Comparing PPO and A2C Algorithms for Game Levels Generation using Reinforcement Learning

Pakanun Noawaroongroj
National Institute of Development Administration
pakanun.noa@stu.nida.ac.th

Ekarat Rattagan
National Institute of Development Administration
ekarat@as.nida.ac.th

July 10, 2023

Abstract

This study investigates the application of Reinforcement Learning techniques within the PCGRL (Procedural Content Generation via Reinforcement Learning) framework to generate customized game levels. The study employs two distinct algorithms, namely Advantage Actor Critic (A2C) and Proximal Policy Optimization (PPO), and compares their performance across various change percentages and three different representations: Narrow, Turtle, and Wide. The primary objective is to identify the model with the highest success percentage and the most efficient generation time. The findings indicate that the A2C algorithm, specifically with the wide representation model, demonstrates superior performance, achieving an impressive 18% success percentage while requiring a mere 14 seconds to generate 50 game levels.

***K*eywords** Reinforcement Learning (RL) · PCGRL, Procedural Content Generation (PCG) · Level Generation · Advantage Actor Critic Algorithms (A2C) · Proximal Policy Optimization Algorithms (PPO)

1 Introduction

The video game industry has experienced significant growth and transformation over the past few decades, evolving from arcade machines to the widespread availability of games on smartphones. In recent years, the gaming market has shown remarkable expansion, with projections indicating a surpassing of 200 billion USD by the end of 2024, according to Newzoo. Furthermore, Newzoo forecasts an impressive Compound Annual Growth Rate (CAGR) of 8.7%, resulting in a market value of 218.7 billion USD.

In Thailand, the gaming industry has also witnessed a notable rise in recent years, driven by changing consumer behaviors favoring online platforms. The COVID-19 pandemic has further accelerated this trend, with more individuals turning to gaming as a form of entertainment. Recognizing the potential and importance of the gaming sector, the Digital Economy Promotion Agency (Depa) initiated the "Thai Game Industry to Global" scheme in February 2023. This program aims to enhance the skills and game production capabilities of local developers, enabling them to compete on a global scale.

2 Motivation

The motivation behind this study is to develop a model capable of generating game levels that align with specific game conditions. This model should be capable of generating a large quantity of game levels efficiently, serving as a valuable guideline for game developers. To achieve this objective, we employ Reinforcement Learning techniques, specifically focusing on the Proximal Policy Optimization (PPO) and Advantage Actor Critic (A2C) algorithms. By leveraging these algorithms, we aim to explore the potential of generating game levels that meet desired criteria while minimizing the time required for generation.

3 Background

3.1 Reinforcement Learning

Reinforcement Learning is a branch of machine learning that focuses on an agent's interaction with an environment to learn how to make a sequence of decisions or actions in order to maximize a cumulative reward. It draws inspiration from the way humans and animals learn from trial and error through feedback.

In reinforcement learning, the agent learns by taking actions in an environment and receiving feedback in the form of rewards or punishments. The objective is for the agent to learn a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward over time.

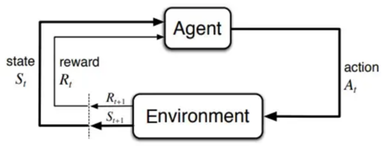


Figure 1: The system architecture of Reinforcement Learning.

The RL framework consists of the following key components:

1. Agent: The learner or decision-maker that interacts with the environment.
2. Environment: The external system or problem domain in which the agent operates. It can be as simple as a game or as complex as a real-world scenario.
3. State: The current representation of the environment at a given time. It provides the necessary information for the agent to make decisions.
4. Action: The choices available to the agent at a particular state.
5. Reward: The feedback signal provided by the environment to the agent after each action. It indicates the desirability of the agent's action and is used to guide the learning process.

3.2 Advantage Actor Critic Algorithms (A2C)

A2C is a popular reinforcement learning algorithm that combines the actor-critic method with advantage estimation. The actor-critic method uses two neural networks - the actor and the critic - to learn a policy and a value function, respectively. The policy (actor) is responsible for selecting actions based on the current state, while the value function (critic) estimates the expected return from a given state.

