbol{d}$	=	spacecraft position vector
d_i	=	spacecraft distance to primary body i
E	=	number of epochs

*Ph.D. Graduate, Ann and H.J. Smead Aerospace Engineering Sciences, 3775 Discovery Dr., Student Member AIAA.
†Assistant Professor, Ann and H.J. Smead Aerospace Engineering Sciences, 3775 Discovery Dr., Senior Member AIAA.
‡Technologist, Mission Design & Navigation, 4800 Oak Grove Drive, M/S 301-121, Pasadena, CA 91109, Member AIAA.
The work in this paper also appears in the first author's Ph.D. dissertation titled "Low-Thrust Trajectory Design in Multi-Body Systems via Multi-Objective Reinforcement Learning", submitted in April 2022 to the University of Colorado Boulder [1].

$\boldsymbol{F}(\boldsymbol{X})$ = constraint vector

g_0 = gravitational acceleration measured at the surface of Earth, m/s^2

I_{sp} = engine specific impulse, s

J = objective function

K_i = number of agents for policy i

L = loss function

l^* = characteristic distance, km

l_r = learning rate

M = number of mini batches

m^* = characteristic mass, kg

$m_{s/c}$ = nondimensional spacecraft mass

m_0 = initial spacecraft mass, kg

N = number of policies

$\boldsymbol{N_C}$ = free variable vector describing a coast arc

$\boldsymbol{N_{LT}}$ = free variable vector describing a low-thrust arc

n = number of low-thrust arcs

P = control policy

q = the number of coast arcs along the final periodic orbit

R = probability difference between two policies

r = instantaneous reward

S = entropy

$\boldsymbol{s}$ = state vector input to neural networks

T = engine thrust magnitude, N

t = time step

t^* = characteristic time, s

U^* = pseudo-potential function

$\boldsymbol{u}$ = action vector

$\hat{\boldsymbol{u}}$ = thrust direction unit vector

V = value function

$\boldsymbol{v}$ = spacecraft velocity vector

$\boldsymbol{X}$ = free variable vector

$\boldsymbol{x}$ = spacecraft state vector

γ = discount factor

$\boldsymbol{\Delta d}$ = spacecraft position deviation vector

Δt = nondimensional propagation time

$\boldsymbol{\Delta v}$ = spacecraft velocity deviation vector

ε = clipping parameter

$\boldsymbol{\theta}_i$ = training parameter vector for neural network i

κ = identifies reference path used to calculate state deviation

λ = Generalized Advantage Estimator factor

μ = mass ratio

π = policy function

τ = maximum number of time steps

Ω = bonus or penalty associated with a termination condition

Superscripts and Subscripts

$CLIP$ = clipped objective function

max = maximum value

PG = policy gradient

t = specific time step at which a quantity is evaluated

VF = value error

W = weighted

I. Introduction

Low-thrust transfers within multi-body gravitational systems are of increasing interest for their potential to facilitate new mission concepts. Recent missions such as Dawn, Hayabusa, and Deep Space 1 have demonstrated the capability of low-thrust propulsion systems and have paved the way for additional missions from BepiColombo to Lunar IceCube [2-5]. Based on this precedent, an increasing number of mission concepts use a low-thrust engine to reach destinations beyond low-Earth orbit. Due to limited thrust capabilities and operational constraints, a point solution for a trajectory that delivers the spacecraft to a desired destination is often sought to support feasibility assessments. Yet, simply designing and redesigning a feasible trajectory as the mission parameters evolve may be challenging and time-consuming. Incorporating a low-thrust propulsion system also significantly increases the complexity and dimensionality of the trajectory design space. Furthermore, there are often multiple objectives that must be balanced during design and optimization, motivating a wider exploration of the solution space [6].

One approach to exploring a multi-objective transfer problem is to generate a subset of nondominating solutions

within the global solution space [7]. Recovering a sufficient array of these solutions, however, may be challenging. For instance, classical optimization techniques including direct, indirect, and hybrid optimization techniques typically require an initial guess [8]. Generating these initial guesses may be time consuming and effort intensive, and is compounded when co-states and other non-intuitive variables must be defined [8]. Additionally, some traditional optimization approaches may struggle to either meet objectives and constraints that may not be continuously differentiable or converge on locally-optimal solutions given a poor initial guess [8–10]. Multi-objective genetic algorithms, in combination with an optimization method, have been used to design low-thrust trajectories for interplanetary and multi-body transfers [7, 11, 12]. However, genetic algorithms often require significant computational resources and selecting non-intuitive parameter values [13]. Inspired by its success in other fields, an alternative approach is to autonomously generate a region of the multi-objective trajectory design space using techniques from machine learning.

Deep Reinforcement Learning (DRL), which has been increasingly used within numerous industries, offers one approach to autonomously generating a variety of low-thrust transfers. Recent investigations, for example, have applied DRL to the following trajectory and maneuver design applications within multi-body gravitational systems: Miller and Linares designed transfers between Earth-Moon distant retrograde orbits [14]; Bonasera et al. designed impulsive station-keeping maneuvers for a spacecraft near a Sun-Earth L2 halo orbit [15]; Bosanac et al. designed reconfiguration maneuvers for a starshade [16]; Boone et al. generated orbit transfers for a spacecraft subject to uncertainty [17]; Federici et al. designed low-thrust transfers between planar libration point orbits without an initial guess [18]; LaFarge et al. constructed guidance schemes for spacecraft targeting a nominal trajectory [19]; Zavoli and Federici designed robust interplanetary transfers [20]; and Das-Stuart, Howell, and Folta used fundamental dynamical structures to produce an initial guess for an optimization scheme [21]. Further, Izzo, Sprague, and Tailor and Shirobokov et al. have provided a broader survey of recent applications with additional references [22, 23]. Traditional DRL methods train neural networks to autonomously recover solutions that maximize the cumulative reward, defined using a single scalar reward function, over an entire trajectory [24]. Furthermore, these neural networks may be constructed to encode control policies for a variety of transfers subject to ranges of boundary conditions or model parameters as opposed to a single point solution. As an extension to DRL, Multi-Objective Deep Reinforcement Learning (MODRL) methods simultaneously train multiple policies, each with a distinct reward function, by sharing environmental information across all policies [25]. This approach enables a more efficient use of computational resources when recovering transfers within a region of a multi-objective solution space when compared with multiple, independent runs of a traditional DRL method for a low-thrust, multi-body trajectory design scenario [1].

A Multi-Reward Proximal Policy Optimization (MRPPO) algorithm is presented to leverage the success of single reward Proximal Policy Optimization (PPO) to solve a multi-objective optimization problem. Although a variety of RL algorithms could be used, PPO has been selected due to its favorable convergence properties in chaotic environments and robustness to hyperparameter selection. Specifically, Miller and Linares, Bosanac et al., Bonasera et al., Boone et

al., Federici et al., LaFarge et al., and Zavoli and Federici have used PPO to recover policies that generate transfers for a variety of scenarios, including in multi-body environments [14–20]. More generally, Henderson et al. and Andrychowicz et al. apply PPO to chaotic environments and explore hyperparameter selections [26, 27]. MRPPO uses PPO as a building block within a structure that enables parallel training of multiple policies and introduces information sharing by using the same propagation data gathered from the environment for all policies. Sharing information and, therefore, reducing the number of unique experiences within the environment decreases the required computation time and resources during training because the numerical integration steps that govern the state transitions often command a significant portion of the total training time. Because multiple policies, each with unique objective prioritizations, are simultaneously trained on a single scenario, the required computational resources are decreased, stability is observed in the training process, and a subset of the multi-objective design space may be more efficiently generated. However, like PPO, the policies developed using MRPPO and the performance of the algorithm are influenced by hyperparameter selections [26, 27].

This paper demonstrates an MODRL approach to recovering a segment of the low-thrust transfer design space between two halo orbits near L_2 in the Earth-Moon circular restricted three-body problem (CR3BP). Generating a region of the low-thrust transfer design space involves identifying feasible solutions that also balance minimizing the flight time with minimizing propellant mass usage. To generate these solutions, the transfer design problem is translated into a DRL problem that also supports a successful and efficient training process. Inspired by the concept of a waypoint that is often used in path planning, this DRL problem is formulated using information about the best transfers generated during training. This approach balances exploitation by incorporating the best transfers into the state and reward definitions with exploration by updating these best transfers using training data [28]. Then, MRPPO is used to simultaneously train multiple policies, each with a scalar reward function that reflects a distinct relative weighting between the two competing objectives of minimizing flight time and propellant mass usage. In addition, a hyperparameter exploration is performed using a grid-search approach to examine their influence on the training process and resulting policies in this particular DRL problem. For a baseline set of hyperparameters, the trained policies are evaluated to generate an array of direct transfers with flight times below a specified threshold that span a region of a multi-objective solution space. While generating transfers with a wider variety of geometries and flight times is an interesting area of ongoing research, work by Sullivan et al. and Sullivan demonstrated the recovery of transfers with a variety of flight times and geometries in other scenarios [1, 29]. The validity of these results is examined via a comparison to the nearby solutions to a standard constrained optimization problem. The low-thrust transfers generated by MRPPO in the selected scenario closely resemble the locally optimal solutions, but are generated without reliance on a human domain expert to supply initial guesses for a variety of point solutions. Accordingly, MODRL may offer an alternative and autonomous approach to recovering a subset of trajectories that exist within a multi-objective solution space.

II. Background

A. Dynamical Model

The motion of a spacecraft in cislunar space is modeled in this paper using the CR3BP as a foundation. In this model, a spacecraft of comparatively negligible mass is assumed to be influenced by the gravity of two primary bodies, the Earth and Moon. The primaries are both modeled as point masses following circular orbits about their barycenter. Then, a rotating coordinate system with an origin at the system barycenter and axes $(\hat{x}, \hat{y}, \hat{z})$, is used: the $\hat{x}$ axis is directed from the Earth to the Moon, the $\hat{z}$ axis is aligned with the orbital angular momentum vectors of the primaries, and the $\hat{y}$ axis completes the right-handed triad. Nondimensionalization is also employed via the distance, mass, and time characteristic quantities l^*, m^*, and t^* [30]. A quantity that significantly influences the solution space is the mass ratio, μ that equals the nondimensional mass of the Moon [30]. Finally, the state of a spacecraft is expressed as $\boldsymbol{x} = [\boldsymbol{d}^T, \boldsymbol{v}^T]^T$ in the rotating frame where $\boldsymbol{d} = [x, y, z]^T$ and $\boldsymbol{v} = [\dot{x}, \dot{y}, \dot{z}]^T$.

