Evaluating Reinforcement Learning and Search Algorithms: PPO, A2C, and A in various dynamic Settings

Abdurrahim Fazail
School of Computer Science
University of Nottingham Malaysia
line 5: email address or ORCID

Masud
School of Computer Science
University of Nottingham Malaysia
hcyms2@nottingham.edu.my

***Abstract*— This paper explores the design and implementation of an Intelligent Agent, the Autonomous Warehouse Robot (AWR), leveraging reinforcement learning (RL) to optimize navigation and task execution in dynamic warehouse environments. The study compares three algorithms—Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and A* to evaluate their efficiency in pathfinding, obstacle avoidance, and adaptability to unknown environments. The agent's performance is assessed through episodic rewards, policy stability, and adaptation speed in simulated 2D grids. Results highlight trade-offs between RL flexibility and classical algorithm precision, offering insights into real-world deployment challenges, such as localization and safety.**

Keywords—agent state, PPO, A2C, A*, episode,

I. INTRODUCTION

The chosen Intelligent Agent from the choices provided is AWR (Autonomous Warehouse Robot). The report aims to explain in depth the details of the Intelligent Agent that is designed, along with its features, characteristics, the type of model and other miscellaneous information. The literature review highlights the benefits and reliability of robots that can be utilized in working environments, particularly warehouses with high labour demand. The report aims to bridge the gap between novelty and application since factors such as safety serve as a hurdle for mass production for all operations in labour-intensive industries.

II. LITERATURE REVIEW

The existence of Autonomous Mobile Robots in warehouses is theoretically beneficial with current technology capable of producing them, [1]. Providing them to working warehouses would mitigate repetitive tasks that would otherwise be done by humans and reduce fatigue of employee workers as well. In addition, the most wasted warehouse expenses in most circumstances come from the logistical act of moving materials from one area to another [1]. The factors that are considered when taking into account the control of Autonomous Warehouse Robots (AWR) are: *Localization, Artificial Intelligence, and path planning*. Due to relevancy, more emphasis will be placed on Artificial intelligence and localization.

Localization

Localization in robots involves the mapping of its environment to determine its positioning in any area. The only way for robots with no prior data to navigate an environment to represent it in simpler forms [2]. An example of a common algorithm that aids the robot in perceiving the environment with the use of sensors and also estimates the position of the robot simultaneously is
SLAM; with the Simultaneous Localization and Mapping algorithm, the robot can visualize geometrical structures (representing landmarks, obstacles, etc.) also called a map.

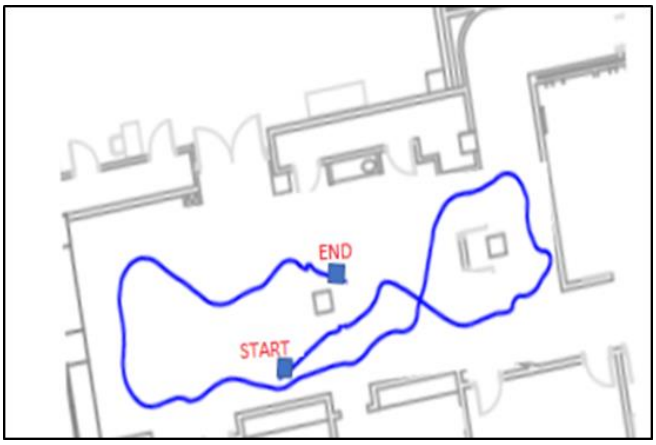


Figure 2. Reinforcement Learning Diagram, [3]li

In addition, another example of a localization method for AWR is hybrid navigation [1]. The method includes a fusion of sensors from devices with short and medium ranges. The medium range device such as a GPS could triangulate the estimated position of the robot in a wide range area and if a precise position is needed, a short-range device could be used such as a magnetic strip. Of course, the AWR represented in the report does not require any Localization algorithms such as SLAM due to it being simulated and is dependent on reinforcement learning for environment mapping; however, in real world scenarios it has to be taken into account for physical autonomous robots.

