Reinforcement Learning and Loss of Plasticity Phenomenon in Coverage Path Planning Environments: An Exploratory Study

João Lucas Cadorniga[1], **Pedro Pertusi**[1], **Fabrício J. Barth**[1]

[1]INSPER. São Paulo, SP - Brazil

{joaolmbc, pedrovmp}@al.insper.edu.br, fabriciojb@insper.edu.br

Abstract. *This paper investigates the phenomenon of loss of plasticity in the context of Reinforcement Learning (RL) applied to Coverage Path Planning (CPP) environments, particularly with the use of curriculum learning. Unlike shortest-path problems, CPP focuses on systematically covering an area. Recent advances in deep RL often face challenges with continual learning, where the network's ability to adapt diminishes with successive new tasks. We developed a reproducible* `Gym` *environment for single-agent CPP with varying difficulty levels and evaluated three RL models: Deep Q-Network (DQN), Proximal Policy Optimization (PPO) without L2 regularization, and PPO with L2 regularization. Our findings indicate that loss of plasticity is present in this CPP task, especially with the DQN model, as agents struggled to generalize and retain knowledge across different map layouts. Curriculum learning, in this context, did not improve the agent's performance and sometimes appeared to hinder learning new levels. While PPO models showed slight improvements in learning later levels compared to DQN, none of the tested models fully mitigated the loss of plasticity.*

1. Introduction

Coverage Path Planning (CPP) refers to the problem of generating a trajectory that allows an agent, such as a cleaning robot, drone, or inspection vehicle, to visit every point in a map region efficiently. Unlike shortest-path problems, CPP requires systematically covering an area without gaps or unnecessary overlap, especially in multi-agent environments [Tan et al. 2021, Galceran and Carreras 2013].

Recent advances in deep reinforcement learning (RL) have achieved great performance across many environments, yet these methods typically face a big challenge when attempting at a continual stream of new tasks that have small differences. An example of this limitation is "loss of plasticity", whereby a deep network's capacity to learn diminishes as it encounters successive learning problems. Dohare [Dohare et al. 2024] demonstrates that standard deep-learning algorithms, both in supervised and reinforcement learning settings, gradually lose their ability to adapt, performing no better than shallow models after extended continual learning.

In their study, a reinforcement learning environment is tested, in which a simulated ant-like robot is tasked with moving forward as rapidly and efficiently as possible. However, it is unclear whether the loss of plasticity phenomenon is also present in coverage path planning (CPP) environments, especially with the use of curriculum learning, a training strategy that involves gradually increasing the difficulty of tasks to improve learning efficiency.

This study aims to investigate the phenomenon of loss of plasticity in the context of CPP, where the agent must learn to navigate and cover different layouts of a map. As a starting point, we will only cover single-agent CPP, where the agent is tasked with covering a single map layout alone.

To accomplish the primary objective outlined above, this paper focuses on the subsequent sub-goals: (i) create and release a reproducible environment for *single-agent* CPP following the `Gym` environment specification [Towers et al. 2024], enabling further research in coverage tasks; (ii) investigate whether loss of plasticity also occurs in curriculum learning in a CPP environment and if curriculum learning mitigates or exacerbates it, and; compare agents trained using RL against a greedy algorithm (A^*) planner as a gold standard, measuring the performance gap and identifying if reinforcement learning may offer complementary advantages.

2. Implementation

The environment implemented in this project is a 2D grid world. Within this grid, the agent can move in four cardinal directions: up, down, left, and right. The environment features three distinct types of tiles: (i) *empty tiles*: the agent can move freely on these tiles; (ii) *wall tiles*: the agent cannot move through these tiles, and; (iii) *goal tile*: the agent receives a reward of +1 when it reaches this tile.