The equations of motion for a spacecraft in the natural CR3BP are augmented to capture both the additional acceleration imparted by a low-thrust engine as well as the resulting evolution of the spacecraft mass. In this paper, the spacecraft is assumed to be equipped with a variable thrust, constant specific impulse engine. Under this assumption, these equations of motion are expressed nondimensionally in the rotating frame as

$$\ddot{x} - 2\dot{y} = \frac{\partial U^*}{\partial x} + a_x \qquad \ddot{y} + 2\dot{x} = \frac{\partial U^*}{\partial y} + a_y \qquad \ddot{z} = \frac{\partial U^*}{\partial z} + a_z \qquad \dot{m}_{s/c} = \frac{-Tt^*}{I_{sp} g_0 m_0} \tag{1}$$

where

$$U^* = (1/2)(x^2 + y^2) + (1-\mu)/d_1 + \mu/d_2 \qquad d_1 = \sqrt{(x+\mu)^2 + y^2 + z^2} \qquad d_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2} \tag{2}$$

and $g_0 = 9.81$ m/s^2 [31]. The acceleration imparted by the low-thrust engine is calculated in the rotating frame as

$$\boldsymbol{a} = a_x\hat{x} + a_y\hat{y} + a_z\hat{z} = \frac{Tt^{*2}}{1000 m_{s/c} m_0 l^*}\hat{\boldsymbol{u}} \tag{3}$$

for the unit vector $\hat{\boldsymbol{u}}$ expressed in the rotating frame. The final equation in Eq. 1 reflects the change in the spacecraft mass due to propellant mass flow. When $T = 0$, Eq. 1 reduces to the equations of motion for the natural CR3BP.

The natural CR3BP admits a constant of integration and a variety of fundamental solutions that are useful in the trajectory design process. The Jacobi constant, $C_J = 2U^* - \dot{x}^2 - \dot{y}^2 - \dot{z}^2$, is an energy-type parameter that remains constant along a natural trajectory in the CR3BP [30]. This quantity is valuable for describing the relative energy of states in the CR3BP. In addition, five equilibrium points, denoted L_1-L_5, in the Earth-Moon CR3BP each admit periodic orbit families [30]. One family of periodic orbits of significant interest in the space community is the Earth-Moon

L_2 southern halo orbit family; for example, a subset of highly inclined members with a constant line-of-sight to the Earth and extensive coverage of the lunar surface are currently of interest in the identification of mission orbits for the Lunar Gateway [32]. In this paper, two halo orbits with Jacobi constants of 3.11 and 3.07 and moderate inclinations are selected as the initial and final orbits, respectively, for each low-thrust transfer.

B. Deep Reinforcement Learning

DRL has been increasingly used within the astrodynamics community to solve optimization problems in the CR3BP [15–19, 28, 29]. In this approach, data gathered by an agent interacting with an environment is used to train a policy that maps states to the actions that maximize the value function, mathematically capturing the expected future reward at every state [33]. One type of DRL algorithm uses two deep neural networks in an actor-critic structure to each approximate the policy and value function, thereby aiding policy convergence and computational efficiency while training [33, 34]. One method for training the actor-critic neural networks that has demonstrated advantageous convergence properties in chaotic environments is PPO, which recovers a single policy per training session [26, 27, 35].

1. Feed Forward Neural Networks

Neural networks approximate nonlinear mappings between inputs and outputs via a layered structure resembling the composition of neurons in a brain [36]. Feed forward neural networks accept the state of the agent into the first layer, which is fully connected to the proceeding layer with each connection assigned a weight [36]. Then, the nodes in the first hidden layer take in each of the weighted inputs, add a bias, and feed the result through an activation function, e.g., the hyperbolic tangent (Tanh), to add nonlinearity to the function approximation [37]. For deep neural networks, this process is continued through multiple hidden layers, which often requires fewer nodes per layer to recover locally optimal solutions and tends to perform better in complex scenarios than single hidden layer neural networks [26, 27].

The training parameters $\boldsymbol{\theta}$ for the neural network include the weights and biases and are updated throughout training [33]. One approach to updating the training parameters uses a stochastic gradient descent optimization algorithm, which tends to perform well in applications of DRL to noisy, high-dimensional environments [33]. In this analysis, the Adam optimization algorithm is leveraged due to its efficient and well-tested convergence properties across many environments [27, 38]. While feed forward neural networks are rapidly constructed and evaluated, training neural networks may be time-consuming and sensitive to the structure.

2. Actor-Critic Networks

State-of-the-art single objective DRL algorithms often use actor-critic methods to train a policy that maximizes the value function within an environment [33]. This approach uses an actor and a critic that may each be constructed using deep neural networks [34]. The actor neural network learns $\pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{s}_t)$, which aims to produce a locally optimal action,

$\boldsymbol{u}_t$, for an input state, $\boldsymbol{s}_t$, at time t [34]. Meanwhile, the critic estimates the value function for the associated policy function, reward function, and for the states in the environment [33, 34]. The value function for policy π, is written as

$$V^{\pi}(\boldsymbol{s}_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t(\boldsymbol{s}_t, \boldsymbol{u}_t, \boldsymbol{s}_{t+1})\right] \tag{4}$$

where γ discounts future rewards more heavily than current rewards [39]. In this paper, both of the neural networks are initialized with no knowledge about the environment. However, by interacting with the environment via state-action-reward experiences, the neural networks may be trained to produce a locally optimal policy. In addition, separately learning the policy and value function often leads to robust convergence properties in complex environments [34].

3. Proximal Policy Optimization

PPO is a state-of-the-art DRL algorithm that is used in this paper to train deep neural networks in an actor-critic structure to recover a locally optimal policy for a defined reward function. To update the neural networks using PPO, a set number of state-action-reward experiences are gathered from the environment [35]. The advantage of each experience is estimated via Generalized Advantage Estimation (GAE) to determine the state-action pairs that produce more desirable behavior in the environment [35]. This method uses the current reward for a state-action pair and the expected discounted reward from future time steps to calculate $\hat{A}_t^{\pi}(\boldsymbol{s}_t, \boldsymbol{u}_t) = \sum_{\ell=0}^{\infty}(\gamma\lambda)^{\ell}\partial_{t+\ell}$ as the estimated advantage of state-action pair $(\boldsymbol{s}_t, \boldsymbol{u}_t)$ and where $\partial_t = r_t(\boldsymbol{s}_t, \boldsymbol{u}_t, \boldsymbol{s}_{t+1}) + \gamma V^{\pi}(\boldsymbol{s}_{t+1}) - V^{\pi}(\boldsymbol{s}_t)$ is the estimated advantage of $\boldsymbol{u}_t$ [39]. Once the estimated advantages have been calculated for each state-action-reward experience, PPO updates the training parameters using an objective function that prevents destabilization of the networks [35].

PPO uses a soft constraint in the objective function to discourage updates from destabilizing the neural networks, thereby aiding convergence [35]. This objective function includes a term comparing the probabilities of the old and new policies. This quantity is most commonly defined using a ratio of the log-likelihood values that reflect the statistical likelihood of actions being selected by a policy. However, the limitations imposed by machine precision as well as the low probability of actions selected by one policy being selected by a distinct policy cause some of the log-likelihood values to approach zero when used in a multi-reward framework. Thus, the denominator of a probability ratio would approach zero and cause the training parameters to diverge. Motivated by the application of PPO to a multi-reward formulation, this paper compares the old and new policies using the difference, employed by Zhang et al. and Holubar and Wiering, that is calculated as $R_{i,t}(\boldsymbol{\theta}_{i,j}) = \pi_{\boldsymbol{\theta},i,j}(\boldsymbol{u}_t|\boldsymbol{s}_t) - \pi_{\boldsymbol{\theta},i,j-1}(\boldsymbol{u}_t|\boldsymbol{s}_t)$ for policy i at update j, to inhibit statistically improbable actions from derailing the training process [40, 41]. Then, the soft constraint bounds the probability difference to within the range $(-\varepsilon \leq R_{i,t}(\boldsymbol{\theta}_{i,j}) \leq +\varepsilon)$ to discourage the probability difference from diverging away from zero [35]. This soft constraint appears in a term within the objective function that, for state-action pair $(\boldsymbol{s}_t, \boldsymbol{u}_t)$ at time t, is expressed as $L_{i,t}^{CLIP}(\boldsymbol{\theta}_{i,j}) = \hat{\mathbb{E}}_t[\min(R_{i,t}(\boldsymbol{\theta}_{i,j})\hat{A}_{i,t}, \text{clip}(R_{i,t}(\boldsymbol{\theta}_{i,j}), -\varepsilon, \varepsilon)\hat{A}_{i,t})]$ and evaluated using the training data gathered for an

update to the networks [35]. Two additional terms appear in the objective function used by PPO: (a) an entropy term that encourages exploration within the environment and (b) a squared-error loss term that encourages the approximation of the value function by measuring the error of the estimated value function with respect to the actual value computed using the training data [35]. This modified objective function is mathematically written as

$$L_{i,t}^{CLIP+VF+S}(\boldsymbol{\theta}_{i,j}) = \hat{\mathbb{E}}_t[L_{i,t}^{CLIP}(\boldsymbol{\theta}_{i,j}) - c_1 L_{i,t}^{VF}(\boldsymbol{\theta}_{i,j}) + c_2 S[\pi_{\boldsymbol{\theta},i,j}](\boldsymbol{s}_t)] \tag{5}$$

where $L_{i,t}^{VF}(\boldsymbol{\theta}_{i,j}) = (V_{\boldsymbol{\theta}}^{est}(\boldsymbol{s}_t) - V_{i,t}^{act})^2$ is the squared error loss term between the estimated values and actual computed values and $S[\pi_{\boldsymbol{\theta},i,j}](\boldsymbol{s}_t)$ gradually decreases during training as the networks learn more about the environment [35]. This objective function formulation and structure provides relatively stable and efficient performance in complex environments for single objective problems, thereby motivating its use in a multi-objective framework [26, 27].

III. Multi-Reward Proximal Policy Optimization

MRPPO uses the same objective function as PPO in a structure that enables multiple policies, each with distinct reward functions, to be trained simultaneously by sharing state-action propagation information across all policies. The structure for MRPPO is summarized in Fig. 1 with N policies, denoted π_i for $i = [1, \ldots, N]$, each controlling K_i agents and independently acting within the environment. Each policy receives the current state at time t of each assigned agent, and uses the actor neural network to generate the actions. Then, the dynamical model returns the state at time $t + 1$ to generate a state-action transition, $(\boldsymbol{s}_t, \boldsymbol{u}_t, \boldsymbol{s}_{t+1})$. This process is repeated until τ steps have been generated for each policy, with a shared memory object storing all the state-action pairs gathered within the environment since the last policy update. Then, the transitions for one policy are shared with every other policy; thus, each policy learns from an additional $(N-1)\tau$ transitions in the environment. Finally, each policy feeds the $N\tau$ transitions into the assigned reward function to generate the state-action-reward experiences used to update the policy. This process is repeated until a terminating condition is reached; in this paper, termination is defined using a set number of updates.

