Comparing Observation and Action Representations for Reinforcement Learning in μRTS

Shengyi Huang and Santiago Ontañón

Drexel University

{*sh3397,so367*}*@drexel.edu*

October 3, 2019

Overview

Problem Statement

- **Problem**: how do we design observation and action representation of RTS games for DRL (Deep Reinforcement Learning)?

This paper:

- We compared two intuitive representations:
    - a *global* representation where the observation represents the whole game state, and the RL agent needs to choose which unit to issue actions to, and which actions to execute
    - a *local* representation where the observation is represented from the point of view of an individual unit, and the RL agent picks actions for each unit independently.

RTS Games

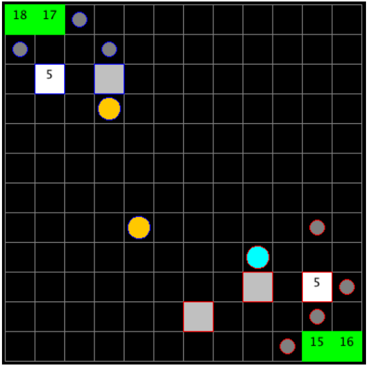
Figure: StarCraft II (PC, 2010)

RTS Games

- RTS Games have the characteristics of *real-time*:
  - Real-time decision making
  - Extremely large branchng factor for seach-based approaches
  - Hundreds of unit to control
  - Sparse rewards for RL
  - Partially-observable

μRTS

- Open-source https://github.com/santiontanon/microrts
- Minimalistic
- Deterministic
- Fully or partially-observable options
- Available AI techniques: UCT, NaiveMCTS, Portfolio greedy search, AHTN, ...



Existing Work

RL techniques in RTS games:

- Alisp to specify a list of commands, and Q-learning to tune the parameters [Marthi et al., 2005] in Wargus.
- Q-learning to learn a policy for each class of units (peasants, knights, barracks, etc) [Marthi et al., 2005] in Wargus.
- Option framework and heuristic algorithms to simplify the game space [Tavares and Chaimowicz, 2018].

Existing Work

Recent development utilizing neural networks with RL:

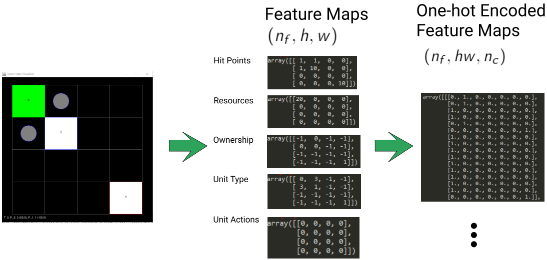
- Provide feature maps and low-level actions to train agents. [Vinyals et al., 2017]
- High-level actions and curriculum. training[Tian et al., 2017, Sun et al., 2018, Lee et al., 2018]
- HRL (Hierarchical Reinforcement Learning) [Liu et al., 2019].
- Multi-Agent representation [Samvelyan et al., 2019]

Possible Representations

- Observations
  - Raw pixels?
  - Discretized states (A binary array of attributes“haveBarracks”, “haveRanged”, “isAttacking”)?
  - Feature maps?
  - Localized feature maps with camera movement?
- Actions
  - Mouse clicks and key strokes?
  - Give commands to control each unit (soilder1: attack, soilder2: noop)?
  - Give commands to select interested unit and then control it?

Global Representation

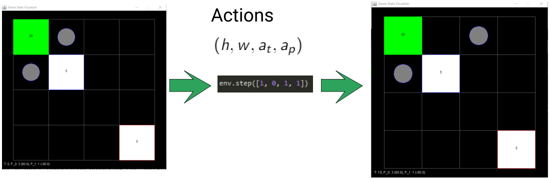
- **Intuition:** let the agent *see* and *control* everything.
- **Implementation:**



where n_f denotes the number of feature maps, h, w the height and width of the map, and n_c the number of feature planes

Global Representation

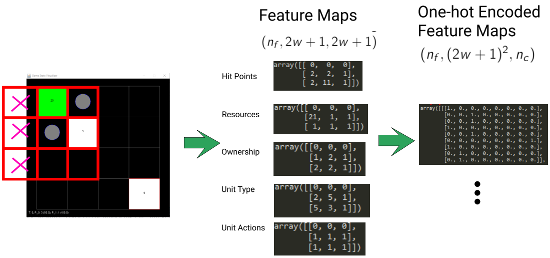
- **Intuition:** let the agent *see* and *control* everything.
- **Implementation:**



where a_t and a_p denotes the action type and action parameteres, respectively.

