Centralized control for multi-agent RL in a complex Real-Time-Strategy game

Author: Roger Creus Castanyer

Published: arXiv e-prints, **arXiv:** 2304.13004
on 25 Apr 2023

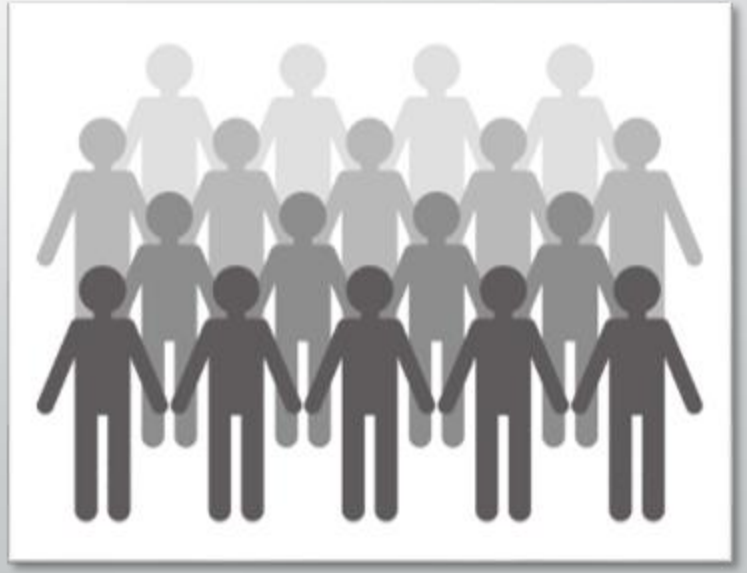
Presenter: Haruto Tanno

Introduction

- RL success in complex domains
- Growing interest in MARL
- RTS games are challenging environments

Problem Setting

- Multiple agents in shared environment
- Observations depend on other agents
- Non-stationary dynamics



Lux AI Environment

- Kaggle competition (Lux AI v2)
- 1 vs 1 game
- Resource gathering & allocation

Lux AI Environment

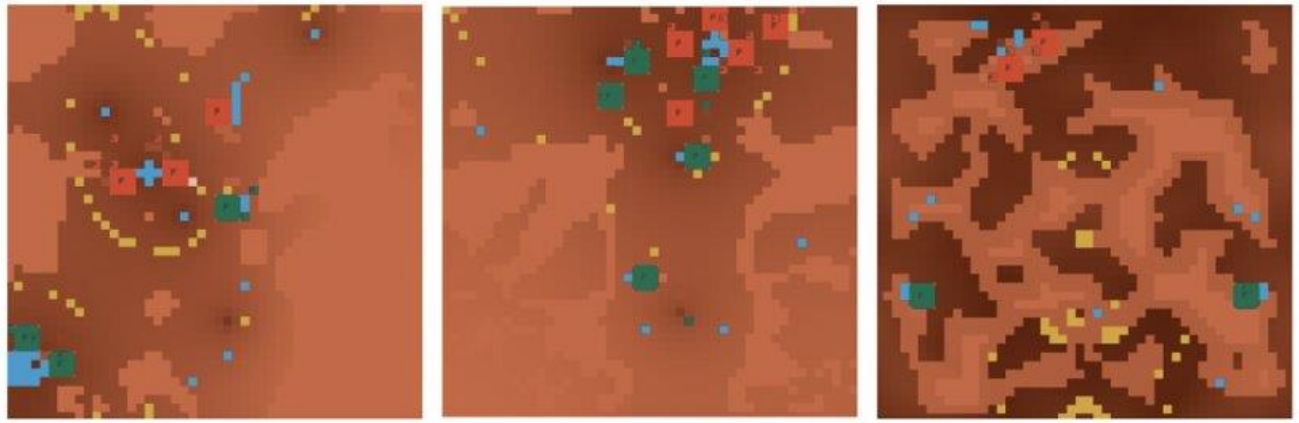


Figure 1: Three different 48x48 maps of the *Lux AI v2* environment. Player1 and Player2 compete for resources in pseudo-randomly generated maps. Yellow cells indicate *ore*, blue cells indicate *ice*, brown cells indicate *ruble*, 3x3 squares with the letter F are *factories*, and little squares with the letters L and H are *light* and *heavy* robots respectively.

Challenges

- Entity-based observations
- Variable number of units
- Different action spaces
- Sparse resources
- Non-stationary environment

Core Approach

- Centralized training
- Single-agent formulation
- Proximal Policy Optimization (PPO)

Observation Design

- 48 × 48 grid
- 17 feature maps
- Encodes units, factories, resources

Observation Design

Table 1: Description of the 17 feature maps that conform the observations fed to the agent. Negative values always indicate properties related to the opponent (e.g. a -200 in unit_ice means an enemy unit has 200 ice in it's cargo. Each of the feature maps is the same size of the map 48x48.