The agent starts at a random position on the grid and must visit all goal tiles, making this a CPP (Coverage Path Planning) problem. The reward for a given agent action follows 3 rules:

1. If the agent visits a tile of the goal area, the reward is $+2$;
2. If the agent visited the entire goal area, the reward is incremented by $+30$;
3. Any other movement yields a reward of -1.

To encourage efficiency, the agent receives a reward of +2 for each unique goal tile visited and incurs a penalty of -1 for each step taken. Those reward rules (i.e., reward function) are applied to any agents (A^* or RL-based). Invalid actions, such as attempting to move off the map or through a wall tile, result in the agent remaining in its current state. The agent operates with complete observability of the environment.

The environment consists of five (5) levels, all with the same grid size but with a changing number of goal tiles and walls. The levels are designed to increase in difficulty, with the first level being the easiest and the last level being the hardest. The levels are presented in Figure 1, the target tiles are represented in green, the walls in black, and the empty tiles in white. Using A^* (with Manhattan distance as the heuristic function) as our gold standard, we ran 20 experiments to find the minimum, maximum, and average rewards for each level, not only proving that they are solvable but also providing reward numbers with which to compare the agents.

The A^* algorithm is a well-known pathfinding algorithm that finds the shortest path from the start node to the goal node. It uses a heuristic function to estimate the cost of reaching the goal from the current node. The A^* algorithm is used as a gold standard algorithm to compare the performance of the reinforcement learning models. The results of the A^* are presented in Table 1.

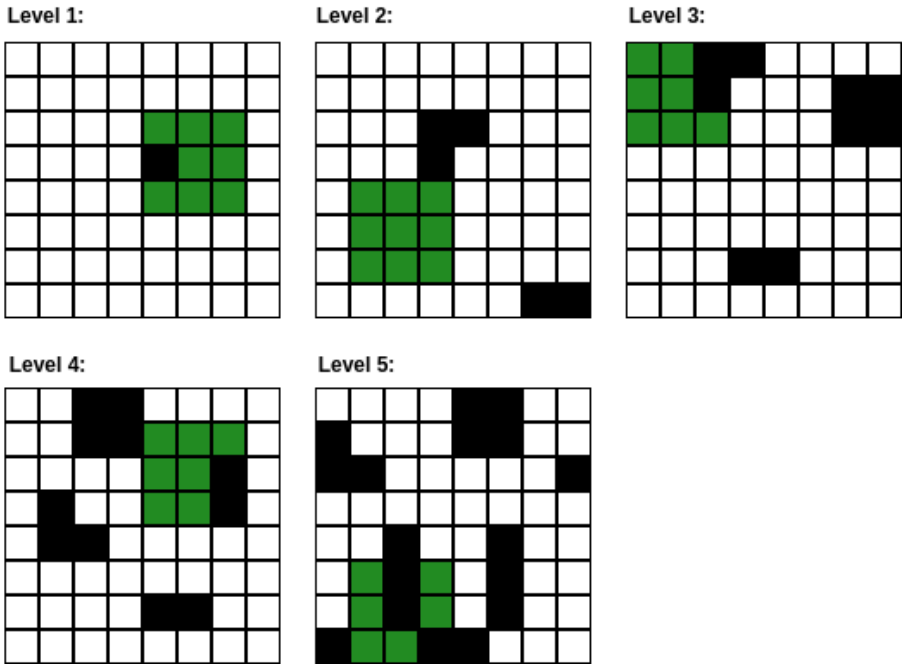

Figure 1. Environment maps

Cost / Level	1	2	3	4	5
Mean	42.25	45.50	40.65	38.25	30.90
Minimum	40	43	37	38	24
Maximum	45	48	44	44	36

Table 1. The cumulative reward of A^* algorithm for each level

In this paper, we evaluated three different algorithms to train agents: the Deep Q-Network (DQN) algorithm [van Hasselt et al. 2015], Proximal Policy Optimization (PPO) algorithm [Schulman et al. 2017] without and with L2 regularization. The PPO algorithm without and with L2 regularization were chosen based on [Dohare et al. 2024], which showed that the use of PPO, and then PPO with L2 regularization helped to mitigate the problem of loss of plasticity.