Local Representation

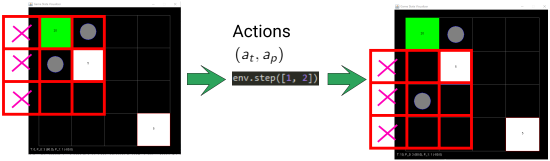
- **Intuition:** let the agent be each unit and *see* everything around it and *control* itself.
- **Implementation:**



where w is the window size of the local unit observation.

Local Representation

- **Intuition:** let the agent be each unit and *see* everything around it and *control* itself.
- **Implementation:**



where a_t and a_p denotes the action type and action parameters, respectively.

Other Details and Insights

- If the action produced is invalid, then it will be replaced by an NOOP action.
- The agent can only issue one action to one unit at each timestep.
- The local representation rotates through units and asks the RL agent to produce actions for the selected unit only.

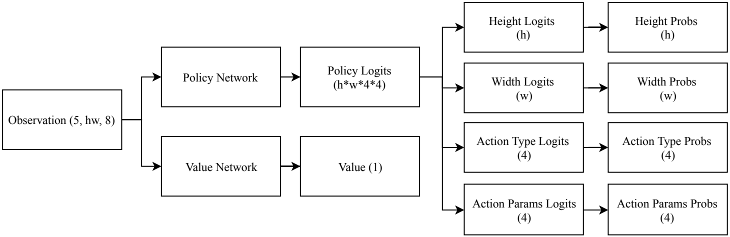
A2C

Algorithm 1 Advantage Actor Critic

```
Initialize policy function π with random weights θ
Initialize value function v with random weights θ′
for episode = 1, M do
    Reset the game and get s_1
    for t = 1, T do
        Sample action a_t from π(s_t)
        Execute action a_t and record reward r_t and state s_{t+1}
        If s_{t+1} is terminal, break
    end for
    for t = 1, T do
        Calculate the value v_t
        Calculate the advantage A = Σ_{i=1}^{T} γ^i r_t − v_t
        θ = θ + α∇_θ A log π(a_t|s_t) + π_θ(a_t|s_t) log π_θ(a_t|s_t)
        θ′ = θ′ + β∇_{θ′} A^2
    end for
end for
```

Specifics

How exactly do we input s_t to A2C and generate action a_t?



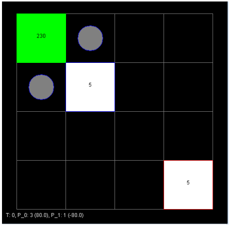
Specifics

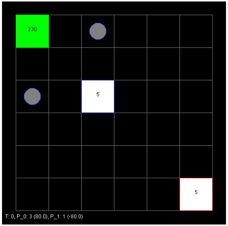
How exactly do we calculate $\log \pi_\theta(a_t|s_t)$ and $\pi_\theta(a_t|s_t) \log \pi_\theta(a_t|s_t)$?

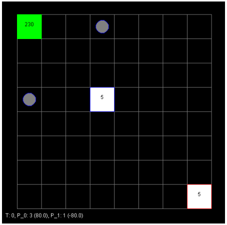
$$\begin{aligned}
\log \pi_\theta(a_t|s_t) &= \log \pi_\theta(a_t^{\text{x}}|s_t) \\
&+ \log \pi_\theta(a_t^{\text{y}}|s_t) \\
&+ \log \pi_\theta(a_t^{\text{action type}}|s_t) \\
&+ \log \pi_\theta(a_t^{\text{action param}}|s_t) \\
\pi_\theta(a_t|s_t) \log \pi_\theta(a_t|s_t) &= \pi_\theta(a_t^{\text{x}}|s_t) \log \pi_\theta(a_t^{\text{x}}|s_t) \\
&+ \pi_\theta(a_t^{\text{y}}|s_t) \log \pi_\theta(a_t^{\text{y}}|s_t) \\
&+ \pi_\theta(a_t^{\text{action type}}|s_t) \log \pi_\theta(a_t^{\text{action type}}|s_t) \\
&+ \pi_\theta(a_t^{\text{action param}}|s_t) \log \pi_\theta(a_t^{\text{action param}}|s_t)
\end{aligned}$$