Information sharing across multiple policies trained using PPO has been demonstrated by Zhang et al. to produce higher total rewards for an environment with a single reward function [40]; the success of this work motivates extending information sharing across policies to an environment with multiple reward functions. Using PPO in a multi-reward architecture is not expected to violate the critical assumptions of PPO. For instance, the formulation of MRPPO assumes that the state, action, and reward information for each policy spans a continuous space; by constructing the reward function using a set of linearly weighted functions, the policies are expected to maximize sufficiently similar objectives in an environment. Additionally, using data generated by multiple policies in MRPPO is not expected to violate any critical assumptions of PPO because policies are trained in PPO by reusing data across multiple epochs and mini-batches and the objective function incorporates clipping [42]. Through this architecture, MRPPO enables a policy to experience

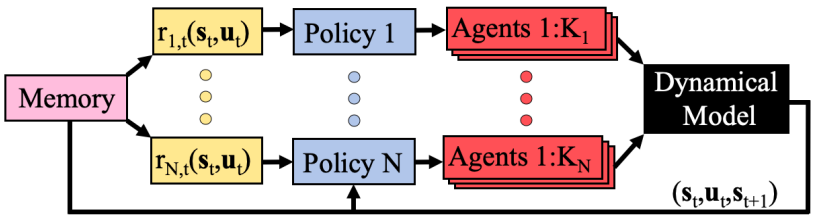


Fig. 1 MRPPO architecture illustrating N policies, each with an assigned reward function $r_{i,t}(s_t, u_t)$, controlling K_i agents subject to the dynamical model.

potentially beneficial actions undertaken by other policies and discounts harmful actions. Because numerical integration often accounts for a significant portion of the training time for a policy in a multi-body system, sharing propagation information also significantly reduces the required computational time and resources. With this formulation, MRPPO is used to train multiple policies with the same training parameters on a scenario to produce locally optimal behavior for each policy and assigned reward function.

The training process and resulting policies are influenced by multiple hyperparameters and construction parameters. Hyperparameters that influence the policies trained by MRPPO include:

- τ: sets the number of state-action-reward experiences gathered using each agent prior to updating the parameters of the neural networks. The total number of state-action-reward experiences used to update the neural networks is a function of both the total number of agents, K, and the number of environmental steps, i.e., $K \times \tau$.
- E: controls how many times the state-action-reward experiences are used to update the parameters.
- M: divides the total number of experiences into smaller datasets, each used to update the neural network parameters. Thus, the total number of updates for a set of experiences is $E \times M$.
- γ: discounts future rewards when computing the estimated value and advantage functions.
- λ: influences calculating the advantages and reduces the variance in the estimated advantage computation.
- l_r: affects how quickly the neural networks are updated from a set of state-action-reward experiences using the Adam optimizer, set equal for both the actor and critic neural networks.
- ε: limits the size of the updates to the networks in the objective function.
- c_1: scales the value loss error in the objective function.
- c_2: determines the amount of entropy in the objective function.

In addition, the number of hidden layers and hidden nodes per layer for both the actor and the critic, which are assumed to possess independent layers, also affect the ability of the policy to learn the optimal behavior [27]. Furthermore, the weight initialization scheme for the neural networks affects the trained policies due to the local basins of convergence that they may fall within; for instance, orthogonal, Xavier, and random techniques may each produce distinct results. Similarly, the activation function used within each layer influences the resulting trained policy. The baseline hyperparameters and construction parameters for MRPPO are manually selected in a simple trajectory design scenario and using insights

from hyperparameter explorations for PPO performed by Andrychowicz et al. and Henderson et al. [26, 27].

IV. Using MRPPO to Recover Trajectories in the CR3BP

MRPPO is used to train policies to design transfers between two L_2 southern halo orbits for a low-thrust-enabled spacecraft in the CR3BP; the results span a portion of the solution space, balancing simultaneously minimizing the propellant mass usage and flight time. To apply MRPPO, this transfer design problem must be translated into an DRL problem. First, the state and action are formulated to support the neural networks learning locally optimal transfers and control histories in the low-thrust-enabled CR3BP. A reward function is formulated to capture the following objectives: targeting the final orbit, minimizing propellant mass usage, minimizing flight time, and penalizing any constraint violations. Each policy is then used to design transfers from initial conditions drawn near a specified departing location along the initial orbit with a reward function that reflects a distinct relative weighting between competing objectives. A set of policies are then simultaneously trained to solve the presented DRL problem from initial conditions near multiple states along the initial orbit and for multiple relative weightings of the flight time and propellant mass objectives.

A. Overview of Transfer Design Problem

In this preliminary proof-of-concept, low-thrust transfers are constructed to balance minimizing flight time and propellant mass usage while connecting two nearby periodic orbits in the Earth-Moon CR3BP. This paper assumes a 180 kg ESPA-class SmallSat [43] equipped with a variable-thrust engine, a maximum available thrust of $T_{max} = 0.25N$, and a constant specific impulse of $I_{sp} = 3000s$. While T_{max} is selected to be significantly higher than current SmallSat technological capabilities, it is similar to the upper bound of the 0.025-0.236N thrust range of NASA's Evolutionary Xenon Thruster-Commercial (NEXT-C) [44]. The NEXT-C engine, with a thruster mass of up to 14 kg and a power processing unit mass of up to 36 kg, was used onboard the 610 kg impactor spacecraft in the Double Asteroid Redirect Test (DART) mission along with 60 kg of xenon [44, 45]. Note that previous investigations by the authors have demonstrated the capacity for MRPPO to recover transfers for spacecraft with a lower maximum available thrust and longer flight times in other trajectory design scenarios [28, 29]. To limit the computational effort that is required in this proof-of-concept, the spacecraft begins at an initial condition near one of only four states along the initial L_2 southern halo orbit with $C_J = 3.11$ and a period of 14.42 days; the selected states are defined in Table 1. Then, perturbations are added to the selected state along the initial orbit to reflect off-nominal initial conditions. Inducing perturbations in the initial conditions also enables the policies to experience a larger portion of the environment and limits overfitting to the exact orbit state. These perturbations are drawn from a Gaussian distribution with a mean of zero and standard deviation of 1×10^{-3} nondimensional units for both position and velocity components, or approximately 384 km and 1 m/s, respectively. The goal is then to design a low-thrust transfer that places the spacecraft in the vicinity of any state along the final L_2 southern halo orbit with $C_J = 3.07$. This orbit is described by a state

Table 1 Selected reference states along the initial 14.42-day L_2 southern halo orbit.

Scenario	Initial Conditions
1	$\boldsymbol{x}_{IC,1} = [1.0855, 0, 0.0626, 0, 0.2735, 0]^T$
2	$\boldsymbol{x}_{IC,2} = [1.1018, 0.1107, 0, 0.0645, 0.0713, -0.1576]^T$
3	$\boldsymbol{x}_{IC,3} = [1.1676, 0, -0.1029, 0, -0.1973, 0]^T$
4	$\boldsymbol{x}_{IC,4} = [1.1018, -0.1107, 0, -0.0645, 0.0713, 0.1576]^T$

$\boldsymbol{x}_{IC,F} = [1.1484, 0, -0.1494, 0, -0.2192, 0]^T$ and an orbit period of 13.81 days.

B. Autonomous Path Planning with MRPPO

Inspired by the use of waypoints in path planning, the DRL problem formulated in this paper uses information about the best transfer generated by each policy during training to improve the training process and the likelihood of policy convergence. Prior investigations in a variety of path planning applications have noted the limitations of DRL algorithms to converge on locally optimal trajectories in complex environments without either a priori domain knowledge or the use of path planning algorithms [46, 47]. To address this challenge, some applications of DRL for path planning have successfully leveraged a set of waypoints that guides an agent through the configuration space while maximizing the reward function; these waypoints may be fixed or iteratively adjusted by the DRL algorithm or a path planning algorithm [46, 47]. These waypoints are often states or position vectors that are defined a priori or autonomously generated. Directly applying this waypoint concept to low-thrust transfer design in a multi-body system would require time and support from a human with sufficient knowledge of the solution space; an expectation that is not always reasonable, particularly in the low-thrust-enabled CR3BP. Rather, this paper builds upon this approach by formulating a DRL problem for transfer design that uses the best transfer generated by each policy during training directly in the state and reward definitions. The best transfer for each policy, labeled within this paper as the policy's 'reference trajectory', is autonomously generated as the agents controlled by that policy explore the environment and updated as the policies learn better control profiles. This approach is empirically observed to facilitate each policy uncovering locally optimal behavior in complex environments while reducing both the bias induced by a human trajectory designer as well as the time and computational resources needed to manually identify a reference trajectory or sequence of waypoints a priori.

For each policy, the reference trajectory is autonomously generated and updated during training. The reference trajectory for each policy is initially undefined. Early in the training process, the reference trajectory is selected from the training data as the trajectory that most closely approaches the target orbit. Then, once a reference trajectory reaches the vicinity of any state along the target orbit, defined in the configuration space with a tolerance of 5,000 km, the reference trajectory is defined as the trajectory generated during training with the highest total reward. Figure 2 displays a conceptual representation of this approach for a single policy designing transfers from a single initial state to any state along the final orbit. In this figure, the initial periodic orbit is represented by the black dashed arc, the final periodic

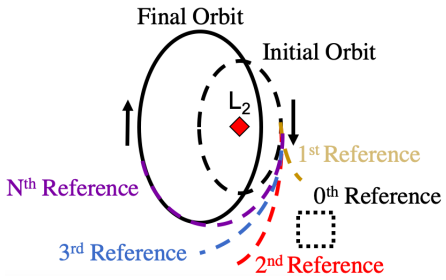


Fig. 2 Updating a reference trajectory for a single policy throughout training in a transfer design scenario.

orbit is denoted by the solid black arc, the undefined initial reference trajectory is displayed as an empty black dashed box, and the reference trajectories are depicted in a variety of hues as they are updated throughout the training process. Given that the transfer design problem is formulated in this proof-of-concept to assign each policy a single reference state along the initial orbit, each policy uses a single reference trajectory. However, ongoing work focuses on using multiple reference trajectories for a single policy that reflect departure from any location along an initial orbit [29].

C. State, Action, and Episode Formulations

Each policy attempts to map a state vector that reflects problem variables to optimal actions that maximize the expected total reward. In this paper, the state vector of the agent is defined as $\boldsymbol{s}_t = [\boldsymbol{d}, \boldsymbol{v}, \Delta\boldsymbol{d}, \Delta\boldsymbol{v}, m_{s/c}, \kappa]^T$. The first two elements reflect the absolute position and velocity vectors of the spacecraft in the Earth-Moon rotating frame. The next two elements reflect the position and velocity deviation of the spacecraft measured from the state along either the policy's reference trajectory or the final orbit that is closest in the configuration space. Accordingly, the value of κ is set as -1 (or 1) if the state deviation is measured from a state along the reference trajectory (or final orbit). However, to prevent chatter when the spacecraft is near both the final orbit and reference trajectory, κ is always equal to 1 once the spacecraft reaches within 5,000 km of any state along the final orbit. Although these position and velocity deviation terms produce a nonminimal description of the spacecraft within the agent state vector, such a description is empirically observed to encourage each policy to design transfers that converge to within a small hypersphere along the final orbit.

The action vector output by the actor neural network reflects the thrust applied by the propulsion system. The action vector is a 4×1 vector defined as $\boldsymbol{u}_t = [u_x, u_y, u_z, u_T]^T$. The first three components of the action vector define the thrust direction in the axes of the Earth-Moon rotating frame while the fourth component determines the thrust magnitude. Once output from the actor neural network, the direction components of the action vector are normalized to produce the unit vector $\hat{\boldsymbol{u}}$ whereas u_T is scaled using the hyperbolic tangent function to guarantee that the value is between -1 and 1. Then, the normalized thrust magnitude is scaled by the maximum thrust magnitude of the low-thrust engine as $T = T_{max}(tanh(u_T) + 1)/2$, ensuring that T does not exceed the maximum available thrust of the engine.

