Adversarial Reinforcement Learning for Procedural Content Generation (ARLPCG)

Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, Konrad Tollmar
SEED, Frostbite - Electronic Arts (EA)

Marty Mukherjee

University of Waterloo

Table of contents

1. Introduction

2. Content of paper

3. Results

4. Conclusion

Introduction

The problem

Training an RL agent on a single fixed game environment can lead to the agent overfitting to that environment and being unable to adapt to unseen environments

In game development, the environment and assets change on a regular basis

Adaptability and generalization ability are desirable in games with player-created content and games that require interaction with human players

Proposed solution

Training agents on environments changing on a continuous basis in order to generalize to unseen environments

Real-time procedural content generation (PCG)

Content of paper

Proposed solution

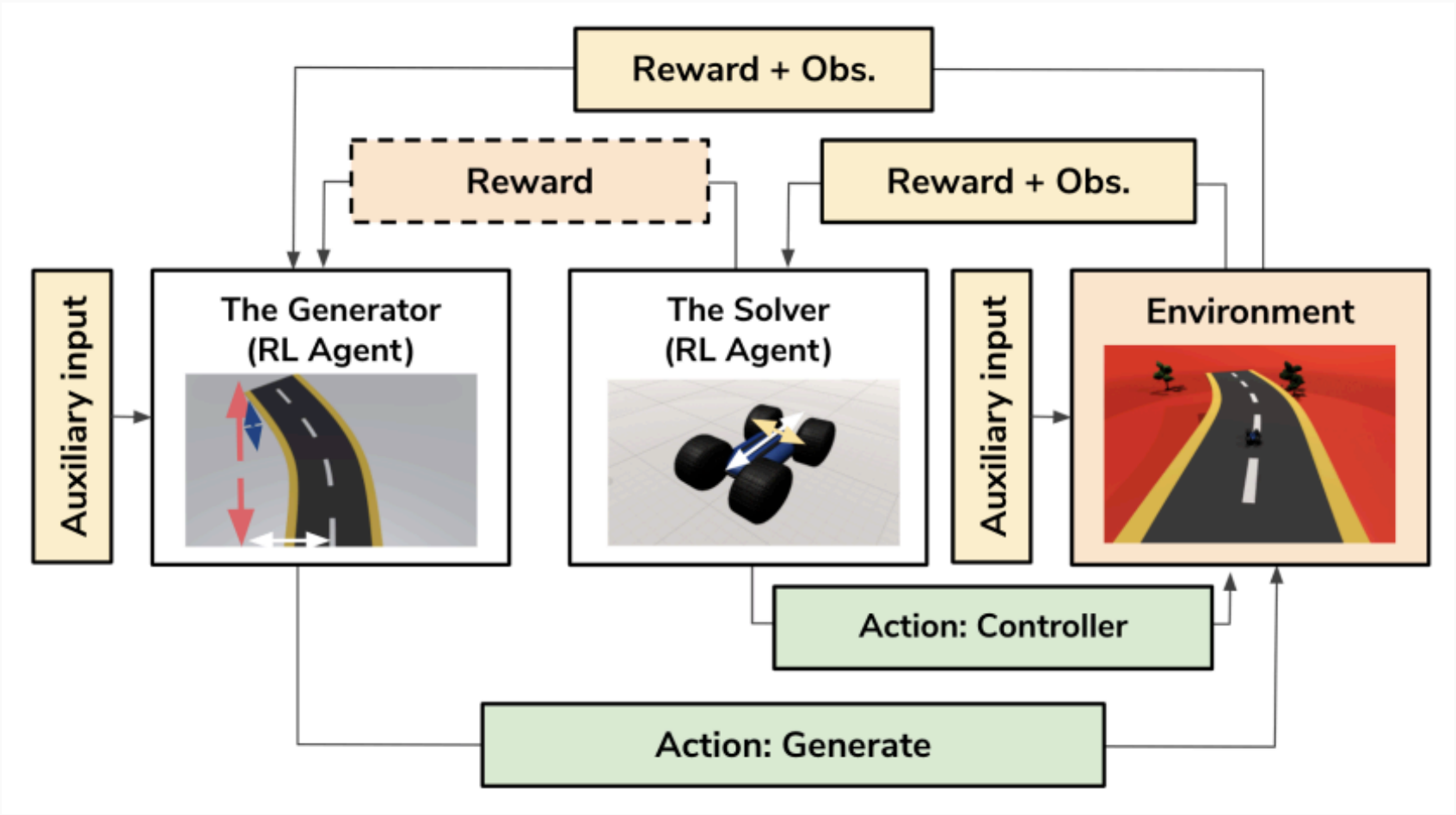


Figure 1: High level view of the architecture of ARLPCG (Gisslén et al. 2021)

The Generator

RL Agent - PPO
Goal: Train a generator to make the solver reach a goal as ”sub-optimally” as possible. Find a balance between impossible and trivial environments.

