TLE-Driven A2C Agent for Terrestrial Coverage Orbital Path Planning

Anantha Narayanan[1], **Pruthwik Mishra**[1], **Bhanu Teja**[2]

[1]Sardar Vallabhbhai National Institute of Technology (SVNIT), Surat
[2]Indian Institute of Science (IISc), Bangalore
u23cs088@coed.svnit.ac.in, pruthwikmishra@aid.svnit.ac.in, bhanubattu@alum.iisc.ac.in

Abstract

The increasing congestion of Low Earth Orbit (LEO) poses persistent challenges to the efficient deployment and safe operation of Earth observation satellites. Mission planners must now consider not only mission-specific requirements, but also the growing risk of collisions with active satellites and space debris. This work presents a reinforcement learning framework using the Advantage Actor-Critic (A2C) algorithm to optimize satellite orbital parameters for precise terrestrial coverage within predefined surface radii. By formulating the problem as a Markov Decision Process (MDP) within a custom OpenAI Gymnasium environment, our method simulates orbital dynamics using classical Keplerian elements. The agent progressively learns to adjust five of the orbital parameters—semi-major axis, eccentricity, inclination, right ascension of ascending node, and the argument of perigee—to achieve targeted terrestrial coverage. Comparative evaluation against Proximal Policy Optimization (PPO) demonstrates A2C's superior performance, achieving $\approx 7.21\%$ higher rewards while converging in 21× fewer timesteps. The A2C agent consistently meets mission objectives in diverse target coordinates while maintaining computational efficiency suitable for real-time mission planning applications. Key contributions include: (1) a TLE-based orbital simulation environment incorporating physics constraints, (2) validation of actor-critic methods' superior performance over trust-region approaches in continuous orbital control, and (3) demonstration of rapid convergence enabling adaptive satellite deployment. This approach demonstrates A2C's potential as a computationally efficient alternative for scalable and intelligent LEO mission planning.

Code — https://github.com/Anantha1605/Orbit-and-Launch-Trajectory-Optimizer

Datasets — https://celestrak.org/NORAD/elements/gp.php?GROUP=active&FORMAT=tle

Introduction

In recent years, space interest has grown multifold, with hundreds of spacecrafts launched to meet surveillance, observatory, connectivity, navigation and other needs. Although this has opened new frontiers, it has created a peculiar problem, in the form of orbital congestion. Mission-planning now has to take into consideration existing orbits, collision courses, and space-debris, thus requiring sophisticated systems that adapt to dynamic requirements and diverse targets. Traditional approaches to orbit design rely on computationally expensive optimization techniques (Song et al. 2018), analytical approximations (Savitri et al. 2017), or heuristic planning methods (Mok et al. 2019) that struggle with evolving environments and real-time constraints. While other optimization methods like Sequential Quadratic Programming (Betts and Huffman 1993) provide optimal solutions for constrained scenarios by incorporating non-linear ordinary differential equations, they become computationally prohibitive when rapid re-planning is needed. Thus, a new approach that optimizes orbital parameters is essential to address these challenges.

Reinforcement learning (RL), in particular, has emerged as a paradigm for solving complex sequential problems, at the forefront of the limitations of traditional methods. The application of deep reinforcement learning to orbital mechanics (Kyuroson et al. 2024) presents unique opportunities to develop intelligent and adaptive satellite control systems. Trust-region methods like PPO have shown promise but require extensive computational resources. This work shows through results that unconstrained actor-critic methods, specifically A2C, can provide a more efficient A learning-based approach to orbital optimization that preserves solution quality.

This work addresses the challenges of orbit planning for target-based earth observation satellites in the Low Earth Orbit(LEO), by formulating it as a Markov Decision Process (MDP) problem, and employs Advantage Actor Critic algorithm, to predict optimal parameters. This approach illustrates the usefulness of modern reinforcement learning algorithms in complex mission planning, emphasizing:

- a novel MDP formulation for satellite orbit optimization
- a custom simulation environment for orbital mechanics using Keplerian elements, and
- demonstrating A2C's effectiveness in learning stable orbital policies.

Problem Formulation

Orbital Elements

Every orbit in space is uniquely defined by its set of classical Keplerian orbital elements. These parameters describe

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

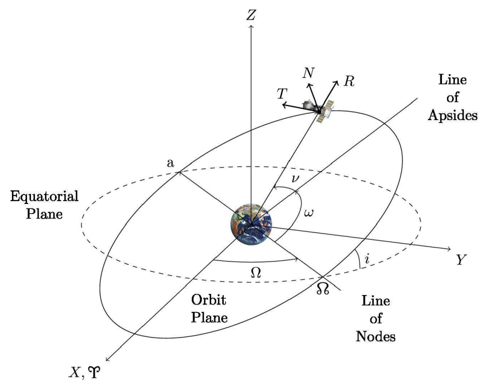

Figure 1: Illustration of the orbital elements used in this study. (Tafanidis et al. 2025)

the size, shape, and orientation of the orbit, as well as the satellite's position along the orbit at a given time.