The action is held constant in the rotating frame along a time step and the state is propagated in the low-thrust-enabled CR3BP for $\Delta t = 3 \times 10^{-2}$ nondimensional time units, or approximately 3 hours. This constant time step supplies the

policies with a reasonable amount of time to learn the results of their actions, but not too long to limit changes in the thrust vector direction. Of course, operational and hardware constraints may limit the frequency of thrust vector changes or require slewing constraints; in that case, the time step may be redefined and thrust vector constraints added as needed. In this paper, the maximum episode length is set equal to 100 time steps, which corresponds to a maximum allowable transfer time of 3 nondimensional time units; slightly less than the orbital periods of the halo orbits, which influences the solution geometries recovered by the policies.

D. Reward Function Formulation

To guide the spacecraft from the initial halo orbit to the final halo orbit, the reward function balances conserving propellant mass with minimizing the flight time. The reward function is defined as

$$r_t(\boldsymbol{s}_t, \boldsymbol{u}_t) = \begin{cases} -1 - c_m m_0 (m_{s/c,t} - m_{s/c,t+\Delta t}) + \Omega & \kappa_t = -1 \\ -10|\boldsymbol{\Delta d}_{t+\Delta t}| - |\boldsymbol{\Delta v}_{t+\Delta t}| - c_m m_0 (m_{s/c,t} - m_{s/c,t+\Delta t}) + \Omega & \kappa_t = 1 \end{cases} \tag{6}$$

and depends on whether $\Delta\boldsymbol{d}$ and $\Delta\boldsymbol{v}$, which appear in the agent state vector, are measured relative to the reference trajectory or the final orbit. When $\Delta\boldsymbol{d}$ and $\Delta\boldsymbol{v}$ are measured from the reference trajectory, i.e. $\kappa_t = -1$, the three terms in the reward function, from left to right, encourage: 1) minimizing flight time via a constant penalty of -1 at each time step, 2) minimizing propellant mass usage, and 3) reaching the final orbit. Although the policy is not incentivized to follow the reference trajectory closely, the reference trajectory does still supply information about targeting the final orbit through the position and velocity deviations terms in the state vector. Once the spacecraft enters the vicinity of the final orbit, i.e., $\kappa_t = 1$, the reward function incentivizes the policy to decrease the state discontinuities with respect to the final orbit while also conserving propellant mass. In both cases, the propellant mass usage term is calculated using the most recent action, $\boldsymbol{u}_t$, and is scaled by c_m to reflect a distinct relative weighting between minimizing propellant mass usage and minimizing flight time for each policy while reaching the final orbit. Each of those terms in the reward function is scaled to be approximately on the order of 10^0 throughout the transfer. Finally, Ω is assigned a nonzero value upon activation of one or more of the following episode termination conditions: (1) $\Omega = 100$ when the spacecraft approaches within 384 km in position and 5.1 m/s in velocity, respectively, of the closest state of the final halo orbit; (2) $\Omega = -1000$ when the spacecraft impacts the Moon; and (3) $\Omega = -1000$ when the spacecraft drifts farther than 7,500 km from both the final orbit and reference trajectory, a value that is selected to allow the policies to explore the environment but not so large as to depart the vicinity of the initial and final orbits.

With the selected formulation of the reward function, the variable mass coefficient creates a multi-objective trade space. Specifically, for each state along the initial orbit, four policies are trained with mass coefficients of $c_m = [0, 13.33, 26.66, 40]$, corresponding to increasingly larger penalties for propellant mass usage. These values of

the mass coefficient ensure that the propellant mass usage term does not dominate the reward function, which would produce policies that do not leave the initial orbit. The resulting policies are collectively used to uncover regions of the solution space that balance minimizing flight time and propellant mass usage with distinct relative weightings.

V. Nonlinear Constrained Optimization

Once locally optimal solutions are recovered using MRPPO, the solutions are compared to those generated via classical constrained optimization. In this paper, constrained optimization is performed using a multiple shooting algorithm in combination with sequential quadratic programming (SQP), implemented via *fmincon* in MATLAB© [48]. This approach uses free variable and constraint vectors that describe a trajectory and the conditions it must satisfy. Given that each method is only able to produce locally optimal solutions, the transfers developed by MRPPO serve as the initial guesses used in optimization. The transfer duration, maximum available thrust, and thrust vector formulation, are all defined equivalently for MRPPO and the constrained optimization scheme, but the distinct formulations of each approach prohibit completely equivalent implementations and, therefore, the expectation of equivalent results. For instance, *fmincon* requires the objective function and constraints to be continuous and possess continuous first order derivatives. Accordingly, the objective function for constrained optimization is reformulated to solely maximize an objective function that resembles the reward function used by MRPPO while state and action continuity is enforced via equality constraints rather than sequential propagation. While the recovered solution spaces are expected to be slightly different, this comparison facilitates a high-level validation of the solution space recovered by MRPPO.

A. Free Variables

The free variable vector describes two distinct components of the transfer between the selected halo orbits: (1) the low-thrust transfer segment and (2) a coast segment along the final periodic orbit. The position, velocity and spacecraft mass at the initial state along the first arc are held fixed during optimization based on a specified initial condition. Accordingly, only the action components for this first arc, $\boldsymbol{u}_1$, may vary and, therefore, are included in the free variable vector. Then, the i-th low-thrust arc is described by the vector $\boldsymbol{N}_{LT,i} = \left[x_{i,0}, y_{i,0}, z_{i,0}, \dot{x}_{i,0}, \dot{y}_{i,0}, \dot{z}_{i,0}, m_{s/c,i,0}, u_{x,i}, u_{y,i}, u_{z,i}, T_i\right]$ for $2 \leq i \leq n$, which captures its initial state and mass as well as the direction and magnitude of the thrust vector where T_i is constrained to lie within the range $[0, T_{max}]$. These free variables for a low-thrust arc do not include the propagation time, which is fixed to the duration of a time step Δt, to ensure a direct comparison to the MRPPO formulation and a consistent flight time. For the i-th arc along the final orbit coast segment, the position, velocity, and propagation time are free variables, summarized as $\boldsymbol{N}_{C,i} = \left[x_{i,0}, y_{i,0}, z_{i,0}, \dot{x}_{i,0}, \dot{y}_{i,0}, \dot{z}_{i,0}, \Delta t_i\right]$ for $1 \leq i \leq q$. Then, the entire free variable vector is written as $\boldsymbol{X} = \left[\boldsymbol{u}_1, \boldsymbol{N}_{LT,2}, \ldots, \boldsymbol{N}_{LT,n}, \boldsymbol{N}_{C,1}, \ldots \boldsymbol{N}_{C,q}\right]^T$. This free variable vector completely describes the low-thrust transfer in a manner that is consistent with the problem variables used in MRPPO.

B. Equality Constraints

A constraint vector is formulated to possess a norm of zero when all the constraints are satisfied. Full state continuity is enforced between the state $\boldsymbol{x}_{i,f}$ at the end of the i-th arc and the state $\boldsymbol{x}_{i+1,0}$ at the beginning of the $i+1$-th arc using the vector $\boldsymbol{c}_i = \boldsymbol{x}_{i,f} - \boldsymbol{x}_{i+1,0} = \boldsymbol{0}$ for $i \leq (n+q-1)$. The state at the end of the final arc is also constrained to ensure insertion into the final orbit, i.e., $\boldsymbol{c}_F = \boldsymbol{x}_{n+q,f} - \boldsymbol{x}_{IC,F} = \boldsymbol{0}$. Similarly, a mass continuity constraint between the end of i-th and start of the $i+1$-th low-thrust arc is written as $m_{i,f} - m_{i+1,0} = 0$ for $i \leq n-1$ and the thrust direction unit vector for the i-th low-thrust arc requires $u_{i,x}^2 + u_{i,y}^2 + u_{i,z}^2 - 1 = 0$ for $i \leq n$. These vectors are concatenated to form the complete constraint vector as $\boldsymbol{F}(\boldsymbol{X}) = \left[\boldsymbol{c}_1^T, \ldots \boldsymbol{c}_F^T, m_{1,f} - m_{2,0}, \ldots, m_{n,f} - m_{n+1,0}, u_{1,x}^2 + u_{1,y}^2 + u_{1,z}^2 - 1, \ldots, u_{n,x}^2 + u_{n,y}^2 + u_{n,z}^2 - 1\right]^T$. Solutions satisfy these constraints when $||\boldsymbol{F}(\boldsymbol{X})|| = 0$ to within 1×10^{-10} nondimensional units.

C. Objective Function

The objective function defined in this paper resembles the reward function in Eq. (6) but with a negative sign to reflect minimizing the objective function rather than maximizing the reward function. In addition, the Ω term that reflects activation of termination criteria is arbitrarily replaced by a constant offset of 100, reflecting that the solutions are constrained to always reach the final orbit during optimization. Accordingly, the objective function is written as

$$J = 100 + \sum_{t=1}^{n} \begin{cases} 1 + c_m m_0 (m_{sc,t} - m_{sc,t+\Delta t}) & \kappa_t = -1 \\ 10|\Delta \boldsymbol{d}_{t+\Delta t}| + |\Delta \boldsymbol{v}_{t+\Delta t}| + c_m m_0 (m_{sc,t} - m_{sc,t+\Delta t}) & \kappa_t = 1 \end{cases} \tag{7}$$

where κ_t is determined by the initial guess and held constant during optimization. This approach limits discontinuities in the objective function evaluation between iterations and facilitates a closer comparison between the results generated by the optimization scheme and MRPPO via a fixed transfer time.

VI. Results: Generating Transfers in the Multi-Objective Solution Space

Multiple policies, each with a distinct reward function, are trained to recover unique controls schemes for transfers between two L_2 southern halo orbits in the Earth-Moon CR3BP. Once trained, each policy is evaluated to examine a region of the multi-objective solution space and the balance between propellant mass usage and flight time. Then, the reference trajectories associated with each policy for each initial condition supply initial guesses to the constrained optimization scheme for approximate verification of the results generated via multi-objective reinforcement learning. Finally, the impact of hyperparameter selection on the policies and training is examined.