- Auxiliary input ($\lambda \in [-1, 1]$): Controls the reward function - difficulty and behaviour. As a result, it indirectly controls the generator as well.
  $\lambda = 1$: Very Easy
  $\lambda = 0$: Moderate
  $\lambda = -1$: Very Difficult

The Generator

- Reward structure:

$$r = r_{int} \sum_{i=0}^{n} \lambda_{A_i} \alpha_i + r_{ext} \sum_{i=0}^{n} \lambda_{A_i} \beta_i \tag{1}$$

  Where $\lambda_{A_i} \in [-1, 1]$ is the auxiliary input. r_{int}: internal reward, only depends on the generator output. r_{ext}: external reward, depends on the performance of the solver. α_i, β_i are weighing factors.

The Solver

RL agent - PPO

Reward: Positive reward for progression from one segment to the next as fast as possible. Negative reward for failing. This negative reward term forces the generator to stop creating impossible environments.

Previous work - PCGRL: Procedural Content Generation via Reinforcement Learning

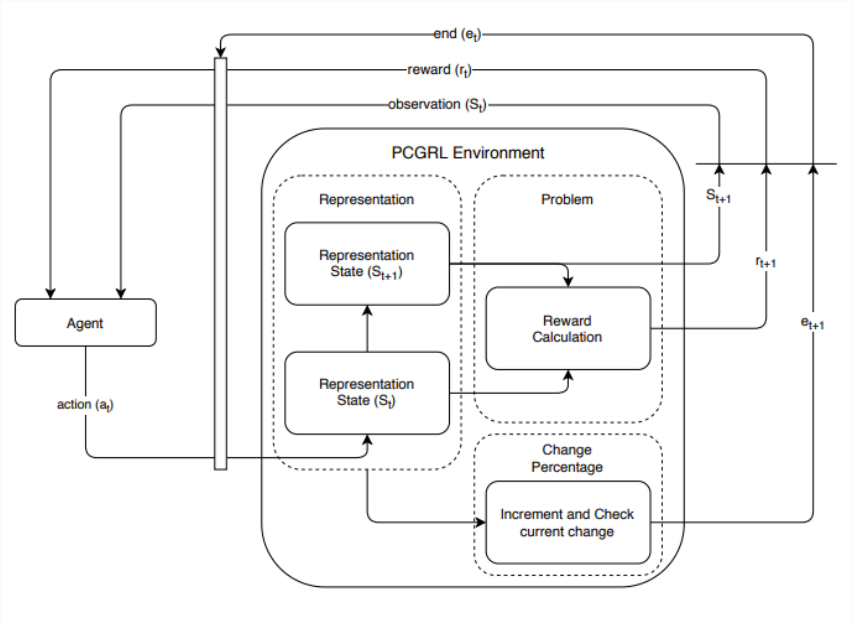


Figure 2: The system architecture for PCGRL (Khalifa et al. 2020)

Difference: PCGRL generates new environments by modifying an existing observation S_t. ARLPCG generates environments from scratch using an auxiliary input.

