

Multi-Agent Proximal Policy Optimization on Flatland 3

Presented by Edoardo Merli



AMLD



FLATLAND 3

Multi-Agent Reinforcement
Learning on Trains

AMLD 2021 Competition
NeurIPS 2020 Competition



2 DJI
Drones

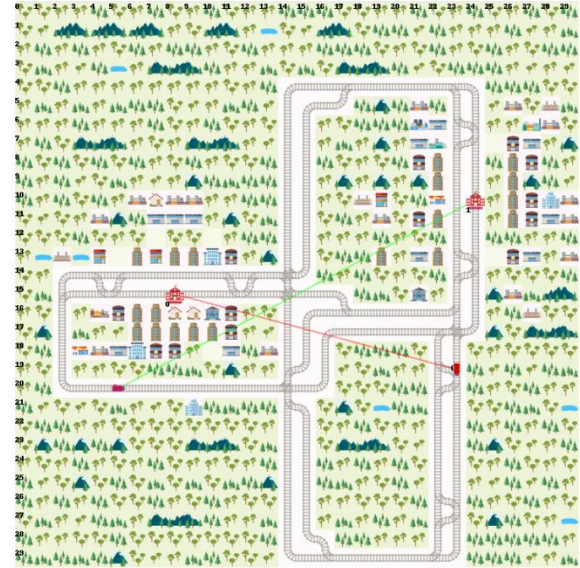


4 Oculus
Quest 2's



The Flatland 3 challenge

- Simulates a railway network on a 2D grid environment
- Goal: route as many trains as possible from departure to target station, as fast as possible
- Trains have different speeds and can experience temporary malfunctions
- Avoid collisions as they make paths unusable, causing deadlocks



$$R_i = \begin{cases} 0, & \text{if } T_i \leq A_i \\ A_i - T_i, & \text{if } A_i < T_i \leq T_{\max} \\ A_i - T_{\max} - d_i^{(T_{\max})}, & \text{otherwise} \end{cases}$$

Our approach – Multi-agent RL with MAPPO

$$L^{TRPO}(\theta) = -\hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \quad L^{CLIP}(\theta) = -\hat{\mathbb{E}}_t [\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t)]$$

Algorithm 1 PPO

Input: Initialized $\pi_\theta, \pi_{\theta_{\text{old}}}, V_\phi$

for $iter = 1, \dots, N$ **do**

 Collect T transitions by following π_θ

 Compute advantage estimates $\hat{A}_1, \dots, \hat{A}_T$ using V_ϕ

for $epoch = 1, \dots, M$ **do**

 Optimize $L^{CLIP}(\theta)$ wrt θ , using π_θ and frozen $\pi_{\theta_{\text{old}}}$

 Optimize $L^{MSE}(\phi)$ wrt ϕ with $\hat{A}_1, \dots, \hat{A}_T$ as targets

$\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$

end for

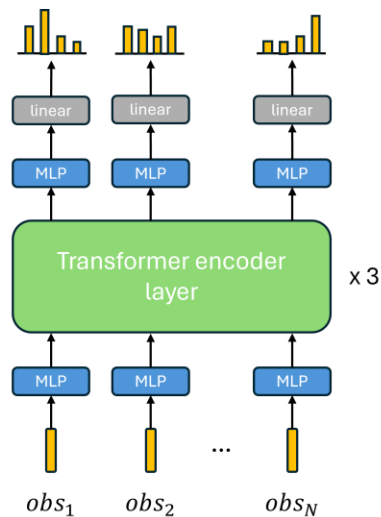
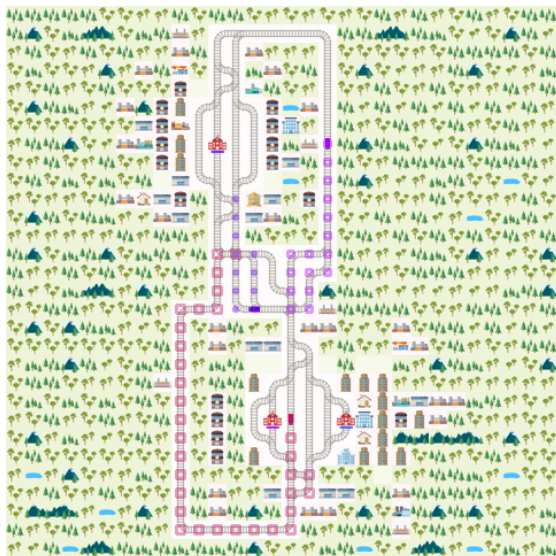
end for

Multi-Agent PPO (MAPPO):

- the policy network is applied on each agent's observation individually
- the value network is applied globally and returns a single value for all agents
- rewards are shared (in our scenario, to promote cooperation)