A2C has several advantages over other reinforcement learning algorithms. It is an on-policy algorithm, meaning it learns directly from the most recent experiences, which can lead to faster convergence. Additionally, A2C utilizes parallelism by running multiple agents simultaneously to collect experiences, which can significantly speed up the learning process.

3.3 Proximal Policy Optimization Algorithms (PPO)

PPO is a state-of-the-art algorithm in the field of reinforcement learning, specifically in the realm of policy optimization. It is designed to find an optimal policy that maximizes the expected cumulative reward in an environment.

PPO is based on the concept of proximal optimization, which aims to update the policy in small steps while ensuring that the new policy does not deviate significantly from the old policy. This helps to maintain stability during the learning process.

PPO has several advantages over other policy optimization algorithms. One of the key benefits is its ability to provide stable and reliable policy updates, thanks to the use of a surrogate objective that limits the policy deviation. This stability helps prevent catastrophic policy collapses or significant policy oscillations during the learning process.

4 Literature review

(Khalifa et al., 2020) have test level generation for three different problems (Binary, Zelda, and Sokoban) with three different representations (Narrow, Turtle, and Wide). The models perform Binary problem well but Zelda and Sokoban, the agent struggled to design hard levels but was able to generate a high amount of playable levels.

(Bhaumik et al., 2020) have compared four tree search algorithms to four optimization algorithms in level generation for three different problems (Binary, Zelda, and Sokoban). They found that GBFS performed very similar to optimization algorithms. While MCTS did not perform well on Binary and Zelda problem but perform very well on the Sokoban problem.

(Shu et al., 2021) have interweaving the EDPCG and the PCGRL frameworks. They test the ED(PRCG)RL framework, EDRL in short, in Super Mario Bros and train RL agents to design endless and playable levels that maximize notions of fun and historical deviation.

(Bontrager and Togelius., 2021) have introduced Generative Playing Networks. The method requires no data, nor domain knowledge, but is computationally intensive. The process is fully differentiable, allowing the agent to directly communicate with the generator about what designs it wants.

No.	Title	Authors	Algorithm	Framework	Finding
1	PCGRL: Procedural Content Generation via Reinforcement Learning	Ahmed Khalifa, Philip Bontrager, Sam Earle, Julian Togelius	PPO	PCGRL	The models perform Binary problem well but Zelda and Sokoban, the agent struggled to design hard levels but was able to generate a high amount of playable levels.
2	Tree Search vs Optimization Approaches for Map Generation	Debosmita Bhaumik, Ahmed Khalifa, Michael Cerny Green, Julian Togelius	GBFS, MCTS	PCGRL	GBFS performed very similar to optimization algorithms. While MCTS did not perform well on Binary and Zelda problem, but perform very well on the Sokoban problem.
3	Experience-Driven PCG via Reinforcement Learning: A Super Mario Bros Study	Tianye Shu, Jialin Liu, Georgios N. Yannakakis	PPO	EDRL	The application of EDRL in SMB makes level generation possible while ensuring certain degree of fun and deviation across level segments.
4	Learning to Generate Levels From Nothing	Philip Bontrager, Julian Togelius	Actor-Critic	GPNs	The generator first generates complex, unsolvable, designs. Then it generates simple solvable designs and finally very easy designs to match the agent's skill.

Table 1: The literature review summary.

5 Solution

We use PCGRL framework (Khalifa et al., 2020) which the architecture inside has some similarity to the normal Reinforcement Learning architecture but add Representation module and the Change Percentage.

Representation assumes the pivotal role in effectuating this transformation. Its primary function encompasses problem initialization, state maintenance, and state modification in response to the actions undertaken by the agent. In order to achieve clarity and ease of understanding, they adopt a representation strategy wherein a generated level is denoted as a 2D array consisting of integers. Each element within the array corresponds to a specific location within the level, and the assigned integer value signifies the type of object present at that particular location.

- Narrow: The most straightforward method of capturing and conveying the problem's intricacies. Drawing inspiration from cellular automata, the agent is presented with the current state and a designated location during each step. Subsequently, it possesses the agency to effectuate modifications specifically at that location.