Name	Description	Range	Normalization
is_day	Whether the current turn is day or night	[0, 1]	-
rubble	Amount of rubble in each cell	[0, 100]	100
ore	Whether there is ore in each cell	[0, 1]	-
ice	Whether there is ice in each cell	[0, 1]	-
lichen	Amount of lichen in each cell	[0, 100]	100
is_resource	Whether there is ore or ice in each cell	[0, 1]	-
light_units	Whether there is a light robot in each cell	[-1, 0, 1]	-
heavy_units	Whether there is a heavy robot in each cell	[-1, 0, 1]	-
unit_ice	Amount of ice in the robot's cargo	[-3000, 3000]	3,000
unit_ore	Amount of ore in the robot's cargo	[-3000, 3000]	3,000
unit_power	Amount of power in the robot's battery	[-1000, 1000]	1,000
unit_on_factory	Whether each robot is on top of a factory	[-1, 0, 1]	-
factories	Whether there is a factory in each tile	[-1, 0, 1]	-
factory_ice	Amount of ice in each factory	$[-\infty, \infty]$	150
factory_ore	Amount of ore in each factory	$[-\infty, \infty]$	150
factory_water	Amount of water in each factory	$[-\infty, \infty]$	150
factory_metal	Amount of metal in each factory	$[-\infty, \infty]$	150

Action Space

- MultiDiscrete actions
- Separate for robots and factories
- Invalid action masking

Action Space

Table 2: Description of the robots' action space

Component	Description	Range
Source Unit	location of the selected unit to perform an action	$[0, h \times w - 1]$
Action Type	NOOP, move, transfer, pickup, dig, self-destruct	[0, 5]
Move param.	up, right, down, left	[0, 3]
Transfer param.	center, up, right, down, left	[0, 4]
Transfer amount	25%, 50%, 75%, 95%	[0, 3]
Transfer resource	ice, ore, power	[0, 2]

Table 3: Description of the factories' action space

Component	Description	Range
Source Unit	location of the selected unit to perform an action	$[0, h \times w - 1]$
Action Type	NOOP, build_light, build_heavy, grow_lichen	[0, 3]

Reward Design

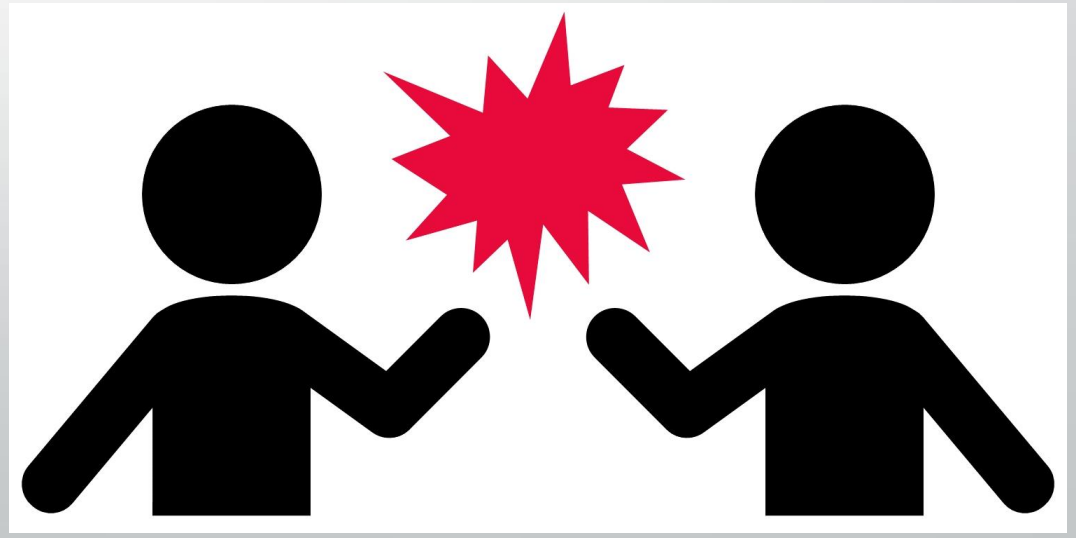
- Sparse reward issue
- Reward shaping
- Focus on water production

Reward Design

- Water generation is key
- Dense reward → sparse reward (later)