The DQN algorithm is a reinforcement learning algorithm that uses a neural network to approximate the Q-value function. The Q-value function is used to estimate the expected future rewards for each action taken in a given state. The DQN algorithm uses experience replay and target networks to stabilize training and improve performance. PPO is a policy gradient method that optimizes the policy directly. It uses a clipped objective function to ensure that the policy update does not deviate too much from the previous policy, which helps to stabilize training. The last algorithm to be used was a PPO with L2 Regularization. This is the same as the previous PPO model but with L2 regularization added to the loss function. L2 regularization helps to prevent overfitting by adding a penalty for large weights in the model. According to **(author?)** [Dohare et al. 2024], this approach can also help mitigate the loss of plasticity. This is the reason why we added this approach in the present study.

All models utilized a Multi-Layer Perceptron (MLP) policy. The `MlpPolicy`

is a multi-layer perceptron that uses fully connected layers to process the input state and output the action probabilities. We used the Stable-Baselines3 library [Raffin et al. 2021] to train agents with the DQN and PPO algorithms. The default MLP model from the Stable-Baselines3 library has the following architecture:

- Input Layer: $in = 4 \times 8 \times 8$ (flattened array of the environment observation), $out = 64$
- Two Hidden Layers: $in = 64$, $out = 64$
- Output Layer: $in = 64$, $out = 4$ (corresponding to the environment's discrete action space)

The four channels of the environment observation are all binary, each encoding specific information about the map (channel zero (0) equal to free space (traversable tiles), channel one (1) representing the agent position, channel two (2) the target tiles, and channel three (3) the visited tiles).

3. Method and Results

After successfully creating a `Gym` [Towers et al. 2024] environment and training the agent using the DQN and PPO algorithms, we visualized the results to understand: (i) Is the agent learning at all? (ii) With curriculum learning, is the agent learning faster? Or is loss of plasticity also a problem for curriculum learning? (iii) Is the agent able to solve the problem like the Gold Standard of the A^* algorithm? What is the advantage of using Reinforcement Learning versus Greedy Algorithms?

For each of the three model architectures tested, the following process was used to train and evaluate the agent:

1. Train the agent on the first level for a sufficient amount of timesteps to completely learn the level.
2. Evaluate the agent on the second level to see if it can generalize to the new level.
3. Re-train the agent on the second level, which means to start with the neural network weights from the previous level training, with the same amount of timesteps and without exploration, to see if the previous training was effective or loss of plasticity would affect the learning.
4. Evaluate the agent on all levels to see if any improvements were made to its performance, and if it forgot the previous levels.
5. Repeat steps 2-4 for the third, fourth, and fifth levels.
6. Finally, create a complete graph of the agent's reward over time for each level.

In this way, we were able to analyze the overall performance of each model in this problem.

3.1. DQN (Deep Q-Network)

The graph of the agent's reward over time is shown in Figure 2. The hyperparameters used were based on research of similar problems[1], as it was not the main focus of this study.

The agent was able to learn the first level in 1,000,000 timesteps and achieved a stable reward, regardless of the starting position. However, when evaluated on the second

[1] $buffer\_size = 100\_000$, $learning\_rate = 1e-4$, $batch\_size = 32$, $gamma = 0.99$