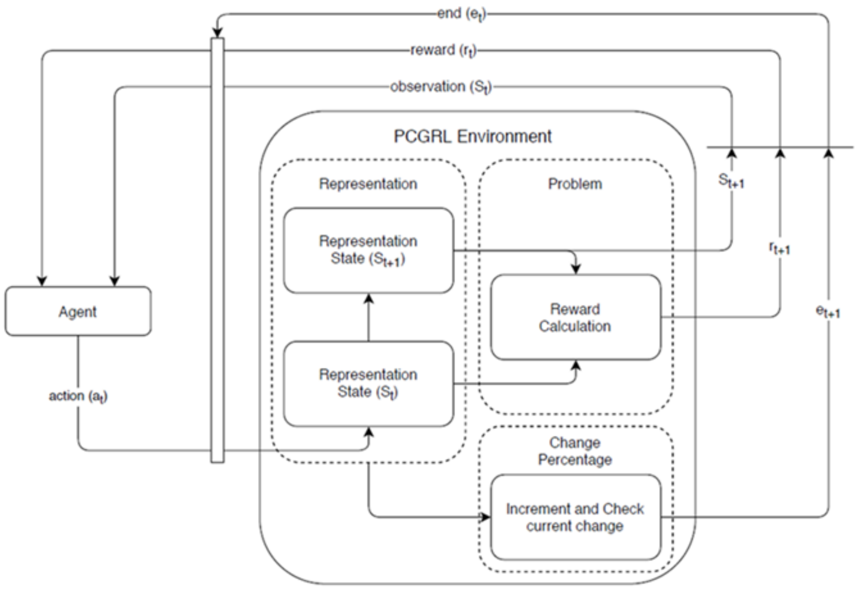


Figure 2: The system architecture for the PCGRL.

- Turtle: The design concept draws inspiration from turtle graphics languages like Logo, which enables the agent to navigate the environment and selectively modify designated tiles during each step of the process.
- Wide: During each sequential iteration, the agent exercises complete autonomy over both the selection of the target location and the subsequent determination of the specific tile type to be assigned.

Change percentage serves as a crucial parameter, as it quantifies the proportion of tiles that the agent is permitted to modify relative to the total size of the level. This parameter plays a pivotal role in constraining the duration of training episodes, thereby preventing the agent from altering all tiles within the level.

6 Experimental setup

6.1 Library

- Python 3.8
- Tensorflow 1.15
- Gym 0.21.0
- Stable-baselines 2.9.0

6.2 Environment

Parameters:

- width: the width of the level that can be change.
- height: the height of the level that can be change.
- probability: the probability of each tile will be presented
- max_enemies: the maximum amount of enemies that can be in the level.
- region: number of connected empty tiles.
- path-length: the longest path across the map

The constructed environment adopts a dungeon theme, comprising various components that contribute to its composition. These components include empty tiles, solid tiles, a player character, diamonds, doors, goblins, bombers, and king pigs.

- width = 11
- height = 7
- probability = “empty”:0.62, “solid”:0.3, “player”:0.01, “diamond”:0.02, “door”:0.02, “globlin”:0.01, “bomber”:0.01, “kingpig”:0.01
- max_enemies = 5
- rewards = “player”:3, “diamond”:3, “door”:3, “enemies”:1, “regions”:5, "path-length": 1

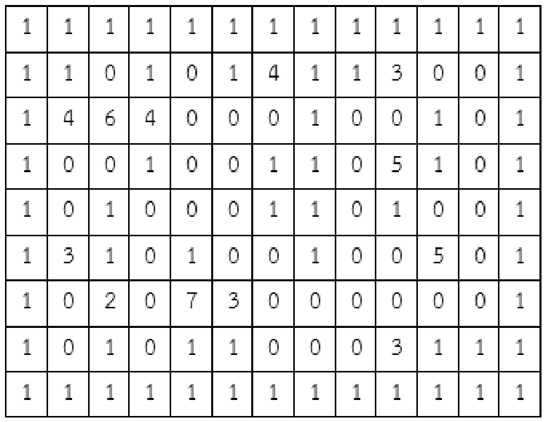
6.3 Action space

Each representation within the framework exhibits a distinct action space. The action spaces for the three representations are as follows:

- Narrow: the action space is characterized by two distinct actions: the no-action, which results in the agent skipping at the current location without making any changes, and the change tile action. The change tile action allows the agent to modify the tile at the current location, with the value of the action corresponding to one of the available tile types, which range from 0 to 7.
- Turtle: the action space is characterized by two distinct types of actions: movement actions and change tile actions. The movement actions allow the agent to navigate within the game environment by moving in one of four directions—up, down, left, or right. On the other hand, the change tile actions enable the agent to modify the tile at its current location. The value associated with the change tile action corresponds to one of the available tile types, which range from 0 to 7.
- Wide: the action space is characterized by two distinct types of actions: the affected location and the change tile action. The affected location refers to the specific coordinates (x and y position) within the game level where the action will take effect. The change tile action involves selecting a value ranging from 0 to 7, representing the available tile types.

6.4 Output

The model generates levels as 2D arrays, which pose challenges in terms of assessing their qualification. To facilitate a more convenient evaluation process, it is necessary to render these arrays into visual representations, allowing for easier inspection of their condition and adherence to specified criteria.



1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	0	1	4	1	1	3	0	0	1
1	4	6	4	0	0	0	1	0	0	1	0	1
1	0	0	1	0	0	1	1	0	5	1	0	1
1	0	1	0	0	0	1	1	0	1	0	0	1
1	3	1	0	1	0	0	1	0	0	5	0	1
1	0	2	0	7	3	0	0	0	0	0	0	1
1	0	1	0	1	1	0	0	0	3	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1

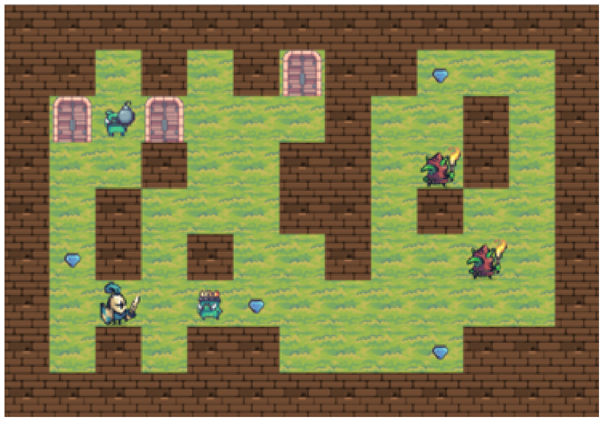


Figure 3: The generated level as 2D arrays (left) and the rendered level (right).

The respective integer representations assigned to each component are as follows:

- Empty tile: 0
- Solid tile: 1
- Player: 2
- Diamond: 3

- Door: 4
- Globlin: 5
- Bomber: 6
- King pig: 7

The employed algorithms in our study encompass the PPO and A2C algorithms. Each algorithm is trained using three representations. The training process for all models encompasses a total of 10 million time steps. In total, our study comprises six distinct models.

7 Experimental result

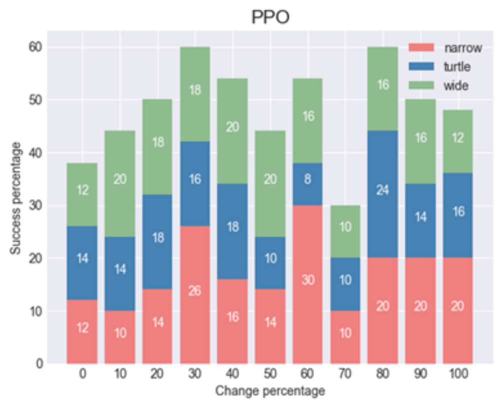
	Narrow	Turtle	Wide
PPO	14 hours 36 minutes	10 hours 39 minutes	12 hours 43 minutes
A2C	14 hours 8 minutes	13 hours 49 minutes	19 hours 32 minutes