Experimental Setup

- Task: harvest resources with different map sizes of 4×4, 6×6, and 8×8.
- It takes 10 timesteps to execute actions (move, harvest, return)
- When the agent harvested or returned the resources, it gets a reward of 10. Otherwise the reward is 0.

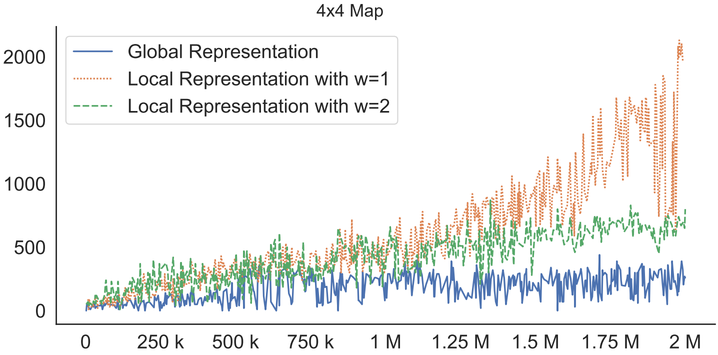






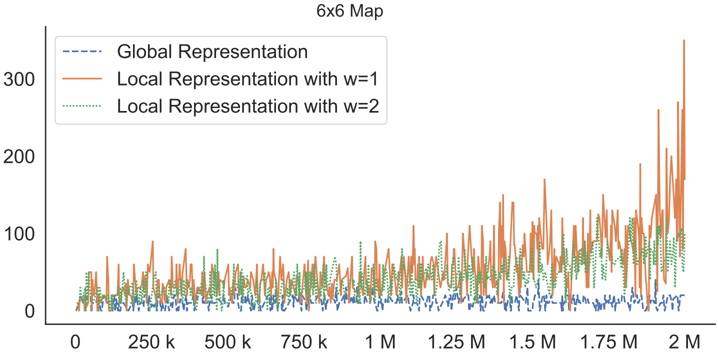
Episode Rewards

Episode rewards (y axis) as a function of training time steps (x-axis) for the 3 map sizes.



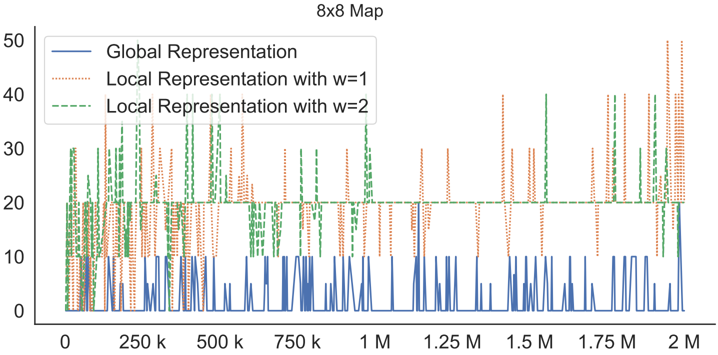
Episode Rewards

Episode rewards (y axis) as a function of training time steps (x-axis) for the 3 map sizes.



Episode Rewards

Episode rewards (y axis) as a function of training time steps (x-axis) for the 3 map sizes.