A. Using MRPPO to Recover Transfers in a Subset of the Solution Space

Four policies are simultaneously trained using MRPPO to determine the trajectories and control histories that guide a low-thrust-enabled SmallSat from initial conditions near one of four departure locations along an L_2 southern halo orbit

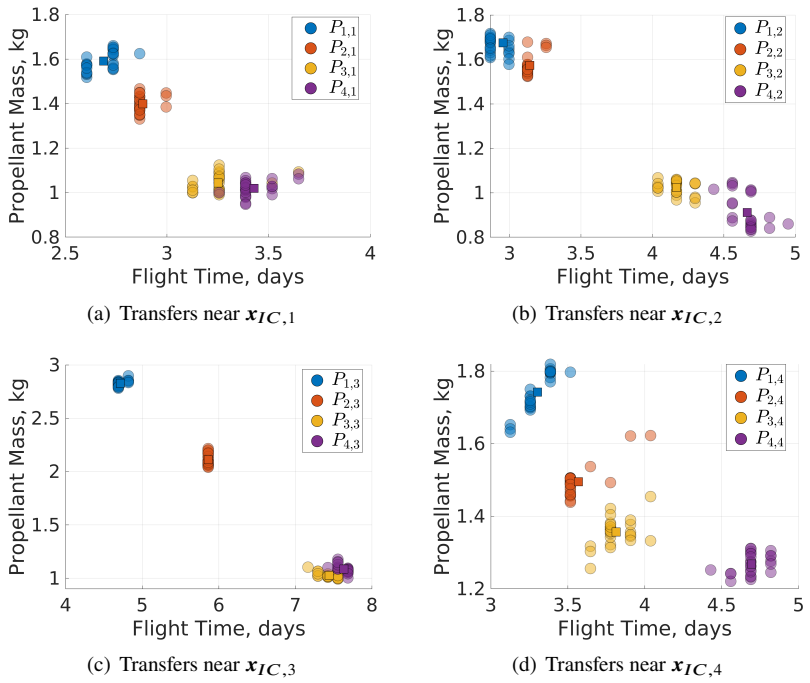
to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP. For the i-th initial reference state, policies $P_{1,i}$ seek to maximize the reward function with no penalty on propellant mass usage, i.e., $c_m = 0$. The proceeding policies, $P_{j,i}$ for $j = [2, 3, 4]$, correspond to larger propellant mass penalties with $c_m = [13.33, 26.66, 40]$, respectively. These policies are trained for 500 updates using a baseline set of hyperparameters and construction parameters, summarized in Table 2, that are manually selected due to their tendency to produce relatively stable performance during training.

Table 2 MRPPO Baseline Hyperparameter and Construction Parameter Values.

Quantity	Value	Quantity	Value
Environmental Steps, τ	4096	Entropy Coefficient, c_2	1×10^{-3}
Epochs, E	5	Maximum Gradient Norm, c_{max}	0.5
Mini Batches, M	4	Actor Hidden Layers, l_a	2
Discount Factor, γ	0.95	Actor Hidden Nodes per Layer, n_a	64
GAE Factor, λ	0.85	Critic Hidden Layers, l_c	2
Actor and Critic Learning Rate, l_r	1×10^{-3}	Critic Hidden Nodes per Layer, n_c	1024
Clipping Parameter, ε	2×10^{-3}	Weight Initialization Scheme	Orthogonal
Value Function Coefficient, c_1	0.5	Activation Function	Tanh

Once the policies are trained and evaluated on a shared set of 25 initial conditions near each reference location along the initial orbit, the mean flight time and propellant mass usage are calculated for the evaluation trajectories for each policy. Figure 3 depicts these characteristics of the trajectories across all four policies, with each subfigure displaying a subset of the recovered multi-objective solution spaces for each departure location. In these figures, the evaluation trajectory characteristics are displayed as semi-transparent circles and the mean values as squares, each shaded by the associated policy. Recall that each evaluation trajectory is uniquely perturbed from the nearby state along the initial periodic orbit; thus, slight differences in the characteristics across the evaluation trajectories are expected. For each initial reference location, the associated solution space indicates, as expected, that policies with a higher value of c_m produce trajectories with a longer flight time and lower required propellant mass. Notably, the trajectories recovered by policies $P_{1,3}$ and $P_{2,3}$ and beginning near $\boldsymbol{x}_{IC,3}$ require a significantly higher propellant mass than near the other initial conditions; the reason for this result becomes clear when analyzing the evaluation trajectories.

The evaluation trajectories generated by each policy for each reference location along the initial orbit reveal differences in geometry, consistent with the distinct basins of convergence for each policy. Figure 4 depicts the reference and evaluation trajectories generated by evaluating all of the policies. The evaluation trajectories are shaded using a variety of colors and the final reference trajectory generated by each policy is depicted as a dotted arc that is completely enveloped by the evaluation trajectories. Transfers originating near $\boldsymbol{x}_{IC,1}$ tend to converge to the final orbit near the maximum value of y; the approach into the final orbit becomes more gradual for policies with a larger c_m. A similar evolution across policies for the transfers beginning near $\boldsymbol{x}_{IC,4}$ is observed: as c_m increases, the transfers reach the vicinity of the final orbit at a location that approaches the maximum value of y. For $\boldsymbol{x}_{IC,2}$, tracking the evolution



(a) Transfers near $\boldsymbol{x}_{IC,1}$

(b) Transfers near $\boldsymbol{x}_{IC,2}$

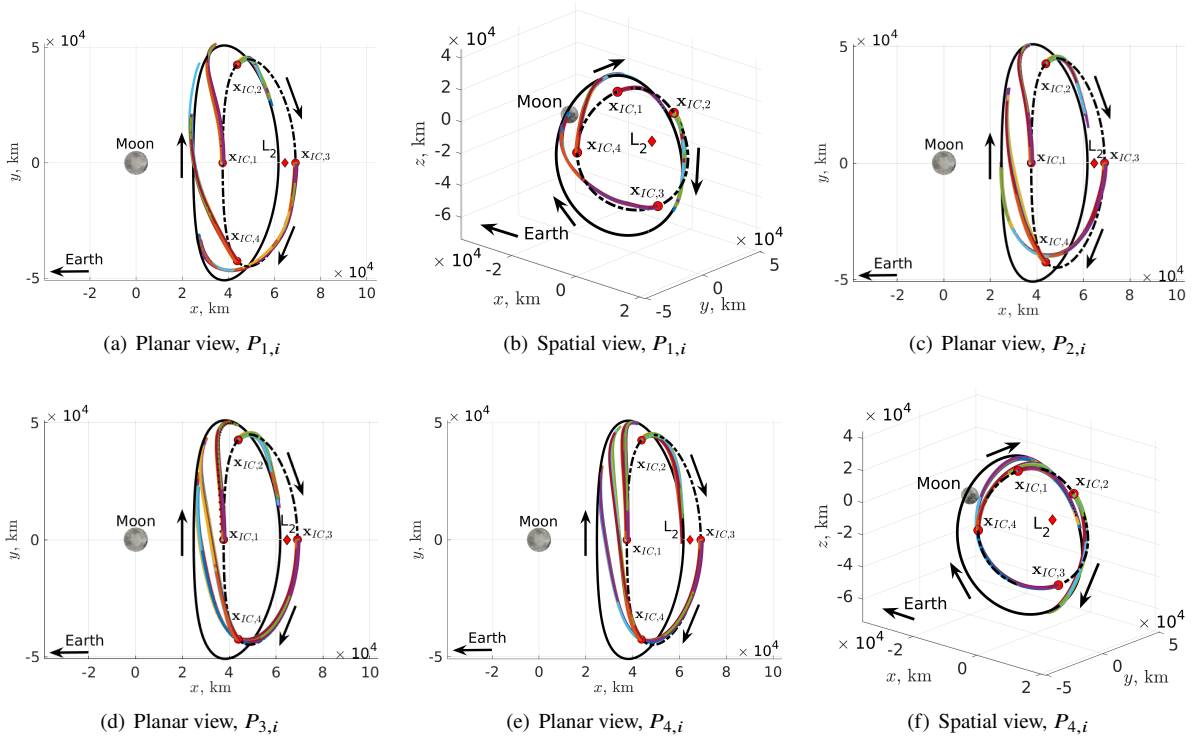
(c) Transfers near $\boldsymbol{x}_{IC,3}$

(d) Transfers near $\boldsymbol{x}_{IC,4}$

Fig. 3 Characteristics of the evaluation trajectories for each scenario for policies trained using MRPPO.

of the evaluation trajectories from $P_{1,2}$ to $P_{4,2}$ reveals the arrival into the vicinity of the final orbit approaching the rightmost x-axis crossing. Finally, for $\boldsymbol{x}_{IC,3}$, policies $P_{1,3}$ and $P_{2,3}$ produce transfers that quickly deviate from the initial orbit and do not geometrically resemble the transfers that originate near any other initial reference locations. However, policies $P_{3,3}$ and $P_{4,3}$ tend to produce transfers that remain close to the initial orbit until reaching the vicinity of $\boldsymbol{x}_{IC,4}$; then, the transfers depart the vicinity of the initial orbit and more closely resemble the transfers for $\boldsymbol{x}_{IC,4}$. This observation suggests that either 1) policies $P_{1,i}$ and $P_{2,i}$ converge towards local maxima that exhibit higher propellant mass requirements than other policies or 2) leaving directly from a state near $\boldsymbol{x}_{IC,4}$ along the initial orbit may offer a significant reduction in the required propellant mass compared to $\boldsymbol{x}_{IC,3}$.

The thrust magnitude along each reference trajectory also reveals distinct differences in one component of the control histories. Figure 5 depicts in each subfigure the thrust magnitude profiles for the reference trajectories leaving from each initial condition for a single policy. In these figures, the thrust magnitudes associated with $P_{1,i}$ possess higher values across the transfers than $P_{4,i}$, consistent with $P_{4,i}$ policies more heavily penalizing propellant mass usage. For example, for transfers near $\boldsymbol{x}_{IC,1}$ across policies with increasing c_m, i.e., from $P_{1,1}$ to $P_{4,1}$, the thrust magnitude is lowest in the middle of the transfer for a longer time interval. In addition, the thrust magnitude significantly decreases at the beginning of the transfer as c_m increases. For $\boldsymbol{x}_{IC,3}$, however, the distinct differences in transfer geometry across policies is also consistent with the differences in thrust magnitude histories. For $P_{1,3}$ and $P_{2,3}$, the thrust magnitude generally possesses values between 50 mN and 225 mN throughout the entire transfer, consistent with the previously



(a) Planar view, $P_{1,i}$ (b) Spatial view, $P_{1,i}$ (c) Planar view, $P_{2,i}$

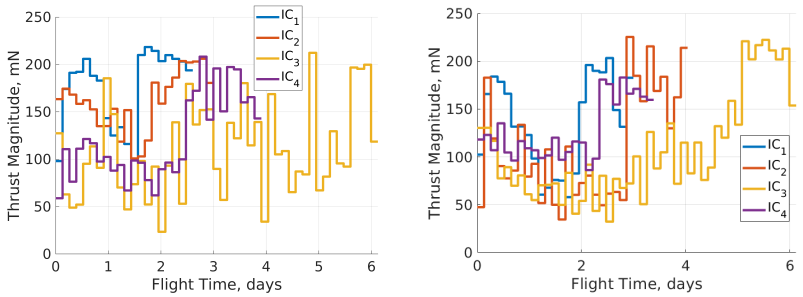
(d) Planar view, $P_{3,i}$ (e) Planar view, $P_{4,i}$ (f) Spatial view, $P_{4,i}$

Fig. 4 Evaluation trajectories and reference trajectories controlled using each policy.

observed rapid departure from the initial orbit and higher propellant mass usage. Policies $P_{3,3}$ and $P_{4,3}$, however, possess low thrust magnitudes, generally below 50 mN, for the first 6 days before rapidly rising to values between 100 mN and 200 mN. This thrust magnitude profile is consistent with the transfers beginning near $\boldsymbol{x}_{IC,4}$ for policies $P_{3,4}$ and $P_{4,4}$ closely following the initial orbit until reaching the vicinity of $\boldsymbol{x}_{IC,3}$ and then departing. Additionally, Fig. 6 compares the thrust direction profiles for policies $P_{1,i}$ and $P_{4,i}$ at initial conditions $\boldsymbol{x}_{IC,2}$ and $\boldsymbol{x}_{IC,4}$. These figures demonstrate that the policies maximizing reward functions with distinct relative weightings between propellant mass usage and flight time tend to produce thrust direction profiles with distinct characteristics.