Table 2: Comparing training time for each model

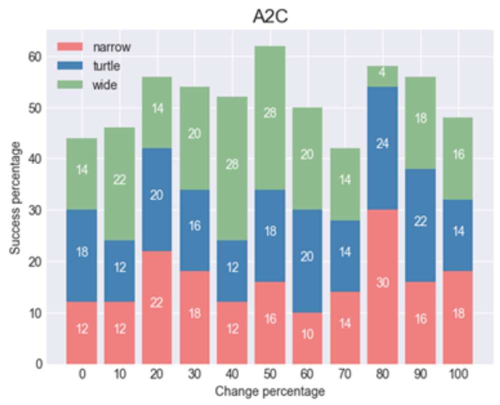
Table 2 presents the training time for each model. Among the evaluated models, the PPO algorithm with turtle representation exhibited the shortest training time, totaling 10 hours and 39 minutes. Conversely, the A2C algorithm with wide representation required the longest training duration, amounting to 19 hours and 32 minutes.

To evaluate the performance of the trained models, we generated a total of 50 levels for each model, varying the change percentage. The generated levels were subjected to specific criteria that must be satisfied, including the following conditions:

- Presence of a single player character.
- Inclusion of both a door and a diamond within the levels.
- Restriction on the number of enemies, ensuring they do not exceed 5.
- Provision of a viable path for the player to reach the diamond and subsequently access the door.



(a) PPO



(b) A2C

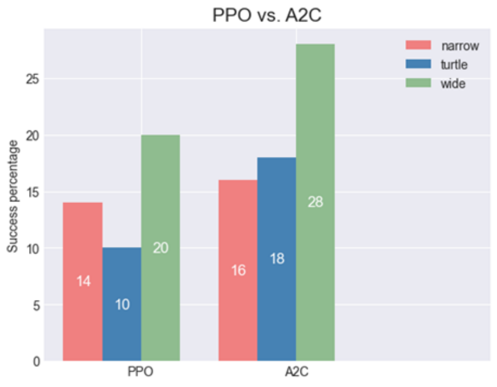
Figure 4: Comparing success percentage of PPO models and A2C models

Figure 4 shows the success percentage achieved by each model, with comparable values ranging between 4% and 28%.

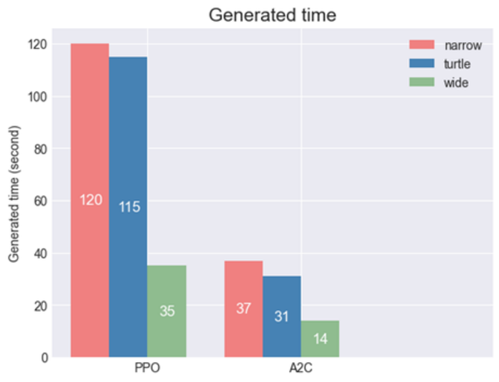
Table 3 shows the average success percentages for each model. Among them, the PPO algorithm with turtle representation demonstrates the lowest success percentage, standing at 14.73%. Conversely, the A2C algorithm with wide representation exhibits the highest success percentage, reaching 18%.

	Narrow	Turtle	Wide
PPO	17.45%	14.73%	16.18%
A2C	16.36%	17.27%	18%

Table 3: Average of success percentage



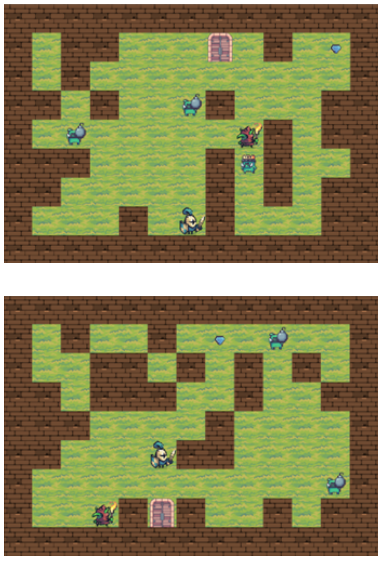
(a) Success percentage at 50% change percentage