Results

Empirical Evaluation

1. Can the Generator create environments that can be controlled with an auxiliary input?
2. Are the Solvers trained with the ARLPCG better at generalization than other approaches?
   – Compared to Solvers trained on a fixed set of environments
   – Compared to Solvers trained on environments determined by rules

Video: https://youtu.be/z7q2PtVsT0I

Experiments - Platform

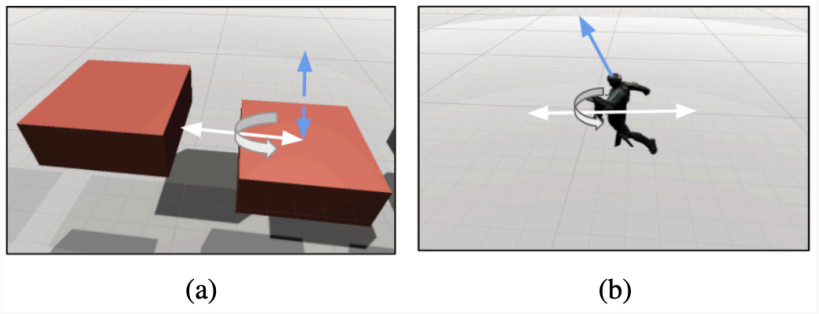


Figure 3: Platform game. (a) The Generator creates blocks. (b) The solver controls the movement of the humanoid. (Gisslén et al. 2021)

The generator decides (a) The distance of the new block from the current block (b) The direction of the new block (c) The vertical displacement
The solver decides (a) Move forward (b) Move backward (c) Turn left/right (d) Jump

Experiments - Driver

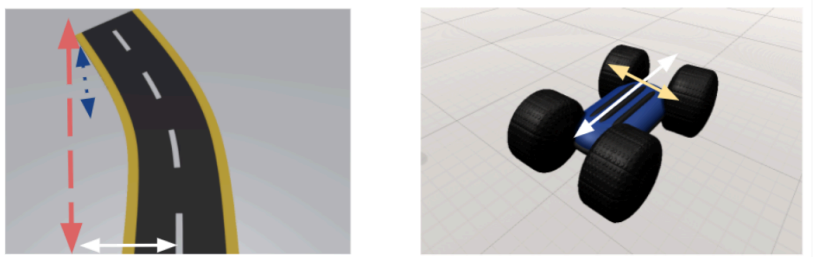


Figure 4: Driver game. (a) The Generator creates roads. (b) The Solver drives the car. (Gisslén et al. 2021)

The generator decides (a) The length of the segment (b) The horizontal curvature (c) The vertical curvature
The solver decides (a) Accelerate/Decelerate (b) Turn left/right

Reward function

Platform game:

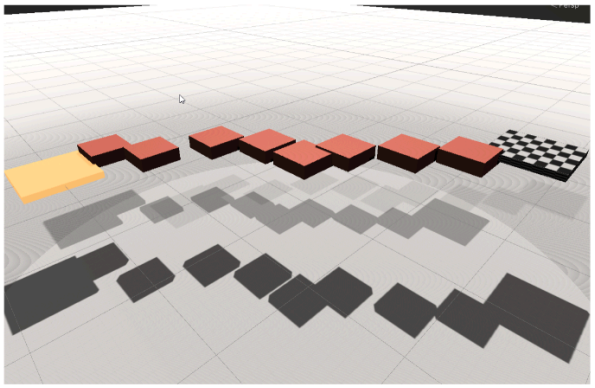
- The generator receives an additional reward of $\lambda_{A_i} \times 10$ every time the solver fails

Driver game:

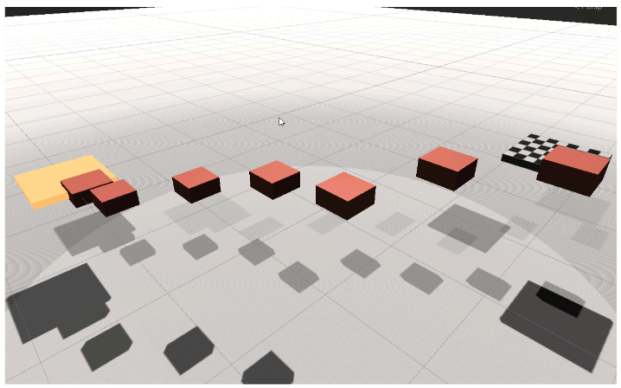
- If $\lambda_{A_i} < 0$, the generator receives an additional positive reward for every time step the vehicle is above a certain threshold above the ground

The generator receives a small negative reward per time step. This forces it to generate an environment so that the solver finishes it or fails it fast.

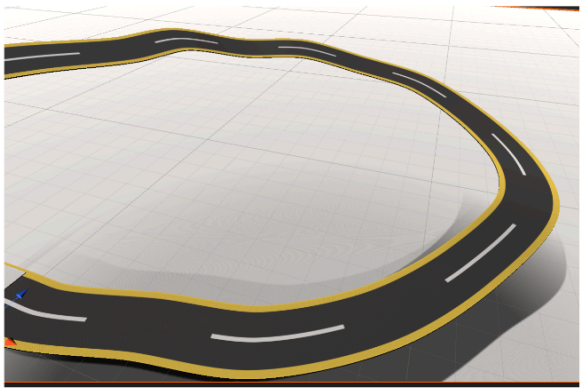
Evaluation - Generator



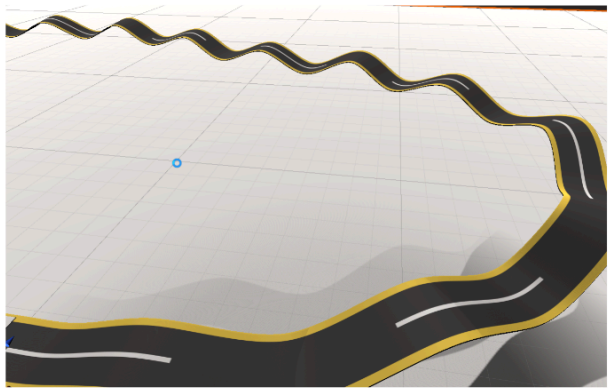
(a) Auxiliary input of value 1.



(b) Auxiliary input of value -1.



(a) Auxiliary input of value 1.



(b) Auxiliary input of value -1.

Figure 5: Examples of environments generated with different auxiliary input values. (Gisslén et al. 2021)

Evaluation - Solver

Aux. value	Fixed Track	Rule PCG	ARLPCG
1 (Easy)	0.0±0.0	0.776±0.154	0.824±0.185
0.5	0.0±0.0	0.692±0.350	0.741±0.285
0 (Moderate)	0.0±0.0	0.386±0.235	0.728±0.284
-0.5 (Hard)	0.0±0.0	0.148±0.165	0.360±0.276
-1 (Hardest)	0.0±0.0	0.053±0.037	0.183±0.139

Figure 6: Average success ratio for the platform game over 50000 trials (Gisslén et al. 2021)

Aux. value	Fixed Track	Rule PCG	ARLPCG
1 (Easy)	0.743±0.369	0.788±0.301	0.997±0.009
0.5	0.507±0.402	0.632±0.422	0.890±0.249
0 (Moderate)	0.280±0.349	0.622±0.425	0.772±0.340
-0.5 (Hard)	0.281±0.336	0.460±0.441	0.643±0.392
-1 (Hardest)	0.206±0.322	0.425±0.444	0.613±0.430

Figure 7: Average success ratio for the driver game over 100000 trials (Gisslén et al. 2021)

Conclusion

Contributions

- Generalization of RL agent on changing environments
- Generation of environments using auxiliary inputs

Future work

- Potential improvements in the structure of auxiliary input
- Using multiple solvers instead of one