Artificial Intelligence

In dynamic environments, robots may encounter obstacles that interfere with its mission which the robot needs to manoeuvre around or avoid [1]. Artificial Intelligence algorithms are implemented to ensure the most optimal path is taken despite obstacles or path disruptions. Reinforcement learning (RL) is a machine learning method that learns the most optimal strategies through a series of trial and error [3]. In Reinforcement Learning, the intelligent agent selects an action and receives immediate feedback- reward or punishment from the environment; The objective is to search for the best course of action that minimizes punishment and maximizes long term

reward by continuing to experiment with different strategies. In the case of AWR, hitting an obstacle counts as a punishment action, thus, the robot continues to navigate through the environment with the constant feedback by the environment (warehouse layout).

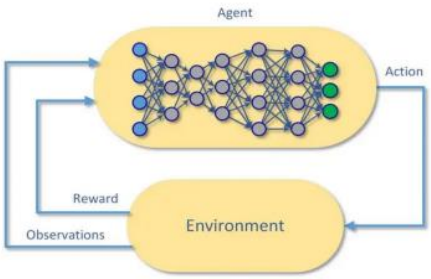


Figure 2. Reinforcement Learning Diagram, [3]

Proximal Policy Optimization is a popular RL algorithm used due to its simplicity and success in various areas of Machine Learning implementation [4]. In regards to AWR, the PPO algorithm can continuously optimize the robot's strategy for selection actions, allowing it to make optimal decisions in unknown or dynamically challenging environments [3]. This mitigates the time taken to learn new environments from scratch as the agent is able to learn quicker with the help of PPO's optimal decision making.

III. Methodology

Task: Train and compare an Intelligent Agent Model to simulate an AWR to pick up and deliver items whilst also avoiding hazardous obstacles within a warehouse environment.

Objectives:

- The Intelligent Agent must find the shortest route to the delivery point in any environment.
- The Intelligent Agent must adapt to hazardous circumstances that occur.
- The Intelligent Agent is able to adapt to new environments and find the next optimal solution as efficiently as possible.

Evaluation Metrics:

- Evaluate individual chosen algorithm metrics.
- Quality of avoidance of hazardous situations (e.g, spills).
- Time taken for the model to adapt to an unknown environment.

The model was trained in Python and the simulated environment was done with the library called Minigrid. The Minigrid library is able to simulate 2D grid environments for use of reinforcement learning or other methods when training Intelligent Agents; for the use of the project, the environment is used to simulate a warehouse layout for the agent to traverse. 10 environments were made for the agent, and each having an exemplary layout of a warehouse.

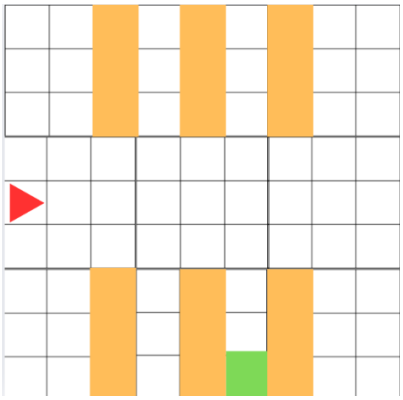


Figure 3. Example of one of the layouts for the agent

After the Looking at Figure 3, the red triangle represents the agent and the green space represents the delivery space. This layout does not include the hazards and the delivery item in place that the agent has to pick up.

Algorithms

The main algorithm utilized for training the model is Policy Proximation Optimization. PPO was chosen due to its popularity and ability to adapt to unknown environments; however, it will be compared to other algorithms for maximum insight in better methods of training for the model. Since PPO is a reinforcement learning method it utilizes a reward function to guide the agent's learning process, with key metrics including: *episodic reward, policy loss, and value function loss*. The episodic reward measures the cumulative rewards obtained per episode, indicating the agent's overall success in the task. Policy loss reflects the stability of policy updates, ensuring the agent does not deviate too drastically from its previous policy; meanwhile value function loss assesses the accuracy of the agent's value predictions, which are vital for reliable policy updates.