Metrics Evaluation

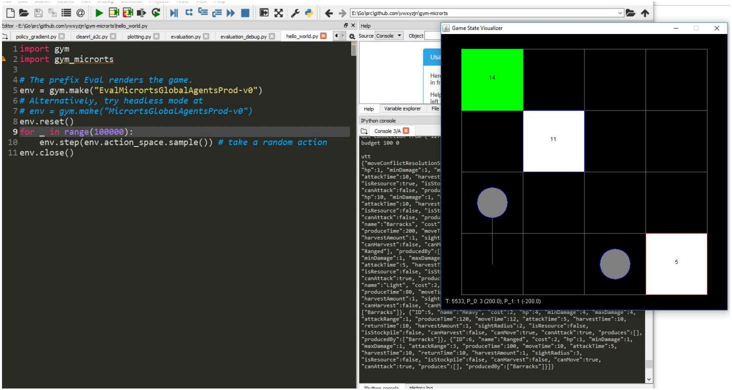
	map	$t_{\text{first harvest}}$	$t_{\text{first return}}$	r
RandomAI	4×4	51.33	142.67	11.87
Global	4×4	99.00	167.73	13.13
Local ($w = 1$)	4×4	29.87	172.47	67.20
Local ($w = 2$)	4×4	45.00	73.73	33.40
RandomAI	6×6	421.33	797.33	2.00
Global	6×6	533.33	1931.20	0.07
Local ($w = 1$)	6×6	59.20	567.40	3.53
Local ($w = 2$)	6×6	62.33	408.73	3.93
RandomAI	8×8	878.67	1480.67	0.87
Global	8×8	1464.53	-	0.00
Local ($w = 1$)	8×8	167.20	1844.20	0.20
Local ($w = 2$)	8×8	89.87	-	0.00

Visualization of Agents

https://youtu.be/--BoBOwnF0s

Gym environment

Our OpenAI Gym library is available at
`https://github.com/vwxyzjn/gym-microrts`



Conclusions

- We compared two intuitive representations
  - a *global* representation that feeds the RL agent the entire game state, and require the agent to learn to locate the unit and control it.
  - a *local* representation that feeds the RL agent the localized game state that is some distance away from the selected unit, and require the agent to learn to control it
- We show that local representation generally outperforms but the training of agents becomes more difficult in larger maps, where the exploration and sparse rewards become a huge problem.

Future Work

- Frame skipping
- Parallel processing
- Larger maps
- DQN-based algorithms
- Partial observability and LSTM
- Self-play

Our Code

- https://github.com/vwxyzjn/gym-microrts
- https://github.com/vwxyzjn/cleanrl
- https://github.com/vwxyzjn/microrts

Thank you. Questions?

References I

Lee, D., Tang, H., Zhang, J. O., Xu, H., Darrell, T., and Abbeel, P. (2018).
Modular architecture for starcraft ii with deep reinforcement learning.
In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Liu, R.-Z., Guo, H., Ji, X., Yu, Y., Xiao, Z., Wu, Y., Pang, Z.-J., and Lu, T. (2019).
Efficient reinforcement learning with a mind-game for full-length starcraft ii.
arXiv preprint arXiv:1903.00715.

References II

Marthi, B., Russell, S. J., Latham, D., and Guestrin, C. (2005).
Concurrent hierarchical reinforcement learning.
In *IJCAI*, pages 779–785.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. (2019).
The starcraft multi-agent challenge.
arXiv preprint arXiv:1902.04043.

References III

Sun, P., Sun, X., Han, L., Xiong, J., Wang, Q., Li, B., Zheng, Y., Liu, J., Liu, Y., Liu, H., et al. (2018).
Tstarbots: Defeating the cheating level builtin ai in starcraft ii in the full game.
arXiv preprint arXiv:1809.07193.

Tavares, A. R. and Chaimowicz, L. (2018).
Tabular reinforcement learning in real-time strategy games via options.
2018 IEEE Conference on Computational Intelligence and Games (CIG), pages 1–8.

References IV

Tian, Y., Gong, Q., Shang, W., Wu, Y., and Zitnick, C. L. (2017).
Elf: An extensive, lightweight and flexible research platform for real-time strategy games.
In *Advances in Neural Information Processing Systems*, pages 2659–2669.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al. (2017).
Starcraft ii: A new challenge for reinforcement learning.
arXiv preprint arXiv:1708.04782.