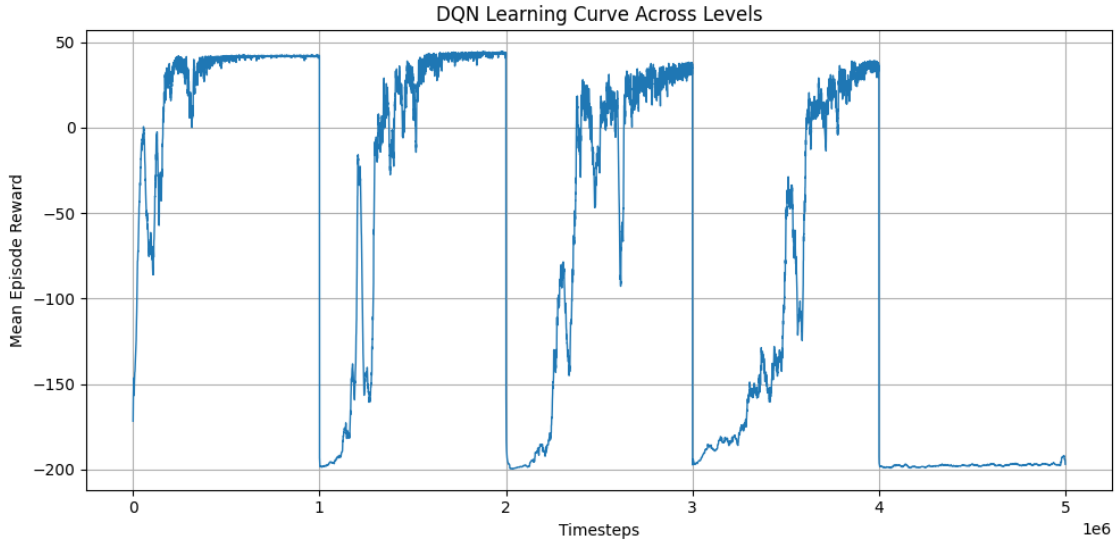

Figure 2. DQN Rewards over Training

level, the agent was unable to generalize and obtained a significantly negative reward, as shown in Table 2. Even with a slightly more challenging level, the agent failed to reuse its previous knowledge. One possible explanation for this behavior is that the model's input consists of the entire grid, rather than a limited view of the agent's surroundings. This may have caused the model to overfit to the full map, hindering its ability to generalize. A possible follow-up study could investigate whether this hypothesis holds true.

Description	**Results**
Mean reward for level 0 (level the model was trained on)	-197.15 ± 4.08
Mean reward for level 1 (level the model was trained on)	-198.80 ± 3.06
Mean reward for level 2 (level the model was trained on)	-199.70 ± 0.90
Mean reward for level 3 (level the model was last trained on)	39.35 ± 2.41
Mean reward for level 4	-199.70 ± 1.31

Table 2. Evaluation of the DQN model trained on level 3 on all levels

The agent was then re-trained on the second level (still for 1,000,000 timesteps) without exploration, and it was able to achieve a stable reward. However, its training was slightly less stable and it took more time to converge, a possible indication of loss of plasticity. After this, it was evaluated on all levels, and the result was not very promising. The agent was able to achieve a stable reward on the second level, but it was unable to generalize to the other levels and completely forgot the first level (Table 2).

Continuing to the next levels, the same behavior was observed. Furthermore, each time the agent took longer to converge and the training was less stable. This is a solid indication of loss of plasticity, as described by [Dohare et al. 2024]. While in their RL experiment, the environment (one of a spider trying to move) did not change, but only the friction of the ground; in our case, the map was changed every level.

Therefore, the DQN model shows strong evidence of the presence of loss of plasticity in curriculum learning, especially in the last level, where the agent was unable to

learn anything at all.

3.2. PPO (Proximal Policy Optimization) w/o L2 Regularization

For the training of the PPO model, the same process was followed as for the DQN model. In this case, 500,000 timesteps were enough for the agent to learn the first level. The graph of the agent's reward over time and an example of its performance on all levels are shown in Figure 3.