PPO Pseudo-Code

for *iteration=1,2, …* **do**
  for *actor=1,2, …, N* **do**
    Run policy $\pi_{\theta_{old}}$ in environment for T time steps
    Compute advantage estimates $\hat{A}_t$ for $t = 1, 2, ..., T$
  end
  Optimize surrogate L wrt. θ.
  $\theta_{old} \leftarrow \theta$
end

Figure 4. PPO Pseudocode [5]

Two other algorithms will be used for comparison, and they are: A* and A2C. A* is a combination of Djikstra's algorithm (guaranteeing shortest paths) and Greedy First search; It leverages on Djikstra's algorithm of absolutely finding the shortest path and Greedy First Search's priority of

'candidates' characteristic together. Thus, resulting in possible optimal and heuristic-driven results.

$$f(n)=g(n)+h(n)$$

f(n) in A*'s core function represents the total estimated cost of the path through n. With g(n) being actual cost and h(n) is the estimated cost from the current square that the agent is on to the goal itself. A2C is another Reinforcement Learning algorithm, which uses policy learning but instead of only raw rewards acting as feedback to the agent, it also utilizes an advantage function; the advantage function measures the quality of an action compared to the average action within the agent's state.

$$A(s,a)=Q(s,a)-V(s)$$

Within the advantage function for A2C, Q(s,a) represents the action score and V(s) as the state score (average reward score the agent would receive from that state); States represent the current situation the agent is in, so for example 2 squares from the goal could be a state. The advantage function encapsulates the comparison of the current reward given to the average reward of the current state, furthering more precision when training the agent. The two differing algorithms were chosen (excluding PPO) as a benchmark of comparison. Since A2C is also a RL algorithm, it prompts a warranted comparison with another RL algorithm: PPO. And A* serves as a 'control variable' of an algorithm as it uses non-Reinforcement Learning methods to train the model, which could reveal the strengths and weaknesses of different method types.

IV. Results

In this section, we present the performance comparison of three reinforcement learning algorithms: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and the A* algorithm, evaluated in the MiniGrid environment.

The training configuration for all algorithms followed a similar setup with the following hyperparameters:

Hyperparameter	Value
Discount factor (γ\gammaγ)	0.99
Learning Rate (lrlrlr):	0.001
GAE Lambda (λ\lambdaλ)	0.95
Entropy Coefficient	0.01
Value Loss Coefficient	0.5
Maximum Gradient Norm	0.5
Recurrence (for PPO and A2C)	4
Adam Epsilon	1e-8
Clip Epsilon	0.2
Epochs	4
Batch size	256

The training was performed on MiniGrid layout 2 which is a balance between a difficult and easy environment using different algorithms, each optimized with the above parameters.

Evaluation Metrics

The evaluation of the training process focused on the following key performance indicators (KPIs):

- **Mean Return:** The average return achieved by the agent per episode during training.
- **Frames per Second (FPS):** A measure of the training speed, indicating how quickly the model processes frames.
- **Training Duration:** The total time taken for training the model.
- **Updates:** The frequency and number of updates performed during training, as this impacts convergence.
- **Entropy:** A measure of the randomness or uncertainty in the agent's action selection. Higher entropy indicates greater exploration, while lower entropy indicates exploitation of the learned policy.

Training Progress and Comparison

PPO (Proximal Policy Optimization)

PPO demonstrated consistent progress, particularly in terms of mean return and policy stability. As shown in the TensorBoard logs, PPO improved gradually with every 2,000 steps, with fluctuations at the start of training that later stabilized as the policy matured.

- **Performance:** PPO reached a mean return of 0.94 after 400,000 frames, demonstrating a sharp and steady learning curve.
- **Losses:** The policy loss and value loss were relatively stable. The value loss decreased significantly over time, contributing to faster convergence than other algorithms.
- **FPS:** PPO achieved a steady FPS of 1,300–2,000, which reflects its relatively fast training speed.
- **Updates:** PPO's updates were infrequent but stable. It performed fewer, larger updates per batch due to the clipping mechanism in PPO, which ensured policy stability and minimized drastic policy changes during training.
- **Entropy:** PPO's entropy was high at the beginning of training, indicating extensive exploration. As training progressed, entropy decreased, signalling the agent was shifting from exploration to exploitation. This decrease in entropy was more pronounced as PPO's learning stabilized, helping it converges to a robust policy.