Our approach – Observations & neural networks

- Flatland 3 provided a few observation types, but we designed our own
- Two versions: max pooled at depth 2 for efficiency (**v1**) or not (**v2**)
- Implemented two different neural network architectures
- **MLP** (lighter but no “message passing”) and **RailTransformer** (heavier but interaction between agents’ activations)



Our approach – Custom reward function

- Often beneficial to design a dense custom reward function in RL
- Optimized as a proxy of the sparse true reward function

$$\hat{A}_t = \sum_{i|done_i^{(t)}} 1 - \frac{\max(t - l_i, 0)}{T_{max}}$$

$$R_t = 0.1 \frac{D_t}{N} + 5 \frac{\hat{A}_t}{N} - 2.5 \frac{L_t}{N}$$

Normalized new departures

Normalized new arrivals +
late arrival penalty

Normalized new deadlocks

Training setup & Results

Training: on 3 different environment configurations («demo», «small», «mini»), respectively with 5, 20 and 50 agent and other hyperparameters identical to those then used for testing

```
,env_size,n_agents,x_dim,y_dim,n_cities,max_rail_pairs_in_city,grid_mode,max_rails_between_cities,malfunction_duration_min,malfunction_duration_max,malfunction_interval,speed_ratios,random_seed
0,demo,5,30,30,2,3,False,2,20,50,1000,"{1.0: 0.25, 0.5: 0.25, 0.33: 0.25, 0.25: 0.25}",0
1,mini,20,35,35,3,2,False,2,20,50,1000,"{1.0: 0.25, 0.5: 0.25, 0.33: 0.25, 0.25: 0.25}",6
2,small,50,40,40,3,2,False,2,20,50,1000,"{1.0: 0.25, 0.5: 0.25, 0.33: 0.25, 0.25: 0.25}",22
```

Trained every combination of:

- **MLP vs RailTransformer**
- **Observation v1 vs v2**
- **Training from scratch vs curriculum learning**

model	demo		mini		small	
	true episodic reward	percentage done	true episodic reward	percentage done	true episodic reward	percentage done
MLP + obs v1	0.84	0.67	0.72	0.36	(curr) 0.58	(curr) 0.058
MLP + obs v2	0.89	0.78	0.70	0.34	(curr) 0.59	(curr) 0.081
RailT + obs v1	0.81	0.62	0.61	0.19	0.55	0.043
RailT + obs v2	Train Div.	Train Div.	0.62	0.21	0.54	0.023

Evaluation setup & Results

Evaluation: locally using the flatland-evaluator package and the test env configurations provided

model	sum normalized reward	mean percentage done
MLP + obs v1	29.22	0.36
MLP + obs v2	28.42	0.34
RailTransformer + obs v1	20.34	0.28
RailTransformer + obs v2	21.99	0.39

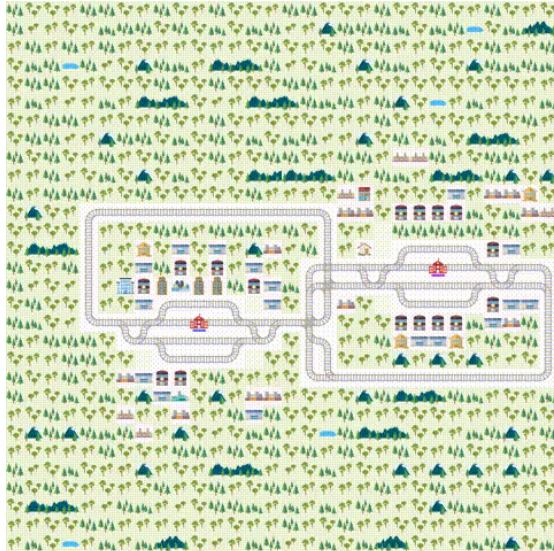
- **MLP + obs v1** performs best
- **MLP** beats **RailTransformer** by a large margin

Our model reaches an «hypothetical» **first place** on the RL track of the Flatland 3 challenge! 🏆