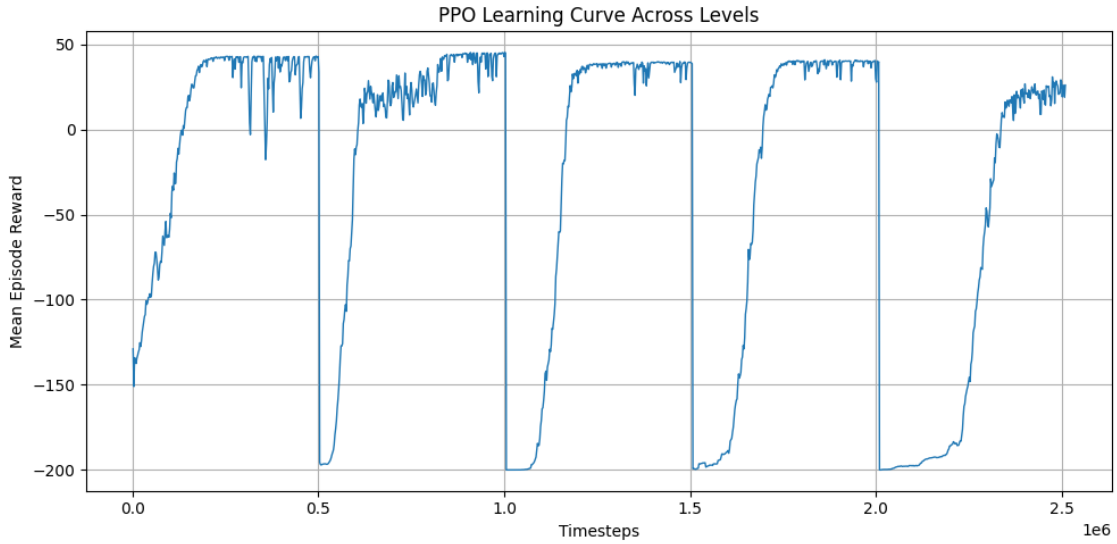

Figure 3. PPO Rewards over Training

The behavior of the agent was very similar to that of the DQN model. The agent was able to learn the first level and achieve a stable reward, but when evaluated on the second level, it was unable to generalize (Table 3). After re-training without exploration on any other level, the agent was able to achieve a stable reward, but forgot the older levels.

Description	Results
Mean reward for level 0 (level the model was trained on)	-194.45 ± 2.38
Mean reward for level 1 (level the model was trained on)	-196.10 ± 6.23
Mean reward for level 2 (level the model was trained on)	-199.55 ± 1.96
Mean reward for level 3 (level the model was last trained on)	40.55 ± 2.20
Mean reward for level 4	-200.00 ± 0.00

Table 3. Evaluation of the PPO model trained on level 3 on all levels

However, a few improvements were noticeable. Different from the DQN model, the PPO agent was able to learn more of the last level, even though it was not able to achieve a stable reward. This could indicate that the PPO model is more robust to loss of plasticity, although it still struggled to learn from certain starting positions.

3.3. PPO with L2 Regularization

Finally, a comparison was conducted between the standard PPO algorithm and a variant incorporating L2 regularization. L2 regularization is a well-established technique for re-

ducing overfitting by penalizing large weight magnitudes in the model. As demonstrated in [Dohare et al. 2024], this approach can also contribute to mitigating the loss of plasticity, motivating its inclusion in the present study. The experimental setup employed the same hyperparameters[2] as the previously tested PPO model, with the only modification being the addition of the L2 regularization term to the loss function.

The same process was followed, and the graph of the agent's reward over time and an example of its performance on all levels are shown in Figure 4 and Table 4.