- **Semi-major axis (a):** Defines the size of the elliptical orbit. It represents half the longest diameter of the orbit and determines the orbital period. For LEO, a typically ranges from approximately 6,700 to 7,500 km, corresponding to altitudes of 300 to 1,200 km above Earth's surface.
- **Eccentricity (e):** Measure of the deviation of a curve from being a perfect circle, typically ranging from 0 to 1. LEO missions often use near-circular orbits ($e \approx$ 0–0.05) to ensure stable coverage and minimize altitude variations.
- **Inclination (i):** It is the angle between the orbital plane and Earth's equatorial plane, typically ranging from 0 to π radians. Inclination dictates the latitudinal coverage, with polar orbits ($i \approx 90°$) enabling global coverage and equatorial orbits ($i \approx 0°$) limiting coverage to low latitudes.
- **Right Ascension of Ascending Node (Ω):** The angle measured in the equatorial plane from the vernal equinox (Υ, the First Point of Aries) to the ascending node, where the satellite crosses the equator from south to north. Ω orients the orbital plane in inertial space and is critical for aligning the orbit with ground targets.
- **Argument of Perigee (ω):** The angle between the ascending node and the orbit's point of closest approach to Earth (perigee), measured within the orbital plane.
- **True Anomaly (υ):** The angle between the ascending node and the orbit's point of closest approach to Earth (perigee), measured within the orbital plane.

For the purpose of orbit optimization, in this work, we focus on the following five elements, which are sufficient to determine the orbital path relative to a central body, such as Earth: semi-major axis, eccentricity, inclination, right ascension of the ascending node, and argument of perigee. The true anomaly (υ) is excluded from optimization, as it represents the satellite's instantaneous position rather than a fixed orbital characteristic.

For LEO missions, these parameters are bounded by both operational constraints (e.g., altitude range) and physical feasibility (e.g., avoiding collisions, satisfying revisit requirements). In the context of this work, the orbital elements are treated as action parameters that can be optimized to maximize the satellite's ability to pass within a predefined threshold to a ground target. Instead of relying on fixed values or manual tuning, we employ a learning-based approach to dynamically adjust during training. This allows the satellite to discover orbital configurations that satisfy coverage objectives while adhering to physical feasibility.

Environment

To facilitate this learning-based orbit optimization, a custom simulation environment was modelled on OpenAI Gymnasium (Towers et al. 2024).

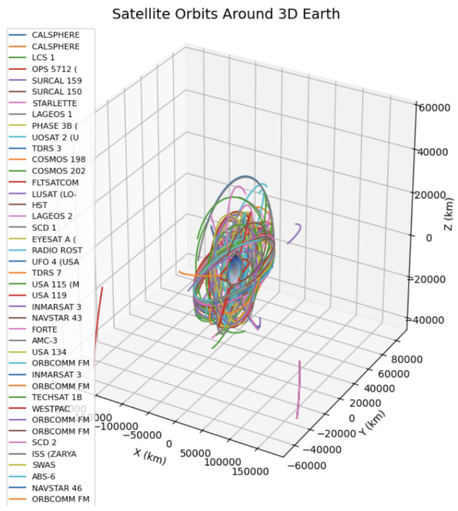

Figure 2: Illustration of the TLE-Based OpenAI Gymnasium environment.

This environment allows for dynamic evaluation and testing using its action and observation spaces, incorporating orbital mechanics. The OpenAI Gymnasium API dictates the functions as standard for compatibility and cross-platform usage with Stable Baselines3.

Initialization and TLE data To ensure realism, scalability, and mission-specific relevance, orbital elements are initialized using parameters constrained by publicly available Two-Line Element (TLE) data of active Earth-orbiting satellites. TLEs are a standardized format used to encode the orbital parameters of satellites, maintained and regularly updated by organizations such as Celestrak and NORAD. This data reflects the actual state of operational satellites and is therefore well-suited for defining plausible orbital configurations in our simulation environment.

Although TLE updates only reflect the instantaneous position of the satellite in its orbital path, this simulation framework focuses on the orbital configuration as a whole rather than the relative position of a satellite in real-time within its orbit. This reduces the need for multiple TLE data retrieval and validation at every simulation step. Instead, the orbital parameters are extracted from the TLE data, which in turn is used to reconstruct the entire orbit. Although static, this method for modelling the environment does not affect real-time capabilities, as the orbits are not based on instantaneous position of the orbital object, but a full representation of the constructed orbit. This approach minimizes the updates overhead, while maintaining real-time mission relevance.

TLE Format and Interpretation A typical TLE consists of two lines of alphanumeric characters that encode various Keplerian elements and satellite identification data. The following is an illustrated example for the TLE data of ISS (ZARYA):

```
ISS (ZARYA)
1 25544U 98067A 24146.63752315 .00009537 00000+0
17465-3 0 9994
2 25544 51.6422 41.9330 0005197 351.2436 8.8447
15.50954063448027
```

The TLE format encodes orbital parameters as follows:

- **Name:** `ISS (ZARYA)`
- **Line 1:**
  - **Satellite catalog number:** 25544
  - **International designator:** 98067A (Launch year: 1998, launch number: 067, piece: A)
  - **Epoch time:** 24146.63752315 (in Julian day format)
  - **First derivative of mean motion:** $\dot{n} = 9.537 \times 10^{-5}$ (orbital decay)
  - **Second derivative of mean motion:** 0.00000 (often set to zero)
  - **BSTAR drag term:** 1.7465×10^{-3} (models atmospheric drag)
  - **Checksum:** 4 (line validation)
- **Line 2:**
  - **Inclination:** $i = 51.6422°$
  - **Right Ascension of the Ascending Node (RAAN):** $\Omega = 41.9330°$
  - **Eccentricity:** $e = 0.0005197$
  - **Argument of Perigee:** $\omega = 351.2436°$
  - **Mean Anomaly:** $M = 8.8447°$
  - **Mean Motion:** $n = 15.50954063$ rev/day
  - **Revolution number at epoch:** 448027

The semi-major axis (a) is derived from the mean motion (n) using the relation

$$a = \sqrt[3]{\frac{\mu}{n^2}}$$

where μ is Earth's gravitational parameter (approximately 398,600 km^3/s^2). For the ISS TLE, $n = 15.50954063\ rev/day$ yields $a \approx 6,792\ km$, corresponding to an altitude of about 414 km above Earth's surface. These TLE-derived bounds inform the allowed range for the six classical Keplerian orbital elements: semi-major axis, eccentricity, inclination, right ascension of the ascending node (RAAN), argument of perigee, and mean anomaly. By constraining the initialization within these ranges, we ensure physically feasible and valid learning states.

Initialization During training, each orbital configuration is randomly initialized within these bounds at the start of an episode or upon policy reset. The typical bounds of the 5 used Keplerian elements for an LEO are:

- *Semi-major axis* (a): 6,700–7,500 km (altitude 300–1,200 km)
- *Eccentricity* (e): 0–0.05 (near-circular orbits)
- *Inclination* (i): 0°–100° (equatorial to near-polar orbits)
- *RAAN* (Ω): 0°–360° (full orbital plane orientations)
- *Argument of Perigee* (ω): 0°–360° (full perigee orientations)

This initialization ensures diversity in training samples, prevents overfitting to a narrow orbital regime, and avoids suboptimal convergence or plateauing in the learning process. Additionally, It enables generalization across a broader spectrum of orbital configurational scenarios, rather than fixed trajectories. This approach enables the learned policy to be more adaptable and ready for practical deployment in real-world mission planning.

Action Space The A2C agent has decisive control over modifying five of the orbital parameters for optimal configuration.

- Semi-major axis (a)
- Eccentricity (e)
- Inclination (i)
- Right Ascension of Ascending Node (Ω)
- Argument of Perigee (ω)

These form a continuous action space, where each element is bounded by physically valid limits specific to Low Earth Orbits (LEO). These parameters are continuously adjusted by the A2C-agent during training and tested in the environment. This continuous nature of the action space allows for exploration of a wider range of orbital configurations to maximize the satellite's proximity to the ground target. The exclusion of true anomaly (υ) from the action space aligns with its role as an instantaneous position parameter rather than a fixed orbital characteristic, as outlined in Section 3.1: Orbital Elements.

Observation Space The Gymnasium environment returns observations to aid both debugging and learning. The observation space provides the reinforcement learning (RL) agent with feedback on the satellite's orbital state and its performance relative to mission objectives.

The observation space is implemented as a `Dict` space that combines `Box` and `Discrete` components, structured as follows:

- **Orbital Elements (Box space):**
  - Semi-major axis (a)
  - Eccentricity (e)
  - Inclination (i)
  - Right Ascension of the Ascending Node (Ω)
  - Argument of Perigee (ω)

- **Ground Target Validity (Discrete):** A binary value (0 or 1) indicating whether the satellite's current orbital trajectory passes within a threshold from the ground target. A value of 1 denotes successful coverage, while 0 indicates failure to comply with this proximity threshold.
- **Coverage Error (Discrete):** A binary value (0 or 1) indicating whether the satellite's orbital altitude satisfies the mission's criteria (i.e., within upper and lower altitude bound constraints). A value of 1 denotes compliance; 0 otherwise.
- **Safety Buffer Distance (Discrete):** A binary value (0 or 1) indicating whether the orbital configuration satisfies the mission's safety threshold. A value of 1 denotes maintenance of a minimum safe distance; 0 otherwise. This is particularly critical for operations in Low Earth Orbit (LEO).

This structured observation format provides a comprehensive state representation, facilitating robust learning and efficient debugging.

Reward Function Design

The reward function forms the structural backbone of Reinforcement Learning, by guiding the learning policy to an optimal state, or in more complex cases like those discussed in this work, a near-optimal state. For ease of understanding, consider a hyperbolic incline in the first two quadrants about the y-axis, and a ball placed at the top. At every instance, the ball rolls down, or mathematically, in the direction of the negative gradient. At a point, the ball stops rolling and settles. Analogously, the reward function moves the RL agents' decisions for orbital parameter estimation towards higher reward generation by minimizing the penalty gradient. The reward structure balances ground target coverage, orbital safety distance, and altitude validity, along with soft constraints on specific parameters like eccentricity and inclination, while penalizing ineffective parametric configurations.