B. Comparison to Transfers Recovered via Classical Optimization

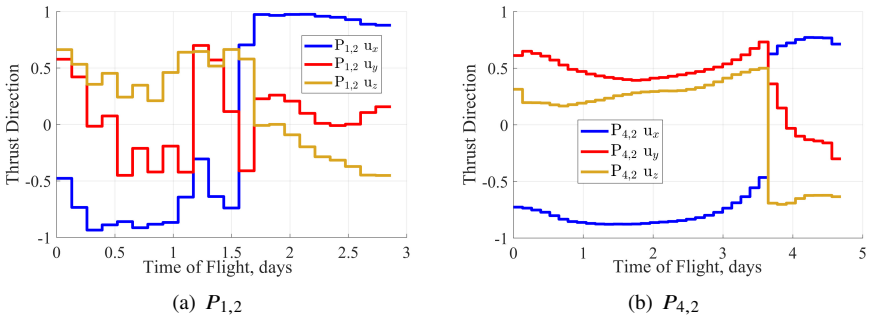
The transfers autonomously generated by MRPPO are used as initial guesses for constrained optimization for comparison to the results produced by MRPPO. In this section, the reference trajectory generated by policy $P_{j,i}$ is used as an initial guess to recover a single locally optimal solution, labeled $O_{j,i}$, by solving the constrained optimization process outlined in Section V and implemented using SQP in *fmincon*. This process is followed for all the reference trajectories from policies $P_{1,i}$ to $P_{4,i}$ for $i = [1, 4]$, producing 16 locally optimal transfers. Figure 7 displays for policies $P_{1,i}$ and $P_{4,i}$ with $i = [1, 4]$ the initial guesses using dashed arcs and the optimal transfers via solid arcs; red arcs correspond to thrust-enabled motion, blue arcs indicate natural motion, whereas black dashed (or solid) arcs denote the



(a) Policies $P_{1,i}$ (b) Policies $P_{2,i}$

(c) Policies $P_{3,i}$ (d) Policies $P_{4,i}$

Fig. 5 Thrust magnitudes for reference trajectories controlled using each policy

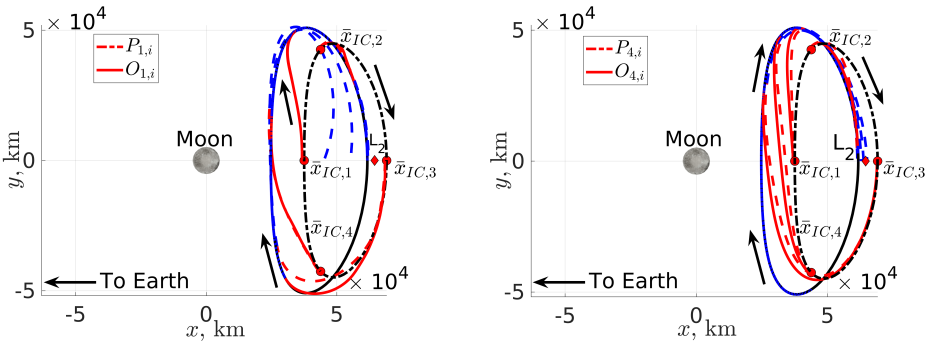


(a) $P_{1,2}$ (b) $P_{4,2}$

(c) $P_{1,4}$ (d) $P_{4,4}$

Fig. 6 Thrust directions for reference trajectories controlled using policies $P_{1,i}$ and $P_{4,i}$.

initial (or final) periodic orbit. Note that the reference trajectories produced by MRPPO target a state along the final orbit to within a specified hypersphere. Thus, naturally propagating the state at the end of a reference trajectory forward



(a) Policies $P_{1,i}$

(b) Policies $P_{4,i}$

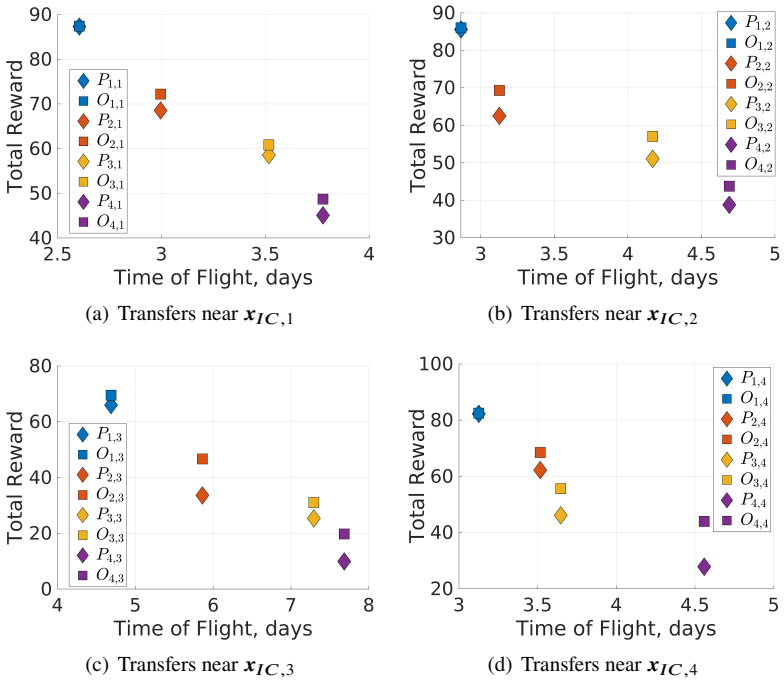
Fig. 7 Initial guesses developed by $P_{1,i}$ and $P_{4,i}$ and optimal transfers recovered via *fmincon* for low-thrust transfers between Earth-Moon L_2 halo orbits.

in time produces one of the blue dashed arcs in Figure 7 that exhibit noticeable deviations from the final orbit. However, these deviations in the initial guess do not appear to significantly impact the recovery of a nearby optimal solution. In fact, in Fig. 7(a), the low-thrust segments of the optimal transfers $O_{1,i}$ for $i = [1, 4]$ remain close to the initial guess and, therefore, the reference trajectories generated by $P_{1,i}$. However, for $P_{4,i}$ in Fig. 7(b), the optimal solutions, $O_{4,i}$ for $i = [1, 4]$, exhibit slight deviations from the initial guess and, therefore, the reference trajectories generated by $P_{4,i}$.

To further examine the differences between the trajectories generated via the policies trained using MRPPO and constrained optimization, Fig. 8 displays the flight time and total reward for the low-thrust transfers associated with each policy at each departure location near the initial periodic orbit. For each initial condition, Policies $P_{1,i}$ recover solutions with close values of the total reward as the optimization scheme. However, as the penalty on propellant mass usage increases, the optimization scheme tends to produce transfers that require less propellant mass usage, which is reflected in the relatively higher value of the total reward. These differences are likely due to distinct implementations: *fmincon* extensively iterates on the control history and transfer from a single initial condition to a state that lies on the final orbit while seeking to limit the propellant mass usage while MRPPO learns from training data to generate transfers from initial conditions that have not been previously seen to within a hypersphere around a state along the final orbit. Nevertheless, these results demonstrate that the policies trained using MRPPO generate solutions with a similar geometry and total reward to those generated with constrained local optimization for the specified set of reward functions. Accordingly, MRPPO enables the recovery of a solution space that is sufficiently similar to a solution space produced by an optimization scheme without requiring a predefined effort intensive initial guess to construct transfers.

C. Hyperparameter Exploration

This section explores the influence of MRPPO hyperparameters on the resulting policies: such an analysis is valuable because poor selections may prevent policies from converging to better locally optimal solutions. Specifically, MRPPO is used to train four policies for only one initial condition, $\boldsymbol{x}_{IC,1}$. These policies are independently trained multiple times



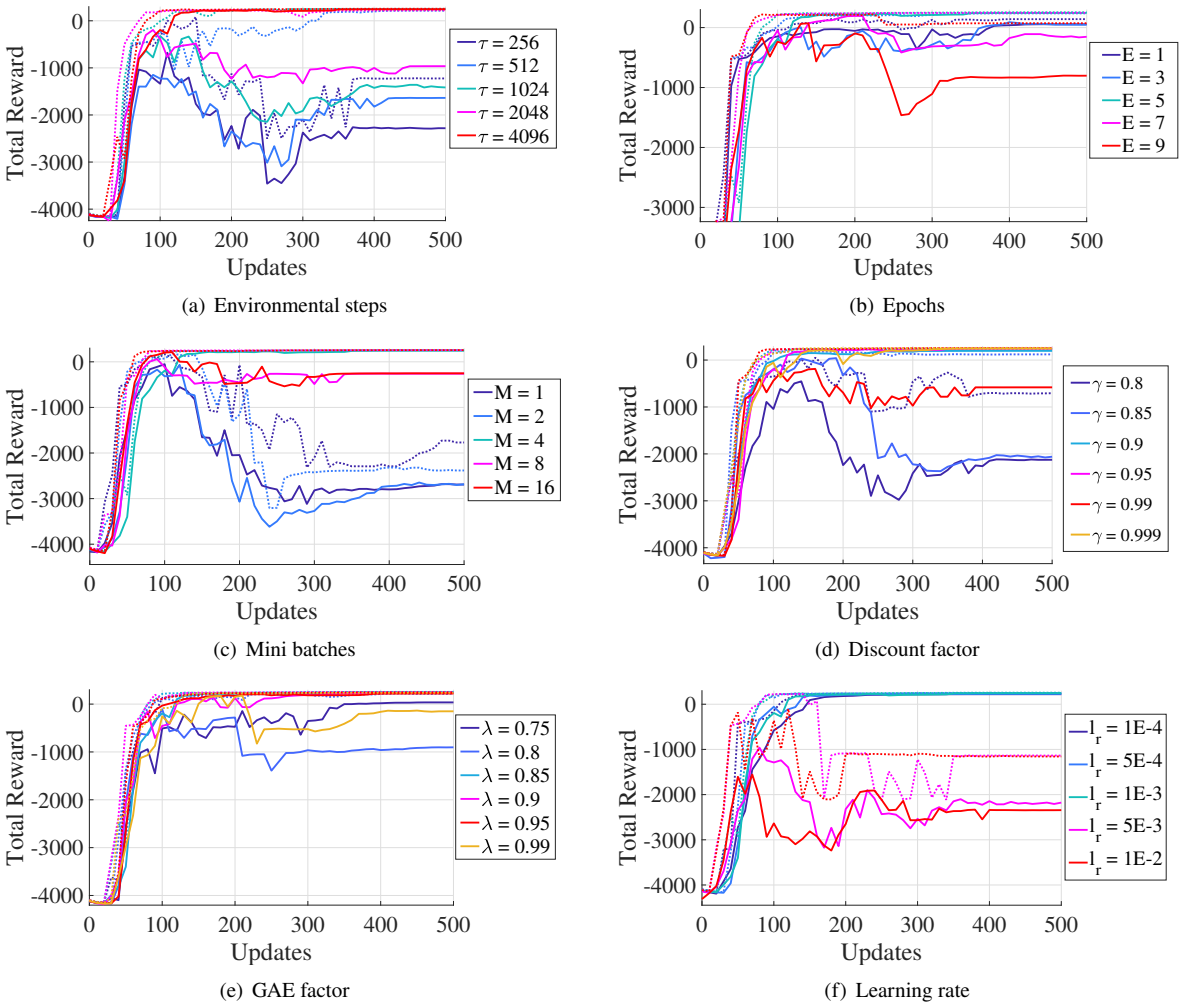
(a) Transfers near $\boldsymbol{x_{IC},1}$ (b) Transfers near $\boldsymbol{x_{IC},2}$