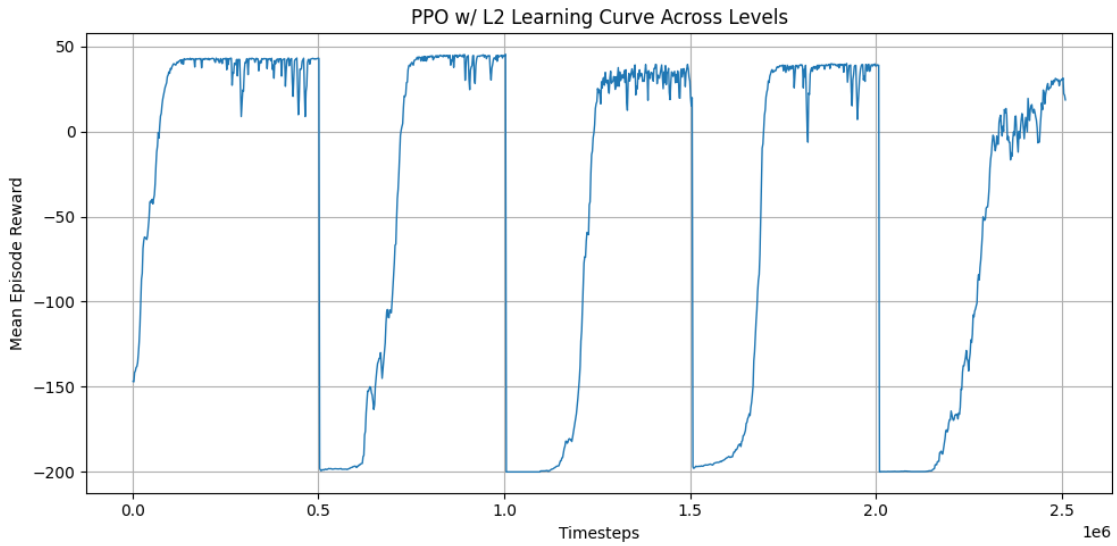

Figure 4. PPO with L2 Regularization Rewards over Training

Description	Results
Mean reward for level 0 (level the model was trained on)	-194.60 ± 2.62
Mean reward for level 1 (level the model was trained on)	-197.90 ± 3.69
Mean reward for level 2 (level the model was trained on)	-148.50 ± 95.68
Mean reward for level 3 (level the model was last trained on)	37.40 ± 3.17
Mean reward for level 4	-200.00 ± 0.00

Table 4. Evaluation of the PPO w L2 model trained on level 3 on all levels

The behavior of the agent was the same as the regular PPO. No improvements were seen in the convergence speed, ability to generalize, or stability of the training. Furthermore, the success level in the last level was extremely similar to the previous PPO model. This could be an indication that the L2 regularization did not help to mitigate the loss of plasticity in this problem.

4. Discussion

The results of the three models show that loss of plasticity is a problem in this environment, especially for the DQN model. To test if the curriculum learning approach was

[2] $learning\_rate = 1e-4$, $n\_steps = 2048$, $batch\_size = 64$, $n\_epochs = 10$, $gamma = 0.99$, $gae\_lambda = 0.95$, $clip\_range = 0.2$, $ent\_coef = 0.01$, $vf\_coef = 0.5$, $max\_grad\_norm = 0.5$

helpful to the learning, we can compare its training for each level to the training of the same model on the same level but without curriculum learning. The model chosen for this comparison was the PPO model with L2 regularization.

As seen in Figure 5, curriculum learning was not capable of improving the performance of the agent. All independently trained levels converged faster, another indication of loss of plasticity, as the previous knowledge of the model was not able to be reused and even seemed to be a hindrance to the learning of the new levels.