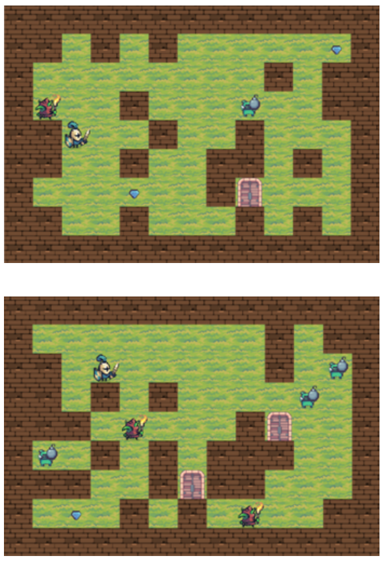
(b) Generated time

Figure 5: Comparing success percentage and generated time at 50% change percentage

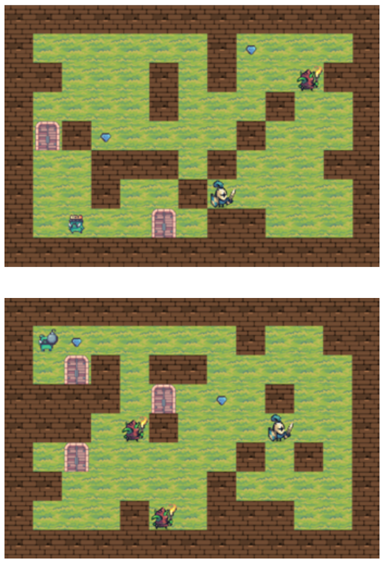
Figure 5 shows the success percentage achieved by each model under a change percentage of 50%, alongside the corresponding generated time for each model. Notably, the PPO algorithm with narrow representation exhibits the longest generated time, amounting to 120 seconds. Conversely, the A2C algorithm with wide representation demonstrates the shortest generated time, requiring only 14 seconds.



(a) Narrow



(b) Turtle

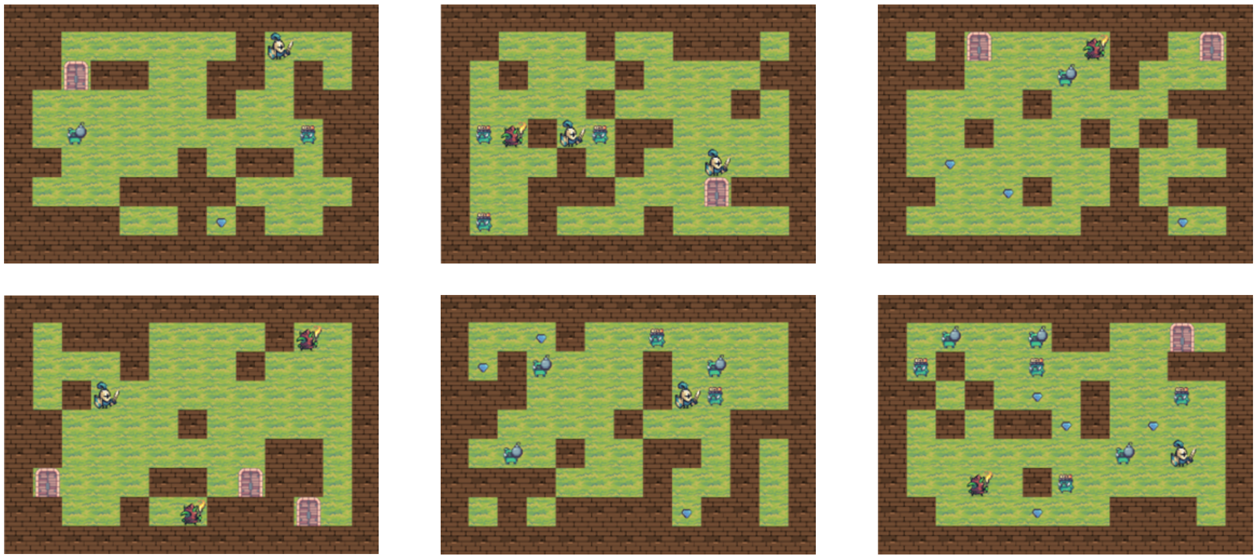


(c) Wide

Figure 6: Example of successfully generated levels from PPO (top) and A2C (bottom) models

Figure 6 shows illustrative examples of successfully generated levels, each demonstrating the attainment of the desired objective where the player is able to navigate to the diamond and subsequently reach the door. These examples serve to visually depict the effectiveness of the generated levels in satisfying the specified conditions.