(c) Transfers near $\boldsymbol{x_{IC},3}$ (d) Transfers near $\boldsymbol{x_{IC},4}$

Fig. 8 Solution space for transfers near each initial condition recovered via optimization while maximizing the total reward from initial guesses developed using MRPPO.

while varying only one hyperparameter and setting the remaining hyperparameters to their baseline values, as displayed in Table 2. This process is repeated for several hyperparameters. The trained policies for each set of hyperparameters are evaluated on a common set of perturbed initial conditions for consistency and characterized using the total reward, equal to the sum of the rewards from the evaluation trajectories for all four policies. The total reward is computed for every tenth update to the policies to study the evolution of the total reward throughout training. Additionally, five runs of each set of hyperparameters are performed due to the stochastic nature of DRL. Figures 9 and 10 depict the following quantities calculated for distinct values of 9 hyperparameters at each update used during training: (1) the maximum total reward in any simulation (dashed line) and (2) the mean of the total reward of all five simulations (solid line). Distinct colors correspond to distinct values of each parameter as listed in the legend.

The influence of the hyperparameters that directly govern the updates to the neural networks on the training and final policies is examined. Figure 9(a) reveals that as the number of environmental steps used during training increases, the mean total reward increases. This trend generally demonstrates the benefit of incorporating more information into the updates of the neural networks at the expense of increased training times and memory requirements. The number of epochs and mini batches, however, displayed in Figs. 9(b) and 9(c) respectively, also influences the total reward throughout training. Setting both hyperparameters to small or high values results in slightly worse performance for all policies. This observation may be attributed to both the sensitive environment and the data gathered in a single update being used to update the networks too few or too many times, thereby biasing the training and forcing the



(a) Environmental steps (b) Epochs

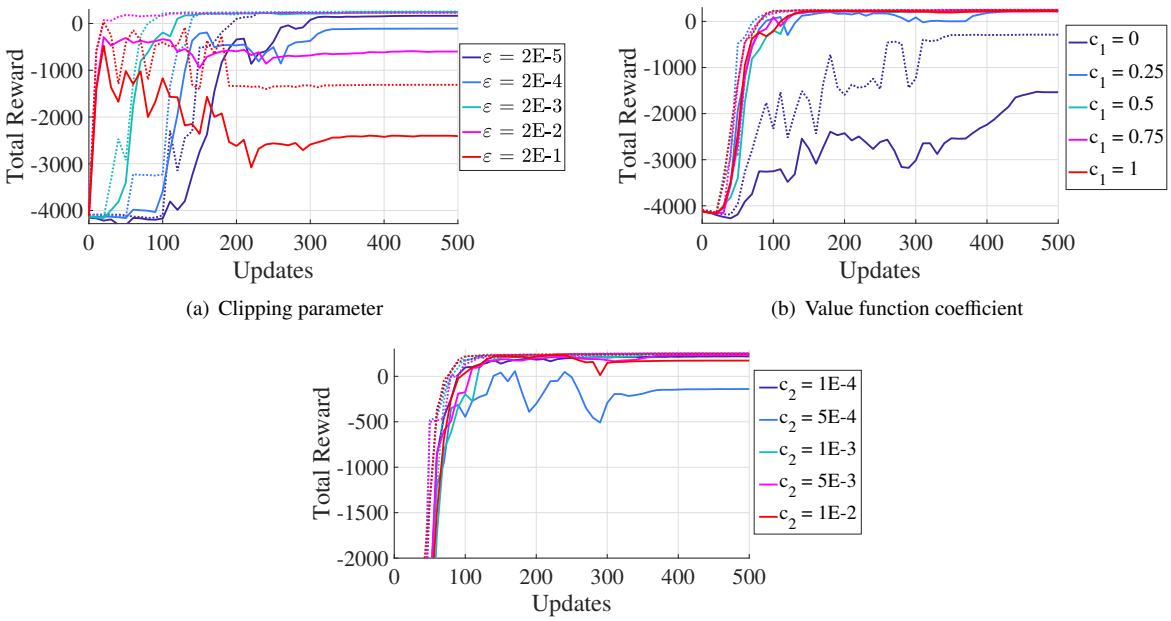
(c) Mini batches (d) Discount factor

(e) GAE factor (f) Learning rate

Fig. 9 Total reward for policies trained using MRPPO for transfer design between Earth-Moon L_2 halo orbits for various values of hyperparameters.

policies to converge to sub-optimal results. Instead, a balanced approach enables data to be used to update the networks multiple times, but not too few or too many so as to produce substandard policies. In Figure 9(d), setting $\gamma \geq 0.9$ enables future rewards to be considered without under-estimating their influence. However, Fig. 9(e) reveals that a balanced approach to selecting the GAE factor as $0.85 \leq \lambda \leq 0.95$ produces high mean total rewards across multiple runs. Finally, Figure 9(f) reveals that a higher learning rate produces policies that significantly diverge while lower learning rates yield more consistent total rewards across policies. These six hyperparameter explorations demonstrate that properly selecting the values that govern the updates to the neural networks is a necessity for training policies with efficient and stable convergence properties and for generating a high, final total reward.

The influence of the hyperparameters used within the objective function defined in Eq. (5) on the resulting policies is also examined. First, Fig. 10(a) reveals that higher values for the clipping parameter produce a faster initial convergence but significant potential for divergence whereas low values result in a slower and smoother convergence of the policies



(a) Clipping parameter

(b) Value function coefficient

(c) Entropy coefficient

Fig. 10 Total reward for policies trained using MRPPO for transfer design between Earth-Moon L_2 halo orbits for various values of hyperparameters.

due to the update size being limited. In this scenario, setting $\varepsilon = 2 \times 10^{-3}$ provides efficient and smooth convergence while also leading to the highest mean total reward. Then, Fig. 10(b) displays the influence of the value function coefficient with $c_1 \geq 0.5$ producing the high maximum and mean total rewards. Similarly, for the entropy coefficient, Fig. 10(c) illustrates that the policies converge to high total reward with no significant difference in performance except when $c_2 = 5 \times 10^{-4}$. Each of these hyperparameters directly affects the evaluation of the objective function during training leading to distinct behavior when evaluated on the same perturbed initial conditions.

VII. Conclusion

An MODRL algorithm is designed using PPO as a foundation to enable multiple policies to be trained simultaneously in a shared environment with a multi-objective solution space; this algorithm is labeled MRPPO. By sharing propagation data, each policy may learn from the potentially beneficial or disadvantageous actions undertaken by other policies, thereby introducing additional exploration into the training. Furthermore, the structure of MRPPO reduces the computational resources and time required to train policies compared to sequentially training them using PPO.

In this paper, MRPPO is applied to a multi-objective trajectory design scenario in the low-thrust-enabled Earth-Moon CR3BP. MRPPO is used to train four policies that produce control histories and transfers for a SmallSat from the vicinity of a reference state along an L_2 southern halo orbit to a nearby higher-energy L_2 southern halo orbit within a single orbital period. These policies also maximize the long-term reward for reward functions with distinct relative

weightings between two competing objectives: minimizing propellant mass usage and flight time. The transfers generated by evaluating each of these policies for a variety of initial conditions near the reference state along initial orbit are characterized to examine a subset of the multi-objective solution space. This process is repeated for a total of four reference states along the initial orbit. Across these policies and reference departure locations, the trained policies successfully produce low-thrust transfers with varying geometries, propellant mass requirements, and flight times to target the vicinity of a state along the final periodic orbit. These transfers, when used as an initial guess for constrained optimization, also possess similar geometries and total rewards to nearby locally optimal solutions. Finally, a preliminary investigation into the effect of selected hyperparameters on the training process and resulting policies is performed. The successful application in this paper to transfer design in a multi-body system supports further exploration of MODRL as a potential approach to autonomously designing solutions that span a region of a multi-objective solution space.

VIII. Acknowledgment

This work was supported by a NASA Space Technology Research Fellowship. Part of this research was performed at the University of Colorado Boulder. Part of the research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). The High Performance Computing resources used in this investigation were provided by funding from the JPL Office of the Chief Information Officer. The authors thank Dr. Alinda Mashiku for serving as the fellowship mentor. The work in this paper also appears in the first author's Ph.D. dissertation titled "Low-Thrust Trajectory Design in Multi-Body Systems via Multi-Objective Reinforcement Learning", submitted in April 2022 to the University of Colorado Boulder [1]. The authors also thank the reviewers for their feedback.

References

[1] Sullivan, C. J., "Low-Thrust Trajectory Design in Multi-Body Systems via Multi-Objective Reinforcement Learning," Ph.D. thesis, University of Colorado Boulder, Boulder, CO, 2022.

[2] Brophy, J., "The Dawn Ion Propulsion System," *The Dawn Mission to Minor Planets 4 Vesta and 1 Ceres*, 2011, pp. 251–261.

[3] Kuninaka, H., Nishiyama, K., Funaki, I., Yamada, T., Shimizu, Y., and Kawaguchi, J., "Powered Flight of Electron Cyclotron Resonance Ion Engines on Hayabusa Explorer," *Journal of Propulsion and Power*, Vol. 23, No. 3, 2007, pp. 544–551. https://doi.org/10.2514/1.25434.

[4] Bosanac, N., Cox, A. D., Howell, K. C., and Folta, D. C., "Trajectory Design for a Cislunar Cubesat Leveraging Dynamical Systems Techniques: The Lunar Icecube Mission," *Acta Astronautica*, Vol. 144, 2018, pp. 283–296. https://doi.org/10.1016/j.actaastro.2017.12.025.

[5] Chaplin, V. H., Goebel, D. M., Lewis, R. A., Lockwood Estrin, F., and Randall, P. N., "Accelerator Grid Life Mod-

eling of T6 Ion Thruster for BepiColombo," *Journal of Propulsion and Power*, Vol. 37, No. 3, 2021, pp. 436–449. https://doi.org/10.2514/1.B37938.

[6] Russell, R., "Primer Vector Theory Applied to Global Low-Thrust Trade Studies," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, March-April 2007, pp. 460–472. https://doi.org/10.2514/1.22984.