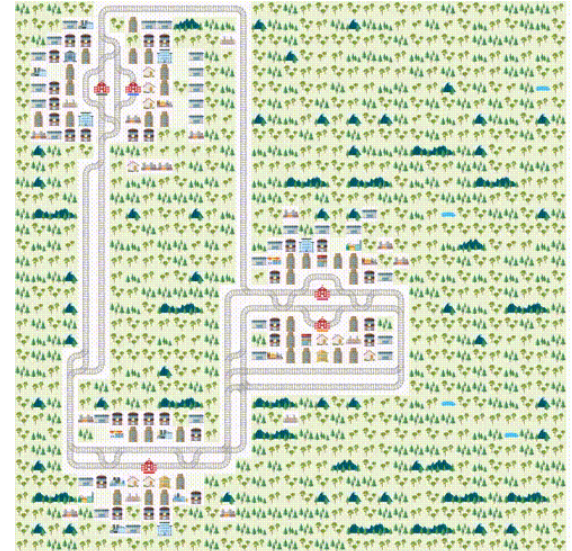
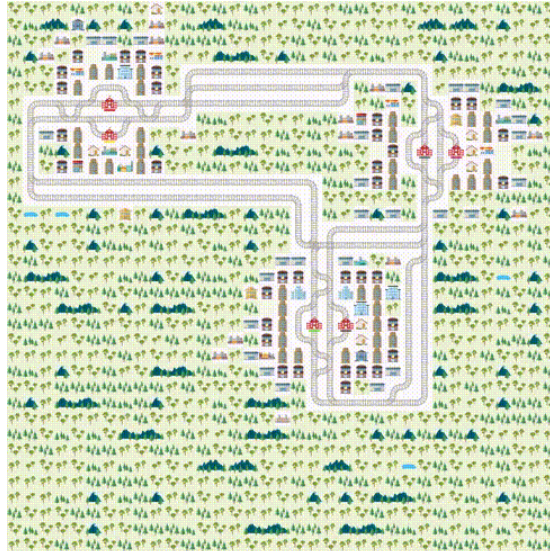
participant	sum normalized reward	mean percentage done
our (MLP + obsv1)	29.22	0.36
WaveTeam	27.868	0.386
SOBA	27.786	0.326
ChewChewChew	19.99	0.306
traintrains	6.71	0.00
g5dna	6.71	0.00

Some videos

«Demo» env



«Mini» env



Conclusions

Limitations

- The result is great among RL approaches, but Operations Research solutions scored up to 4x higher on the evaluation benchmark
- There might still be room for improvement on the RL side of Flatland 3
- Our approach could be made more performant with parallel trajectory collection

Bottom line

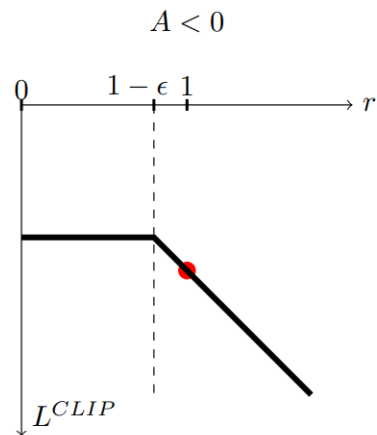
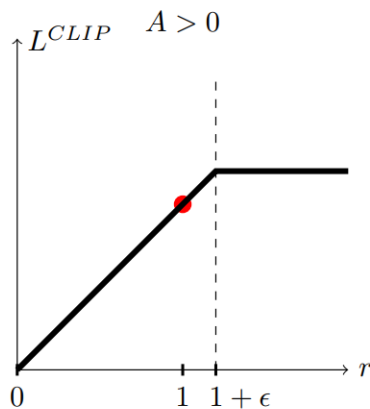
- I'm happy to have reached a tangible result despite the low compute available
- Coding this was tough and took waay more time than expected, but...
it has been extremely rewarding and I learned a lot 😊

Thank you for your attention

PPO's clipped surrogate objective function

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

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$



Generalized Advantage Estimation

$$\hat{V}_t^{(N)} = \sum_{i=t}^{t+N-1} \gamma^{i-t} r_i + \gamma^N V(s_{t+N}) \quad \text{TD(N) target}$$

$$\hat{A}_t^{GAE(\gamma, \lambda)} = (1 - \lambda) \sum_{N>0} \lambda^{N-1} \left(\hat{V}_t^{(N)} - V(s_t) \right) \quad \text{GAE as EWA of TD(N)}$$

Value targets for the Value network

$$\hat{V}_t^{GAE} = (1 - \lambda) \sum_{N>0} \lambda^{N-1} \hat{V}_t^{(N)} \approx V^\pi(s_t)$$

Observation details

	Description	Dim.
Agent features	Train state (one-hot encoded)	7
	Agent is at switch	1
	Agent is near any switch	1
	Agent is one step before a decision switch	1
	Agent is one step before any switch	1
	Agent’s speed (one-hot encoded)	4
	Agent is deadlocked	1
for each one of the 4 directions:		
Subtree features	for each one of the 4 sub-branches	
Subbranch features	there is a transition	1
	there is a path towards the target	1
	that path is the shortest path to the target	1
	there is an opposite-direction agent	1
	there is a same-direction agent	1
	the same-direction agent is at least as fast as this one	1
	the other agent is deadlocked	1
	the other agent is in a “ready to depart state”	1
	the other agent is malfunctioning	1
	the agent’s target is on this path	1