Figure 7 shows representative examples of unsuccessful generated levels, highlighting instances where specific conditions were not met. In the case of the PPO algorithm, the narrow representation example depicts a scenario where



(a) Narrow (b) Turtle (c) Wide

Figure 7: Example of successfully generated levels from PPO (top) and A2C (bottom) models

the player fails to reach the diamond. In the turtle representation, the example exhibits the presence of two players, violating the requirement of having only one player. Lastly, in the wide representation example, the absence of a player contradicts the specified conditions.

Similarly, for the A2C algorithm, the narrow representation example showcases a lack of a diamond, while the turtle representation example lacks a door. In the wide representation, the presence of more than five enemies exceeds the max limit.

8 Conclusions and Future Work

According to the results, it is evident that the success percentages achieved by both the PPO and A2C algorithms are comparable. However, notable differences arise in terms of the generated time. Specifically, the A2C algorithm outperforms the PPO algorithm in terms of speed, with the A2C algorithm demonstrating faster generation times. Specifically, when considering the narrow representation, the A2C algorithm is approximately three times faster than the PPO algorithm. In the case of the turtle representation, the A2C algorithm showcases a speed advantage of roughly four times. Moreover, when employing the wide representation, the A2C algorithm is approximately 2.5 times faster compared to the PPO algorithm.

For further enhance the performance of the models, we propose two potential avenues for improvement. Firstly, extending the training duration could potentially yield better results by allowing the models to acquire more refined strategies and increase their success rates. Alternatively, training an additional agent to play the generated levels and providing feedback to the generator model can facilitate an iterative improvement process. This feedback loop can contribute to refining the generator model's output and enhancing the overall quality of the generated levels. Ultimately, the culmination of these efforts could involve integrating the generator model into a real game using popular game engines such as Unity or Unreal Engine, thereby showcasing its practical application in a tangible gaming environment.

References

[1] Pettersson L. Schneider J. Schulman J. Tang J. Brockman G., Cheung V. and Zaremba W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[2] Mirza M. Graves A. Harley T. Lillicrap T. P. Silver D. Mnih V., Badia P. A. and Kavukcuoglu K. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

[3] Dhariwal P. Radford A. Schulman J., Wolski F. and Klimov O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[4] Klimov O. Nichol A. Plappert M. Radford A. Schulman J. Sidor S. Wu Y. Dhariwal P., Hesse C. and Zhokhov P. Openai baselines. URL `https://github.com/openai/baselines`.

[5] Sutton R. S. and Barto A. G. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning. The MIT Press, 2018.

[6] Green M. C. Bhaumik D., Khalifa A. and Togelius J. Tree search vs optimization approaches for map generation. *arXiv preprint arXiv:1903.11678*, 2020.

[7] Earle S. Khalifa A., Bontrager P. and Togelius J. Pcgrl: Procedural content generation via reinforcement learning. *arXiv preprint arXiv:2001.09212*, 2020.

[8] Newzoo. Games market numbers revenues and audience 2020-2023. URL `https://newzoo.com/resources/blog/newzoo-games-market-numbers-revenues-and-audience-2020-2023`.

[9] Bontrager P. and Togelius J. Learning to generate levels from nothing. *arXiv preprint arXiv:2002.05259*, 2021.

[10] Digital Economy Promotion Agency. Investment quarterly gaming. URL `https://www.depa.or.th/storage/app/media/file/investment-quarterly-gaming-industry220164-Final.pdf`.

[11] Liu J. Shu T. and Yannakakis J. N. Experience-driven pcg via reinforcement learning: A super mario bros study. *arXiv preprint arXiv:2106.15877*, 2021.

[12] Hugging Face. Deep rl course. URL `https://huggingface.co/learn/deep-rl-course/unit0/introduction?fw=pt`.