Comparison with Traditional Objective Functions Proven evolutionary algorithmic approaches (Meziane-Tani et al. 2016) justify a multi-objective fitness combining coverage rate, number of satellites, and satellite altitude as separate weighted objectives in a static optimization framework. The proposed reward function for the RL agent mitigates adaptive weight change trade-off limitations by employing non-linear and dynamic weight updates through continuous-valued differentiable reward for policy updates. Each subcomponent is weighted dynamically, allowing for parallel policy updates, the core intent for using the Advantage Actor-Critic (A2C) agents on vector environments. The ground-target validity reward prioritizes coverage maximization, the safety buffer reward enforces collision avoidance, and the parameter-change penalty implicitly promotes fuel-efficient orbital maintenance.

Structure The total reward is computed as a weighted sum of individual sub-rewards and penalties associated with the following:

- **Ground target validity:** Indicative of the orbit's position over the target ground coordinate.
- **Orbital safety distance:** Indicative of the minimum distance to the closest orbit.
- **Coverage altitude:** Indicative of orbital altitude.

Invalid or extreme parametric values are invalidated by the bounds of the observation space. All rewards are normalized and clipped to avoid numerical instability and ensure smooth convergence.

Coverage Altitude Reward The mean altitude of the satellite—in this case derived from the semi-major axis, as LEO orbits are near circular-is expected to remain within the constrained range $[h_{\min}, h_{\max}]$. Deviations are penalized using a normalized coverage error:

$$\text{normalized error}_{\text{coverage}} = \frac{\text{coverage error}}{\max\left(10^{-6},\ h_{\max} - h_{\min}\right)} \tag{1}$$

The coverage reward R_c is then computed as:

$$R_c = \max\left(0.0,\ 1.0 - \text{normalized error}_{\text{coverage}}\right) \tag{2}$$

A penalty P_c is also defined to induce a measure for the degree of violation:

$$P_c = \min\left(1.0,\ \text{normalized error}_{\text{coverage}}\right) \tag{3}$$

Safety Buffer Distance Reward To maintain a minimum safe distance from other operational satellites, a hyperbolic tangent function is used to shape the reward based on proximity margin. The usage of the hyperbolic tangent function normalizes the reward in the range $[-1, 1]$, while providing a gradual slope for increasing reward.

A safe margin value initializes the range for the minimum distance as follows:

$$\text{safe margin} = d_{\min} - d_{\text{safe}} \tag{4}$$

$$\text{normalized margin}_{\text{safety}} = \text{clip}\left(\frac{\text{safe margin}}{d_{\text{safe}}},\ -1,\ 1\right) \tag{5}$$

The safety reward is computed as:

$$R_s = \frac{1}{2} \cdot \left[\tanh\left(\text{normalized margin}_{\text{safety}}\right) + 1\right] \tag{6}$$

The safety penalty is defined as:

$$P_s = 1 - R_s \tag{7}$$

This smooth shaping avoids abrupt reward transitions, promoting stable policy learning.

Ground Target Validity Reward The mission constraints define a ground target coordinate, and a validation threshold σ, the radius within which the orbit must pass over at any time. The agent receives a high reward when the satellite passes within the designated ground target. The reward decays exponentially with increasing distance:

The distance to the target is normalized as:

$$\text{normalized distance} = \frac{d_{\text{target}}}{\sigma} \tag{8}$$

The target validity reward is computed as:

$$R_t = e^{-3\cdot\text{normalized distance}} \tag{9}$$

The target validity penalty is defined as:

$$P_t = 1 - R_t \tag{10}$$

This exponential decay encourages precise coverage and sharply penalizes distant passes.

Individual Element Constraints To improve parametric approximations and accelerate convergence, individual reward shaping is applied to **eccentricity** and **inclination**.

Eccentricity: For Low Earth Orbits (LEO), the eccentricity is lesser than 0.1, resulting in near-circular orbits. Rewarding or penalizing deviations from this value maintains orbital stability and predictability.

Inclination: Inclination determines the latitudinal coverage of an orbit, making it critical for ensuring ground targets reachability at various locations. By shaping the reward based on inclination, the agent is encouraged to select orbits that maximize coverage while remaining within mission constraints.

$$R_{e,i} = R_e + R_i \tag{11}$$

$$P_{e,i} = P_e + P_i \tag{12}$$

This targeted reward strategy improves policy learning by linking the optimization process with key orbital dynamics, leading to faster and more stable convergence during training.

Final Reward and Objective Bonus The final reward is shaped by a weighted sum that can be dynamically altered as per requirements:

$$R = w_c R_c + w_s R_s + w_t R_t + R_{e,i} \tag{13}$$

For evaluative measure of the reward function, and parameter importance, all the rewards are given equal weightage. A soft multiplicative bonus for each objective shapes the reward for faster convergence:

$$\text{bonus} = 3 \cdot \left(\frac{R_s + R_t + R_c}{3}\right)^3 \tag{14}$$

A sharp penalty is awarded based on the objectives not met:

$$\text{penalty} = \left(1 - \frac{R_c + R_s + R_t}{3}\right)^2 \cdot \frac{P_s + P_r + P_t + P_{e,i}}{5} \tag{15}$$

The final reward is computed by adding all defined components:

$$R_{\text{final}} = R + \text{bonus} - \text{penalty} \tag{16}$$

Clipping To prevent explosive rewards and penalties, stabilize training, and prevent outliers, the total reward is clipped to lie in a pre-defined range as:

$$R_{\text{final}} \in [-10,\ 10] \tag{17}$$

Reinforcement Learning Model

To modify and learn orbital configurations, we employ the synchronous `Advantage Actor-Critic (A2C)` algorithm, implemented using the `Stable-Baselines3` library, and configured with a custom callback, and tailored hyperparameters to ensure reward maximization and rapid convergence for the task.