Self-Play

- Prioritized Fictitious Self-Play
- Play against past versions



Architecture

- Pixel-to-pixel CNN
- ResNet backbone
- Actor-Critic

Architecture

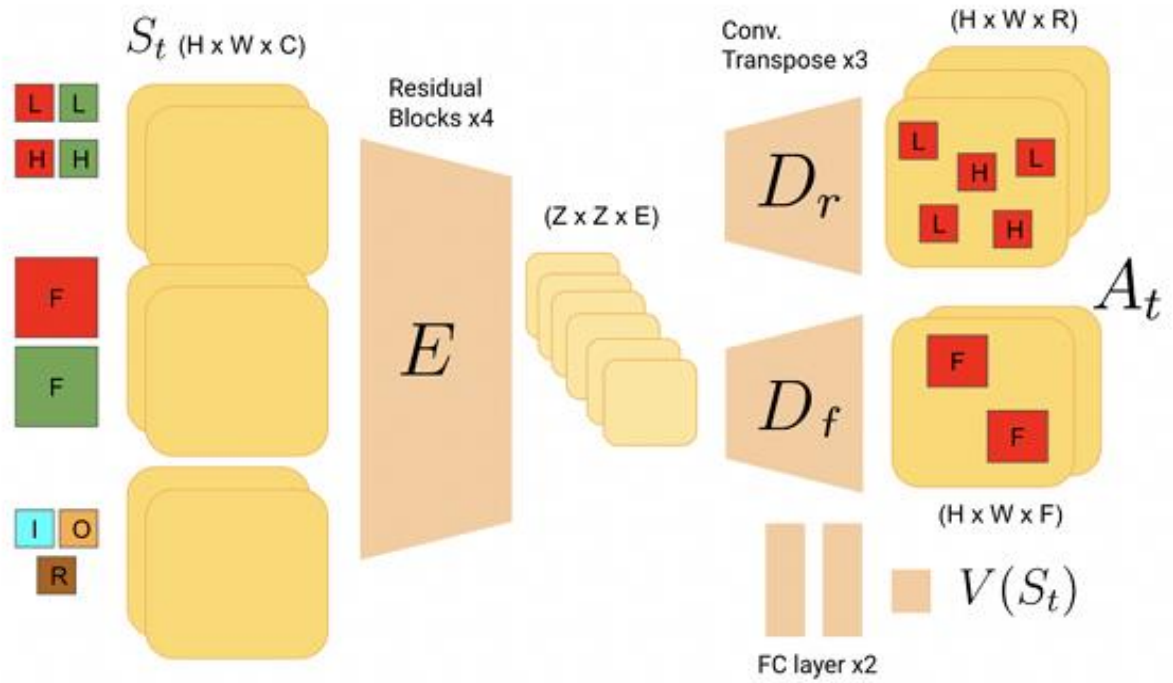
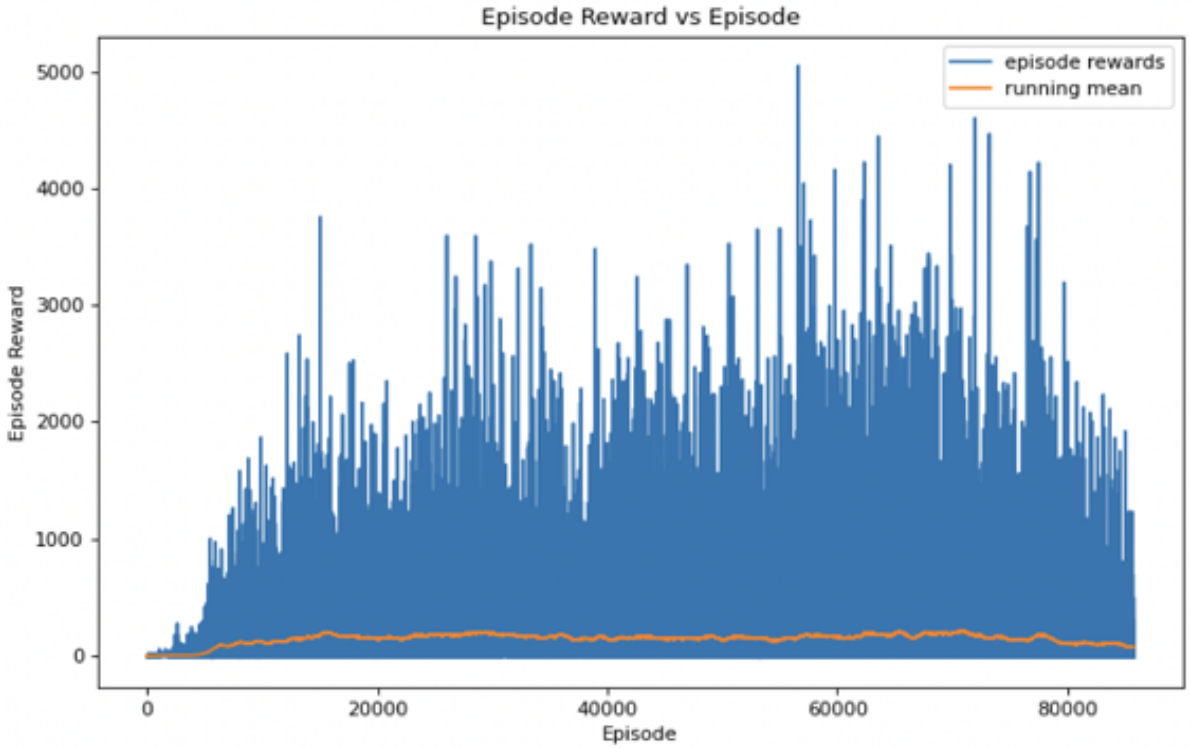