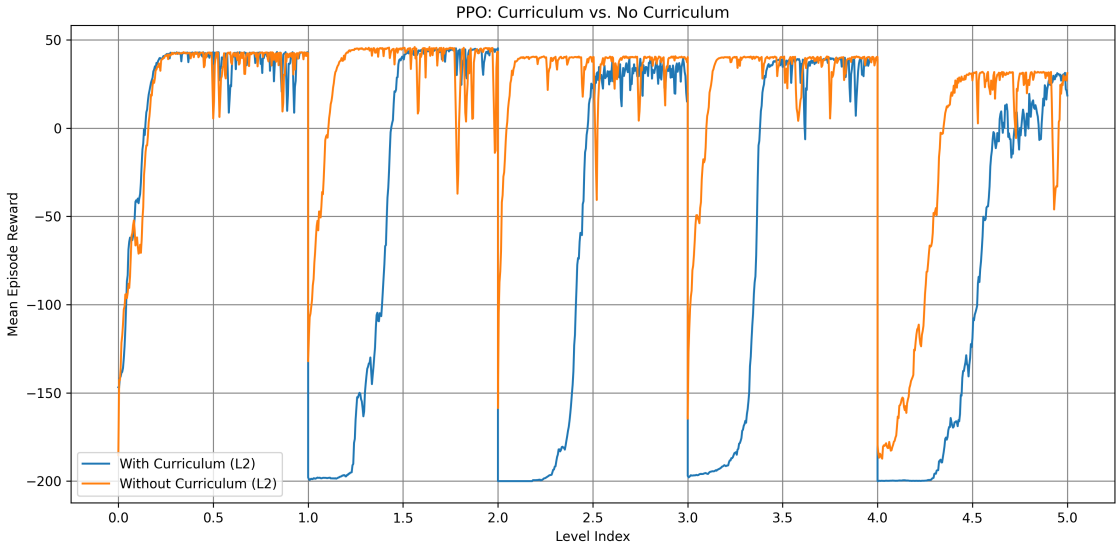

Figure 5. Curriculum Learning vs. No Curriculum Learning Comparison

To be able to compare the three models, we plotted the average reward of each model over all levels (Figure 6). The Figure 6 shows the average reward of each model over all levels.

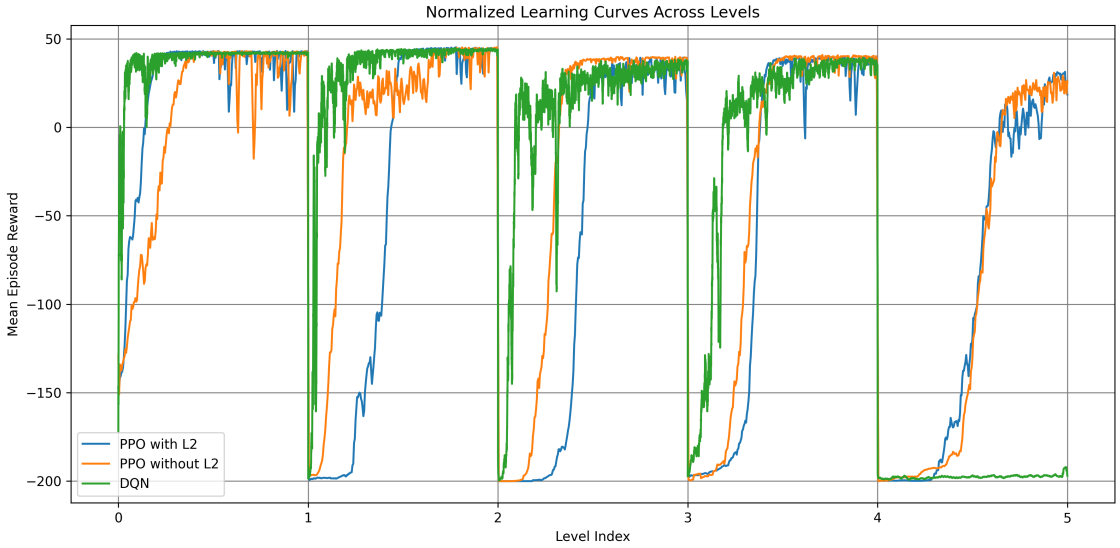

Figure 6. Comparison of Average Rewards Across Models

The PPO model without L2 regularization performed slightly better than the other two, achieving higher average rewards across most of the levels. However, none of the models were able to consistently achieve performance consideraly superior to the A^* baseline, and all showed signs of loss of plasticity, as also seen in Table 5, comparing the average reward of the models trained on each level with A^* over 1000 episodes with different seeds. Furthermore, the higher standard deviation of certain models shows the increasing instability and the lack of capacity to generalize over some initial positions of the agent.