Policy Architecture The policy (actor) and value (critic) networks were both implemented as fully connected feed-forward neural networks, *orthogonally initialized* with the following architecture:

- **Input Layer:** Dimension matches the size of the flattened observation dictionary, including orbital elements and discrete binary flags.
- **Hidden Layers:** Three layers of size 512, 256, 128, respectively. All preceded by `LeakyReLU` Activation function.
- **Output Layer:** The actor outputs the mean and log standard deviation of a multivariate Gaussian distribution.
  - *Action Network:* Outputs the action means for orbital elements.
  - *Critic Network:* Outputs a single scalar value representing the estimated state-value function.

The `LeakyRelu` (Maas et al. 2013) avoids the ‘dying neuron’ problem of `ReLU` (Nair and Hinton 2010) activation, by introducing a small non-zero gradient for negative inputs, thus allowing for continuous weight updates in the network. And this consistent change mitigates the risk of vanishing gradients, a commonly encountered challenge with `Tanh` activation, especially in deep architectures.

A2C Hyperparameter Configuration The hyperparameters chosen and adjusted per requirements for the A2C agent are as shown in Table 1

PPO Hyperparameter Configuration The hyperparameters chosen and adjusted per requirements for the PPO agent are as shown in Table 2

Vectorized Environments For parallel training, the environment is vectorized using `DummyVecEnv` and normalized using `VecNormalize`, improving training stability by reducing scale mismatch across inputs and rewards.

Custom Callback A custom callback enables adaptive intervention during training to address learning plateaus and local optima. This callback extends `BaseCallback` from the Stable Baselines3 framework and introduces dynamic relocation of the agent by force-resetting the training environment when it detects stagnation in performance over successive rollouts.

Hyperparameter	Value
gamma	0.99
gae_lambda	0.98
learning_rate	0.0001
ent_coef	0.03
vf_coef	0.75
max_grad_norm	0.4
use_rms_prop	True
rms_prop_eps	1e-5
n_steps	32
normalize_advantage	True
use_sde	True
sde_sample_freq	75
policy	MultiInputPolicy

Table 1: A2C Hyperparameters used during training

Hyperparameter	Value
gamma	0.99
gae_lambda	0.98
learning_rate	0.0001
ent_coef	0.03
vf_coef	0.75
max_grad_norm	0.4
batch_size	1024 (default)
n_steps	2048 (default)
n_epochs	8
normalize_advantage	True
use_sde	True
sde_sample_freq	75
target_kl	0.3
policy	MultiInputPolicy

Table 2: PPO Hyperparameters used during training

Features of Custom Callback:

- **Plateau Detection:** The callback monitors the agent's average episodic reward after each rollout using a moving window, of size `patience`. If the change in reward across all entries in this window remains below a threshold, the agent is considered to be in a plateaued state.
- **Forced Exploration:** Upon detecting a plateau, the callback forcibly resets the orbital configuration across all vectorized environments. This mandates the agent to explore alternate orbital trajectories and escape a suboptimal local reward maxima.
- **Compatibility with Normalized Environments:** Special care is taken to access and reset environments wrapped under `VecNormalize`, `DummyVecEnv`, or nested vectorization setups.
- **Evaluation-Aware:** The callback periodically evaluates the model's policy using a deterministic policy over a fixed number of episodes (`n_eval_episodes`) in a separate evaluation environment. This ensures that the intervention decision is based on actual policy quality rather than noisy rollouts.

Callback Parameters parameters of the custom callback

Parameter	Value
threshold	0.25
patience	3

Table 3: Callback parameters used for plateau detection and intervention.

Purpose Reinforcement learning in orbital environments can often lead to early convergence to suboptimal policies, due to geometric-constraints and erratic reward gradients. This callback resets the environment when progress stalls, helping the agent explore better strategies. It works alongside entropy, normalization, gradient clipping and state-dependent exploration (SDE) to make training more robust and prevent plateauing, and suboptimal convergence.

Training

Training was carried out using two documented reinforcement learning algorithms of `Stable-Baselines3`: Advantage Actor-Critic(`A2C`) and Proximal Policy Optimization(`PPO`). The PPO agent was trained in 63,448 time steps, While the A2C agent was trained in 3,000 time steps, both with SDE and custom callbacks. Reward shaping via exponential decay and soft penalties enforced early constraint learning. Parallelized dummy vector environments (`DummyVecEnv`) accelerated convergence and enhanced sample efficiency.

Policy Optimization Dynamics

Figures 3 and 4 visualize the training losses associated with policy and value function updates of the PPO agent.