Figure 2: Our pixel-to-pixel architecture. We feed the model with a stack of C feature maps containing information about the units, factories and resources. We use a Residual Network [He et al., 2016] with squeeze-and-excitation layers [Hu et al., 2018] to encode the observations into low-dimensional representations. We feed the representations to three different heads: the critic which outputs a single scalar value representing the state value; the *robot actor* which outputs feature maps with as many dimensions as robot action components; and the *factories actor* which outputs as many actions as factory action components.

Experiments

- 220M steps
- High computational cost
- High variance results

Experiments



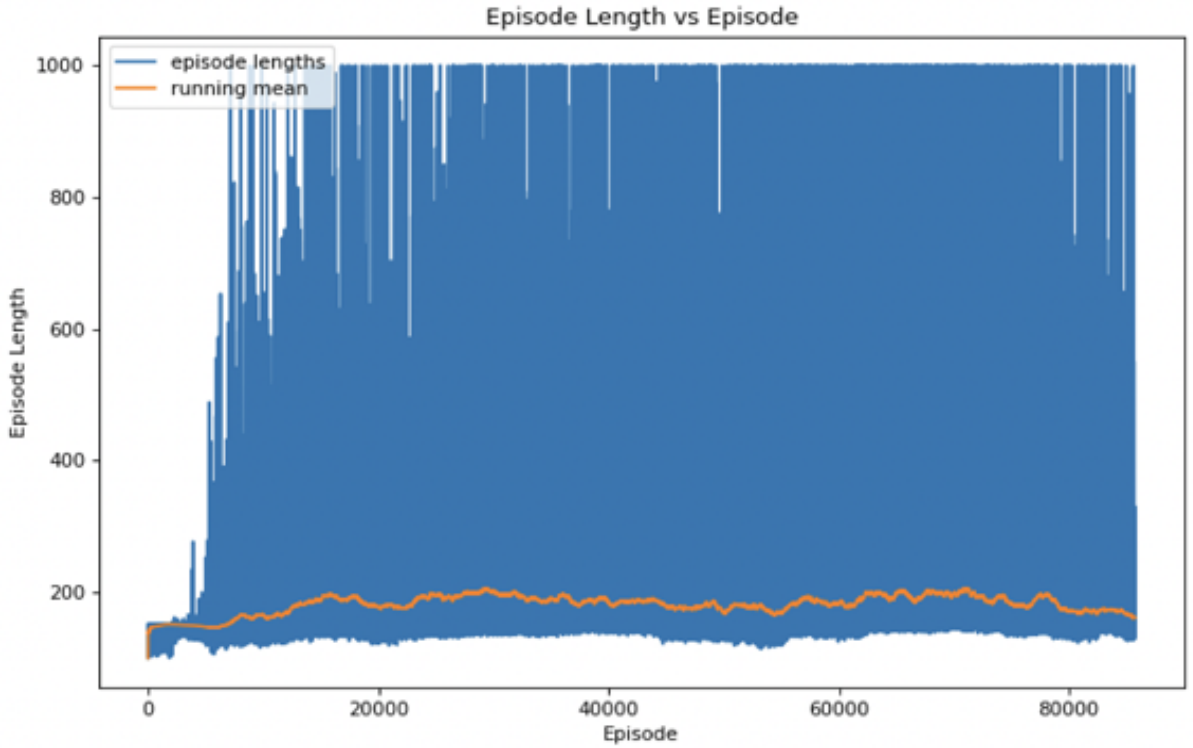


Figure 3: The figure shows the episode reward (left) and episode length (right) during training. The agent is trained for 220M environment interactions. Although the agent learns to gather resources and survive for the 1,000 steps of the game in many episodes, there is a high degree of variance. Since each episode has a significantly different map and a variable number of factories, it is hard for the agent to generalize.

Conclusion

- Centralized approach works
- RTS remains difficult
- Many open challenges