A2C (Advantage Actor-Critic)

A2C showed steady improvements in performance, although the increases in mean return were more gradual compared to PPO. Its learning curve exhibited more subtle growth, with the model's performance stabilizing after the first 400,000 frames.

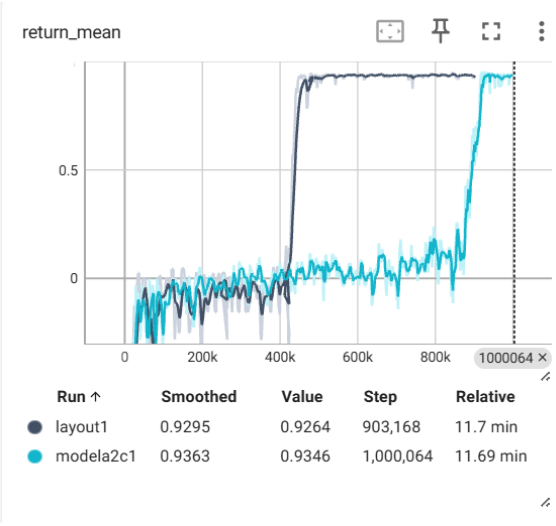
- **Performance:** A2C reached a mean return of around 0.9 after 800,000 frames. This improvement was slower than PPO, especially during the early stages of training, but the agent continued to learn in a more incremental manner.
- **Losses**: Policy and value losses remained higher during early training but began to stabilize after around 40,000 frames, indicating slower convergence compared to PPO.
- **FPS:** A2C displayed an FPS range of 1,300–1,500, which was slightly lower than PPO. This slight reduction in FPS could be attributed to the nature of the advantage function updates.
- **Updates:** A2C performed more frequent updates, potentially because of its less conservative update approach compared to PPO's clipping mechanism. These frequent updates contributed to the agent's incremental improvements but also resulted in more fluctuations early in training.
- **Policy Entropy:** A2C displayed higher entropy values throughout the training process compared to PPO, especially in the earlier phases. The entropy remained moderately high as A2C continued exploring alternative actions, reflecting its more exploratory nature. This may have contributed to the agent's slower convergence but helped avoid overfitting to suboptimal policies.

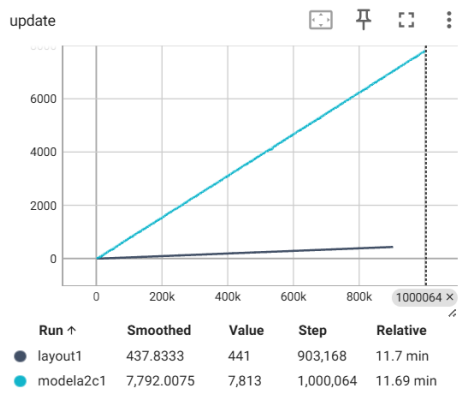
A Algorithm*

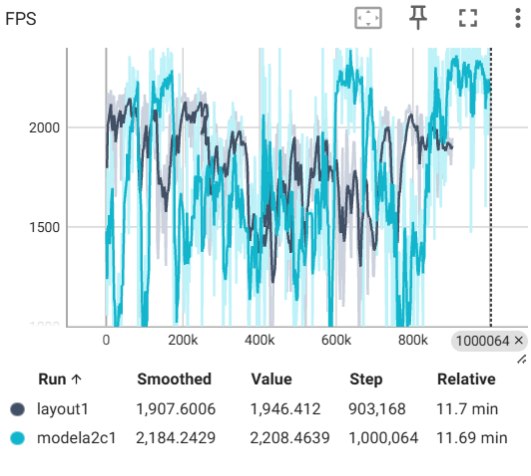
The A* algorithm, typically used in search-based tasks, was applied as a baseline for comparison in terms of action selection rather than continuous policy-based learning. In this setup, A*'s performance could not be directly compared in terms of mean return because it is designed for optimal pathfinding rather than reinforcement learning. However, A* still performed well in navigating to the goal efficiently, with a hundred percent efficiency albeit without the flexibility and learning ability of PPO and A2C.