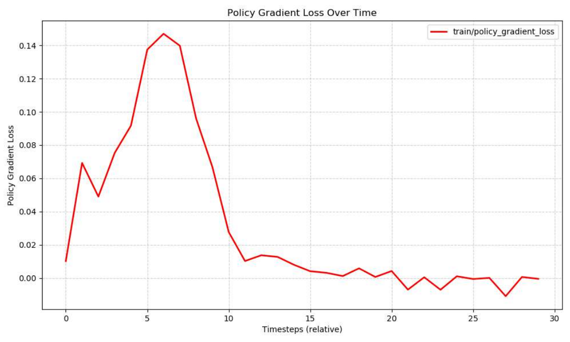

Figure 3: Policy Gradient Loss vs Timesteps (PPO)

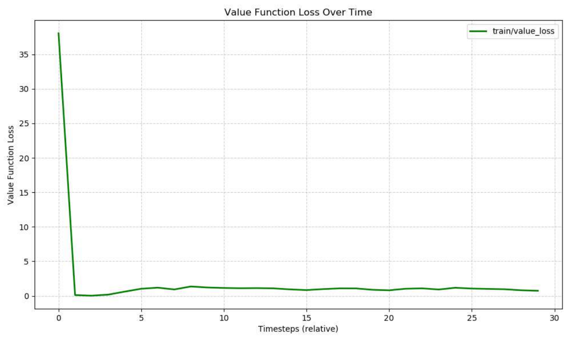

Figure 4: Value Function Loss vs Timesteps (PPO)

General policy optimization methods define the policy gradient loss as:

$$L_\theta^{\mathrm{PG}} = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \cdot \hat{A}_t \right]$$

where π_θ is a stochastic policy, and $\hat{A}_t$ is an estimate of the advantage function at timestep t, defined by:

$$\hat{A}_t \;=\; Discounted\ Rewards \;-\; Baseline\ Estimate$$

PPO objective methods (Schulman et al. 2017b), not too dissimilar from TRPO, defines its clipped surrogate objective as

$$L^{CLIP}(\theta) \;=\; \hat{E}_t \left[min(r_t(\theta)\hat{A}_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$

Where ϵ is a hyperparameter.

TRPO penalizes KL divergence (Joyce 2011) to keep policy updates within the trusted region. Although this approach maintains a stable policy gradient, it can hinder exploration. However, environments constrained by physics, such as those discussed in this work, require sufficient exploration for optimal convergence. Techniques for optimization, as noted by (Schulman et al. 2017a), in their work on Trust Region Policy Optimization, demand a significantly larger number of time steps in training.

PPO mitigates this limitation by employing the clipped surrogate objective and introducing KL divergence as a constraint, enforcing the clipping mechanism. When combined with strategies like forced relocations (refer to subsection custom callback) and more precise hyperparameter tuning—specifically for entropy, gradient normalization, KL convergence, and policy loss—PPO agents can achieve more robust exploration.

In comparison, the employed synchronous (`A2C`), as well as asynchronous actor critic methods (`A3C`) as demonstrated by (Mnih et al. 2016), take a fundamentally different approach that proves more suitable for physics-constrained orbital environments.

Figures 5 and 6 visualize the training losses associated with policy and value function updates of the A2C agent.

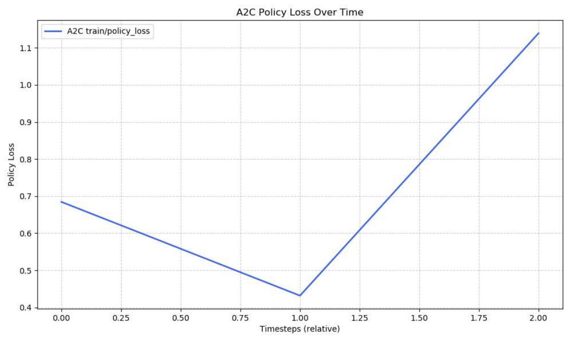

Figure 5: Policy Gradient Loss vs Timesteps (A2C)

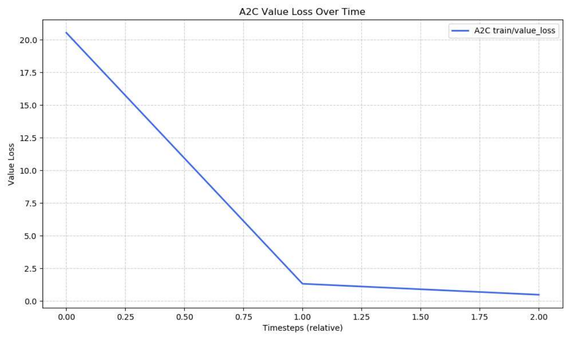

Figure 6: Value Function Loss vs Timesteps (A2C)

A2C utilizes vectorized parallel environments rather than experience replay memory, enabling simultaneous exploration across multiple orbital scenarios. This approach eliminates the memory overhead associated with storing large experience replay data while ensuring that all collected data reflects the current policy, maintaining on-policy learning consistency, advantageous for orbital dynamics where exploration must extend beyond trust regions to discover optimal solutions.