Level	DQN	PPO	PPO L2	A*
0	42.646 ± 1.49	42.723 ± 1.48	42.833 ± 1.61	42.22 ± 1.42
1	44.636 ± 2.39	36.783 ± 44.46	44.835 ± 2.00	45.45 ± 1.84
2	38.286 ± 3.71	29.739 ± 46.56	36.05 ± 28.22	39.92 ± 2.19
3	35.063 ± 30.76	40.316 ± 2.13	34.15 ± 35.19	40.23 ± 2.08
4	-197.549 ± 3.96	25.823 ± 33.18	26.723 ± 31.56	31.18 ± 3.23

Table 5. Mean ± Std Rewards for each Level and Model

5. Conclusion

This study investigated the phenomenon of loss of plasticity in reinforcement learning agents within a Coverage Path Planning (CPP) environment, particularly when employing a curriculum learning approach. We developed a `Gym` environment with five distinct levels of increasing complexity to simulate single-agent CPP tasks. Through extensive experimentation with DQN, PPO, and L2-regularized PPO models, we observed significant evidence of loss of plasticity.

The DQN model demonstrated a clear inability to generalize to new levels and frequently "forgot" previously learned levels, with training stability decreasing over successive tasks. While the PPO models showed some slight robustness compared to DQN, they still exhibited similar patterns of forgetting and generalization challenges, and L2 regularization did not appear to offer a significant advantage in mitigating this issue in our specific environment. Furthermore, our analysis showed that curriculum learning, contrary to expectations, did not improve agent performance or learning speed; in fact, independently trained agents converged faster, suggesting that prior knowledge in the curriculum setup acted as a hindrance rather than an aid.

The results indicate that reinforcement learning agents, in their current tested configurations, struggle to match the gold standard performance of greedy algorithms, such as A^*, in this CPP setting. This highlights a crucial area for future research. While greedy algorithms are limited by local information, RL agents have the potential to learn more efficient long-term strategies, especially in complex or dynamic environments.

A possible follow-up study could involve analyzing if the full grid input to the model, instead of a limited view, contributes to the generalization issue and loss of plasticity. It would also be interesting to explore how agents would behave in a multi-agent environment or a limited view environment. Given the format of our `gym` environment, it would be easy to make the changes and test the agents in these new environments. Additionally, further investigation into hyperparameter tuning and alternative RL architectures tailored for continual learning in dynamic environments could provide valuable insights.

Environments with limited view of the agent's surroundings could also benefit from RL, as the agent would be able to learn to navigate and cover the area more efficiently, while greedy algorithms would be limited to the information available in the current state, which could lead to suboptimal paths. An example is the use of sensors in cleaning robots, where the agent does not have a complete view at once.

Code and Data Availability

All code and data used in this study are publicly available in an anonymous repository to support reproducibility during the peer review process. The repository can be accessed at: `https://github.com/insper-classroom/projeto-intermediario-p-j-cpp`.

References

[Dohare et al. 2024] Dohare, S., Hernandez-Garcia, J. F., Lan, Q., Rahman, P., Mahmood, A. R., and Sutton, R. S. (2024). Loss of plasticity in deep continual learning. *Nature*, 632:768—774.

[Galceran and Carreras 2013] Galceran, E. and Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276.

[Raffin et al. 2021] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

[Schulman et al. 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, abs/1707.06347.

[Tan et al. 2021] Tan, C. S., Mohd-Mokhtar, R., and Arshad, M. R. (2021). A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*, 9:119310–119342.

[Towers et al. 2024] 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.

[van Hasselt et al. 2015] van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. cite arxiv:1509.06461Comment: AAAI 2016.