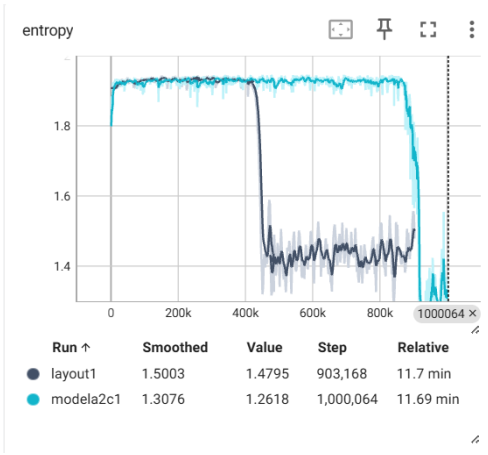
- **Performance:** A* excelled in specific navigation tasks but lacked the learning-based improvement observed in PPO and A2C. Which is hindered with dynamic obstacles such as employees in a warehouse
- **Training Metrics:** As A* was not a policy gradient-based algorithm, metrics like policy loss and value loss were irrelevant in this context.

Dark Blue represents PPO
Light Blue represents A2C









V. Reflections

The training results revealed distinct strengths and weaknesses in the compared algorithms, offering valuable insights into their application for different tasks.

PPO's Efficiency

PPO's ability to stabilize quickly after fluctuations demonstrates its effectiveness in environments requiring policy stability. The steady decrease in entropy suggests a successful shift from exploration to exploitation, making PPO particularly effective for tasks requiring optimal balance in exploration and exploitation.

A2C's Gradual Learning Curve

While A2C's performance was slower to stabilize, its gradual learning curve could be seen as an advantage in environments that require incremental improvements over time. The higher entropy values indicate that A2C is more exploratory, which could help avoid early overfitting to suboptimal solutions, but it comes at the cost of slower convergence.

A in Non-Learning Tasks**

The A* algorithm proved efficient in specific navigation tasks but highlighted a key limitation of search-based algorithms. It lacked the adaptive learning capabilities of PPO and A2C, which are essential when dealing with dynamic environments like those with moving obstacles. This reinforces the importance of learning-based algorithms for tasks that require ongoing adaptation and decision-making. This concludes that perhaps training AWR would be better suited for Reinforcement Learning algorithms.

VI. Future work

For further advancement, this implementation could be applied to real-world physical Intelligent Agents with the added use of physical localization systems (e.g, LiDAR, GPS, etc). It could produce different results compared to the simulated 2D grid due to the extra external factors such as sensor noise, mechanical delays, and environmental unpredictability. Besides that, in a simulated sense, adding more RL algorithms for comparison could also be beneficial in giving more insight into all possible solutions for environment adaptability.

VII. References

[1] R. Keith and H. M. La, "Review of autonomous mobile robots for the warehouse environment," 2024.

[2] A. Singandhupe and H. M. La, "A review of SLAM techniques and security in autonomous driving," 2019.

[3] K. Li, L. Liu, J. Chen, D. Yu, X. Zhou, M. Li, C. Wang, and Z. Li, "Research on reinforcement learning based warehouse robot navigation algorithm in complex warehouse layout," Nov. 2024.

[4] R. Sullivan, S. Huang, A. Kumar, J. P. Dickerson, and J. Suarez, "Reward scale robustness for proximal policy optimization via DreamerV3 tricks," 2023.

[5] B. Faraji, K. Rouhollahi, A. Nezhadi, and Z. Jamalpoor, "Advanced non-linear control based on artificial intelligence tuner for hand tremor suppression," 2022.

Link to github repo: [eemster101/DIA-CW: This repo is for DIA Coursework]