The parallel vectorized environments are more likely to explore different parts of the environment, reducing reliance on explicit entropy regularization hyperparameters. Consequently, the overall experiences collected from the multiple parallel environments (`n_env`) are less likely to be correlated than those from a single environment sequentially generating data. This decorrelation improves gradient estimation, leading to more stable policy updates—a critical factor in orbital mechanics, where small parameter changes can result in significantly different orbits.

Figures 7 and 8 visualize the normalized comparative plots of policy and value losses associated with updates of the PPO and A2C agent.

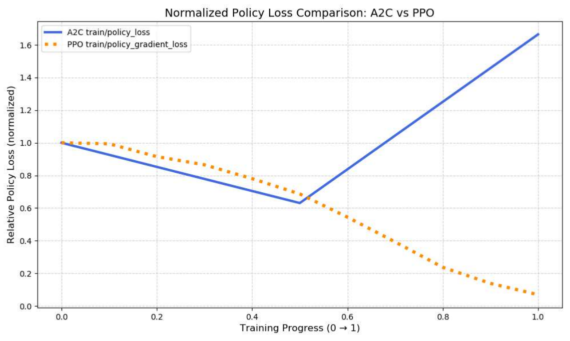

Figure 7: Policy Gradient Loss compared

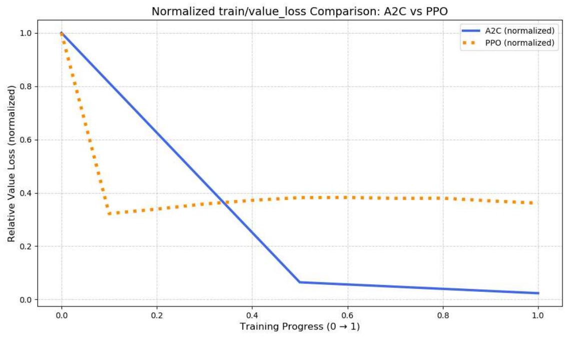

Figure 8: Value Function Loss compared

Prediction & Reward Curve Analysis

A2C's unconstrained policy update generates higher consistent rewards, especially useful in environments demanding aggressive exploration for reward maximization. Unlike the trust-region of PPO, When A2C identifies beneficial orbital configurations that yield high rewards, it immediately takes large policy steps toward these reward-maximizing behaviours.

Figure 9 and 10 illustrate the evolution of the agents' mean episodic reward over time steps.

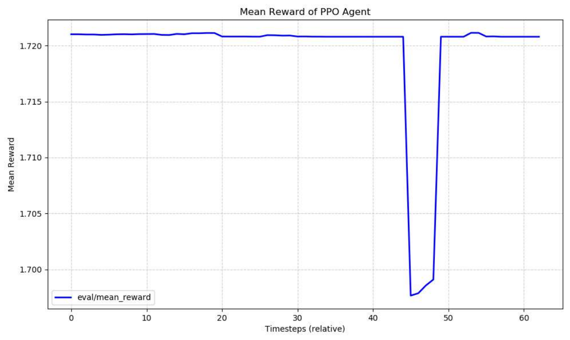

Figure 9: Mean Episodic Reward over time steps for PPO agent

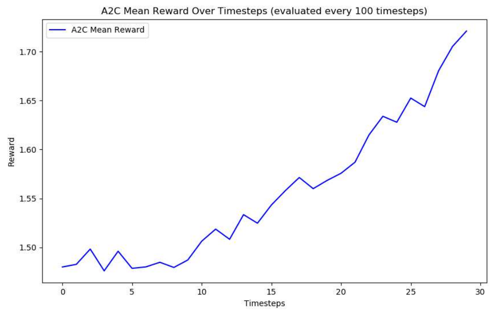

Figure 10: Mean Episodic Reward over time steps for A2C agent

A2C's parallel vector environments enable simultaneous exploration and immediate knowledge propagation, thus increasing probability of high-reward state more frequently than PPO's sequential approach. And its synchronous update mechanism ensures immediate global policy influence in case of updates, creating a tighter feedback loop for policy adjustments. These factors allow A2C to rapidly propagate successful strategies, allowing A2C to identify higher-reward-producing orbital configurations.

Figure 11 presents the rapid exploration and increase of the A2C agent's mean episodic reward over time steps, when compared to the PPO agent.

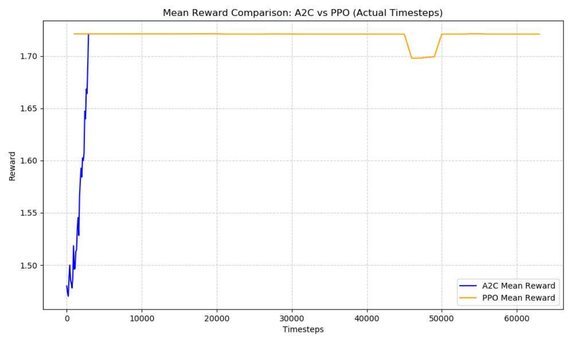

Figure 11: Mean Episodic Reward over time steps for PPO agent

The prediction of the `A2C` model (see table 4) trained to just 3000 time steps yielded better results despite having less training in comparison to the `PPO` model (see table 4) trained to over 63000 time steps. The evaluation reward depends on the objectives met, which, when true, is considered as higher reward. In the plot of mean reward, the reward plot is to explicitly describe the exploration capabilities of each of the agents. And as observed, PPO agent has consistent high mean rewards from the start, but fails to meet objectives because of lack of exploration, or changing orbital parameters.