[7] Coverstone-Carroll, V., Hartmann, J., and Mason, W., "Optimal Multi-Objective Low-Thrust Spacecraft Trajectories," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, No. 2, 2000, pp. 387–402. https://doi.org/10.1016/S0045-7825(99)00393-X.

[8] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207. https://doi.org/10.2514/2.4231.

[9] Sullivan, C. J., Stuart, J., Anderson, R. L., and Bosanac, N., "Designing Low-Thrust Transfers to High-Inclination Science Orbits via Hybrid Optimization," *Journal of Spacecraft and Rockets*, Vol. 58, No. 5, 2021, pp. 1339–1351. https://doi.org/10.2514/1.A34980.

[10] de Sousa-Silva, P., Terra, M., McInnes, C., and Ceriotti, M., "A Heuristic Strategy to Compute Ensembles of Trajectories for 3D Low-Cost Earth-Moon Transfers," *67th International Astronautical Congress, Guadalajara, Mexico, Paper IAC-16-C1.6.12*, 2016.

[11] Shah, V., Beeson, R., and Coverstone, V., "A Method for Optimizing Low-Energy Transfers in the Earth-Moon System using Global Transport and Genetic Algorithms," *AIAA/AAS Astrodynamics Specialist Conference, Paper AIAA 2016-5438*, Long Beach, CA, September 2016. https://doi.org/10.2514/6.2016-5438.

[12] Englander, J. A., Vavrina, M., and Ghosh, A. R., "Multi-Objective Hybrid Optimal Control for Multiple-Flyby Low-Thrust Mission Design," *Advances in the Astronautical Sciences, Vol. 155: 25th AAS/AIAA Space Flight Mechanics Meeting*, Univelt, Inc., San Diego, CA, 2015, pp. 1251–1270. Paper AAS 15-227.

[13] Yang, X.-S., "Chapter 6 - Genetic Algorithms," *Nature-Inspired Optimization Algorithms, Second Edition*, edited by X.-S. Yang, Academic Press, 2021, pp. 91–100. https://doi.org/10.1016/B978-0-12-821986-7.00013-5.

[14] Miller, D., and Linares, R., "Low-Thrust Optimal Control via Reinforcement Learning," *Advances in the Astronautical Sciences, Vol. 168: 29th AAS/AIAA Space Flight Mechanics Meeting*, Univelt, Inc., San Diego, CA, 2019, pp. 1817–1834. Paper AAS 19-560.

[15] Bonasera, S., Bosanac, N., Sullivan, C. J., Elliott, I., Ahmed, N., and McMahon, J., "Designing Sun–Earth L2 Halo Orbit Stationkeeping Maneuvers via Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, 2022. Available online, https://doi.org/10.2514/1.G006783.

[16] Bosanac, N., Bonasera, S., Sullivan, C. J., McMahon, J., and Ahmed, N., "Reinforcement Learning for Reconfiguration Maneuver Design in Multi-Body Systems," *AAS/AIAA Astrodynamics Specialist Conference*, Virtual, August 2021. Paper AAS 21-568.

[17] Boone, S., Bonasera, S., McMahon, J. W., Bosanac, N., and Ahmed, N. R., "Incorporating Observation Uncertainty into Reinforcement Learning-Based Spacecraft Guidance Schemes," *AIAA SciTech 2022 Forum*, San Diego, CA, 2021. Paper AIAA 2022-1765, https://doi.org/10.2514/6.2022-1765.

[18] Federici, L., Scorsoglio, A., Zavoli, A., and Furfaro, R., "Autonomous Guidance for Cislunar Orbit Transfers via Reinforcement Learning," *AAS/AIAA Astrodynamics Specialist Conference*, Virtual, August 2021. Paper AAS 21-610.

[19] LaFarge, N. B., Miller, D., Howell, K. C., and Linares, R., "Autonomous Closed-Loop Guidance Using Reinforcement Learning in a Low-Thrust, Multi-Body Dynamical Environment," *Acta Astronautica*, Vol. 186, 2021, pp. 1–23. https://doi.org/10.1016/j.actaastro.2021.05.014.

[20] Zavoli, A., and Federici, L., "Reinforcement Learning for Robust Trajectory Design of Interplanetary Missions," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 8, 2021, pp. 1440–1453. https://doi.org/10.2514/1.G005794.

[21] Das-Stuart, A., Howell, K. C., and Folta, D. C., "Rapid Trajectory Design in Complex Environments Enabled by Reinforcement Learning and Graph Search Strategies," *Acta Astronautica*, Vol. 171, 2020, pp. 172–195. https://doi.org/10.1016/j.actaastro.2019.04.037.

[22] Izzo, D., Sprague, C. I., and Tailor, D. V., "Machine Learning and Evolutionary Techniques in Interplanetary Trajectory Design," *Modeling and Optimization in Space Engineering*, Springer, 2019, pp. 191–210. https://doi.org/10.1007/978-3-030-10501-3_8.

[23] Shirobokov, M., Trofimov, S., and Ovchinnikov, M., "Survey of Machine Learning Techniques in Spacecraft Control Design," *Acta Astronautica*, Vol. 186, 2021, pp. 87–97. https://doi.org/10.1016/j.actaastro.2021.05.018.

[24] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., "Human-Level Control through Deep Reinforcement Learning," *Nature*, Vol. 518, 2015, pp. 529–533. https://doi.org/10.1038/nature14236.

[25] Nguyen, T. T., Nguyen, N. D., Vamplew, P., Nahavandi, S., Dazeley, R., and Lim, C. P., "A Multi-Objective Deep Reinforcement Learning Framework," *Engineering Applications of Artificial Intelligence*, Vol. 96, 2020, pp. 1–12. https://doi.org/10.1016/j.engappai.2020.103915.

[26] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D., "Deep Reinforcement Learning that Matters," *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, LA, 2018, pp. 3207–3214. https://doi.org/10.48550/arxiv.1709.06560.

[27] Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., and Bachem, O., "What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study," *arXiv preprint arXiv:2006.05990*, June 2020. https://doi.org/10.48550/arxiv.2006.05990.

[28] Sullivan, C. J., Bosanac, N., Mashiku, A. K., and Anderson, R. L., "Multi-Objective Reinforcement Learning for Low-Thrust Transfer Design between Libration Point Orbits," *AAS/AIAA Astrodynamics Specialist Conference*, Virtual, August 2021. Paper AAS 21-568.

[29] Sullivan, C. J., Bosanac, N., Anderson, R. L., and Mashiku, A., "Exploring the Low-Thrust Transfer Design Space in an Ephemeris Model via Multi-Objective Reinforcement Learning," *32nd AIAA/AAS Space Flight Mechanics Meeting, AIAA Scitech 2022 Forum*, San Diego, CA, January 2022. Paper AIAA 2022-1887, https://doi.org/10.2514/6.2022-1887.

[30] Szebehely, V., *Theory of Orbits*, 1st ed., Academic Press, New Haven, CT, 1967, pp. 16–22. https://doi.org/10.1016/B978-0-12-395732-0.X5001-6.

[31] Stuart, J. R., Howell, K. C., and Wilson, R., "Automated Design of Propellant-Optimal, End-to-End, Low-Thrust Trajectories for Trojan Asteroid Tours," *Journal of Spacecraft and Rockets*, , No. 5, 2014, pp. 1631–1647. https://doi.org/10.2514/1.A32748.

[32] Davis, D. C., Phillips, S. M., Howell, K. C., Vutukuri, S., and McCarthy, B. P., "Stationkeeping and Transfer Trajectory Design for Spacecraft in Cislunar Space," *Advances of the Astronautical Sciences, Vol. 162: AAS/AIAA Astrodynamics Specialist Conference*, Univelt, Inc., San Diego, CA, 2017, pp. 3483–3504. Paper AAS 17-826.

[33] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 2018.

[34] Konda, V. R., and Tsitsiklis, J. N., "On Actor-Critic Algorithms," *Society for Industrial and Applied Mathematics Journal on Control and Optimization*, Vol. 42, No. 4, 2003, pp. 1143–1166. https://doi.org/10.1137/S0363012901385691.

[35] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, August 2017. https://doi.org/10.48550/arxiv.1707.06347.

[36] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

[37] Nwankpa, C. E., Ijomah, W., Gachagan, A., and Marshall, S., "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," *arXiv preprint arXiv:1811.03378*, November 2018. https://doi.org/10.48550/arxiv.1811.03378.

[38] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980v9*, December 2014. https://doi.org/10.48550/arxiv.1412.6980.

[39] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *arXiv preprint arXiv:1506.02438*, June 2015. https://doi.org/10.48550/arxiv.1506.02438.

[40] Zhang, Z., Luo, X., Liu, T., Xie, S., Wang, J., Wang, W., Li, Y., and Peng, Y., "Proximal Policy Optimization with Mixed Distributed Training," *IEEE 31st International Conference on Tools with Artificial Intelligence*, Portland, OR, 2019, pp. 1452–1456. https://doi.org/10.1109/ICTAI.2019.00206.

[41] Holubar, M. S., and Wiering, M. A., "Continuous-Action Reinforcement Learning for Playing Racing Games: Comparing SPG to PPO," *arXiv preprint arXiv:2001.05270*, 2020. https://doi.org/10.48550/arxiv.2001.05270.

[42] Chen, X., Diao, D., Chen, H., Yao, H., Yang, J., Piao, H., Sun, Z., Jiang, B., and Chang, Y., "The Sufficiency of Off-policyness and Soft Clipping: PPO is insufficient according to an Off-policy Measure," *arXiv preprint arXiv:2205.10047*, 2022. https://doi.org/10.48550/ARXIV.2205.10047.

[43] Bosanac, N., Alibay, F., and Stuart, J. R., "A Low-Thrust Enabled SmallSat Heliophysics Mission to Sun-Earth L5," *IEEE Aerospace Conference*, Big Sky, MT, 2018. https://doi.org/10.1109/AERO.2018.8396375.

[44] Shastry, R., Thomas, R. E., Soulas, G. C., Gonzalez, M. C., Patterson, M. J., and Tolbert, C. M., "NASA's Evolutionary Xenon Thruster-Commercial (NEXT-C)," *Double Asteroid Redirection Test (DART) Mission*, 2021.

[45] The Johns Hopkins University Applied Physics Laboratory, "DART: Impactor Spacecraft," `https://dart.jhuapl.edu/Mission/Impactor-Spacecraft.php` 2022. Online, last accessed October 2022.

[46] Trinh, T., Vu, D., and Kimura, M., "Point-of-Conflict Prediction for Pedestrian Path-Planning," *Proceedings of the 12th International Conference on Computer Modeling and Simulation*, Association for Computing Machinery, New York, NY, 2020, pp. 88–92. https://doi.org/10.1145/3408066.3408079.

[47] Song, Y., Steinweg, M., Kaufmann, E., and Scaramuzza, D., "Autonomous Drone Racing with Deep Reinforcement Learning," *arXiv preprint arXiv:2103.08624*, 2021. https://doi.org/10.48550/arxiv.2103.08624.

[48] MATLAB Optimization Toolbox, "MATLAB Optimization Toolbox Version 8.2," , 2018. The MathWorks, Natick, MA, USA.