Table 4: Model prediction and output from trained A2C agent

Parameter	**Value**
Semi-major axis (km)	7535.21
Eccentricity	0.05025487
Inclination (rad)	1.5751818
RAAN (rad)	3.1262968
Argument of periapsis (rad)	3.1353853
Cumulative Reward	10.0
Objectives Met	True

Table 5: Model prediction and output from trained PPO agent

Parameter	**Value**
Semi-major axis (km)	7078.1372
Eccentricity	0.09984795
Inclination (rad)	3.0811646
RAAN (rad)	3.4606233
Argument of periapsis (rad)	6.2831764
Cumulative Reward	9.326987
Objectives Met	True

This performance gap thus highlights the advantage of the A2C algorithm in the given MDP formulation, suggesting actor-critic methods to be well suited for orbit optimization tasks requiring extensive exploration, reward exploitation, and policy stability.

Conclusion

This study successfully developed and validated a TLE-based reinforcement learning framework for autonomous satellite orbit optimization, achieving superior performance through the synchronous Advantage Actor-Critic (`A2C`) algorithm over conventional approaches. Experimental results confirm that `A2C` achieved $\approx 7.21\%$ higher cumulative rewards (10.0 vs 9.326987) while requiring $\approx$ 21x fewer training timesteps (3,000 vs 63,488) compared to Proximal Policy Optimization (`PPO`), showing A2C as an optimal algorithm for physics-constrained orbital planning tasks. The use of a custom reward function, combined with state-dependent exploration and custom callback interventions, allowed the agent to overcome sparse reward challenges and converge toward feasible solutions. The policy network and the tailored hyperparameter successfully guided the orbital elements to maximize target coverage while maintaining safety constraints. Key technical details include: (1) a Gymnasium-compatible simulation environment incorporating real TLE data, orbital mechanics and mission constraints, (2) a hybrid reward function tailored to guide exploration and satisfy objectives, (3) successful handling of sparse reward challenges through state-dependent exploration and custom callback interventions, and (4) validation of actor-critic methods' advantage over trust region approaches in continuous orbital domains. This approach demonstrates the potential of reinforcement learning, in particular, actor-critic methods, as a viable alternative to traditional optimization methods in orbital dynamics, particularly for responsive and adaptive orbit planning in low Earth orbit (LEO) missions.

References

Betts, J. T.; and Huffman, W. P. 1993. Path-constrained trajectory optimization using sparse sequential quadratic programming. *Journal of Guidance, Control, and Dynamics*, 16(1): 59–68.

Joyce, J. M. 2011. *Kullback-Leibler Divergence*, 720–722. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-04898-2.

Kyuroson, A.; Banerjee, A.; Tafanidis, N. A.; Satpute, S.; and Nikolakopoulos, G. 2024. Towards fully autonomous orbit management for low-earth orbit satellites based on neuro-evolutionary algorithms and deep reinforcement learning. *European Journal of Control*, 80: 101052. 2024 European Control Conference Special Issue.

Maas, A. L.; Hannun, A. Y.; Ng, A. Y.; et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, volume 30, 3. Atlanta, GA.

Meziane-Tani, I.; Métris, G.; Lion, G.; Deschamps, A.; Bendimerad, F. T.; and Bekhti, M. 2016. Optimization of small satellite constellation design for continuous mutual regional coverage with multi-objective genetic algorithm. *International Journal of Computational Intelligence Systems*, 9(4): 627–637.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783.

Mok, S.-H.; Jo, S.; Bang, H.; and Leeghim, H. 2019. Heuristic-Based Mission Planning for an Agile Earth Observation Satellite. *International Journal of Aeronautical and Space Sciences*, 20(3): 781–791.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Savitri, T.; Kim, Y.; Jo, S.; and Bang, H. 2017. Satellite Constellation Orbit Design Optimization with Combined Genetic Algorithm and Semianalytical Approach. *International Journal of Aerospace Engineering*, 2017(1): 1235692.

Schulman, J.; Levine, S.; Moritz, P.; Jordan, M. I.; and Abbeel, P. 2017a. Trust Region Policy Optimization. arXiv:1502.05477.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017b. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Song, Z.; Chen, X.; Luo, X.; Wang, M.; and Dai, G. 2018. Multi-objective optimization of agile satellite orbit design. *Advances in Space Research*, 62(11): 3053–3064.

Tafanidis, N. A.; Banerjee, A.; Satpute, S.; and Nikolakopoulos, G. 2025. Reinforcement learning-based station keeping using relative orbital elements. *Advances in Space Research*, 76(2): 750–763.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; Cola, G. D.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; Perez-Vicente, R.; Pierré, A.; Schulhoff, S.; Tai, J. J.; Tan, H.; and Younis, O. G. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv:2407.17032.