NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

School of Computer Science and Engineering
College of Engineering

Hierarchical Multi-Agent Reinforcement Learning with Options

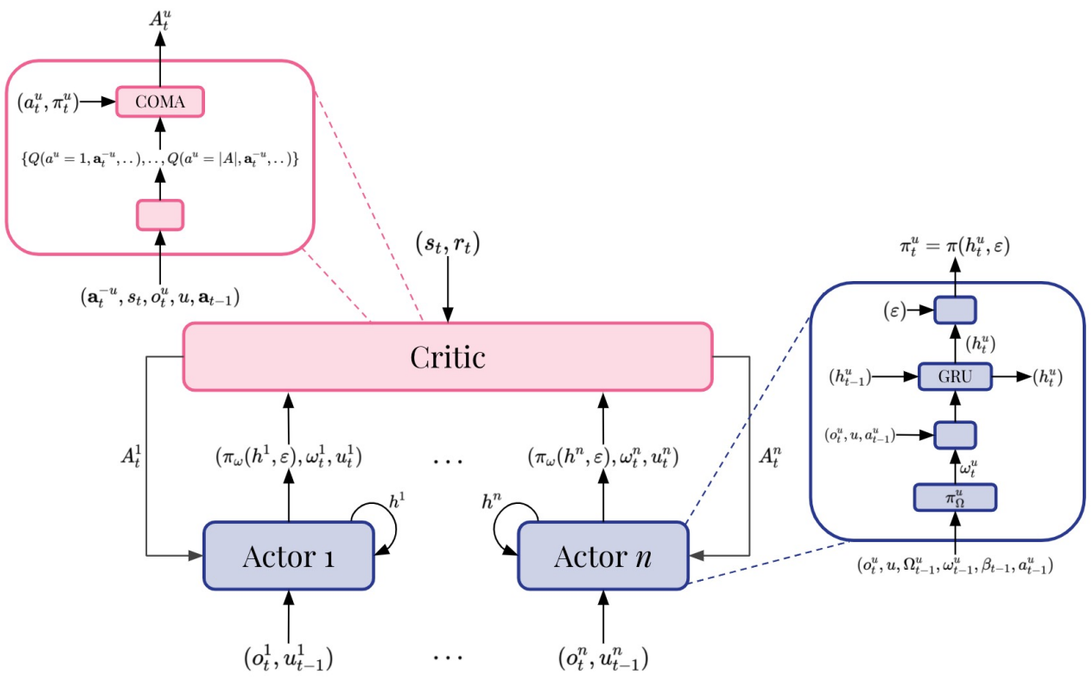
Student: Ang Wan Qi

Supervisor: Asst Prof Lana Obraztsova

Project Objectives

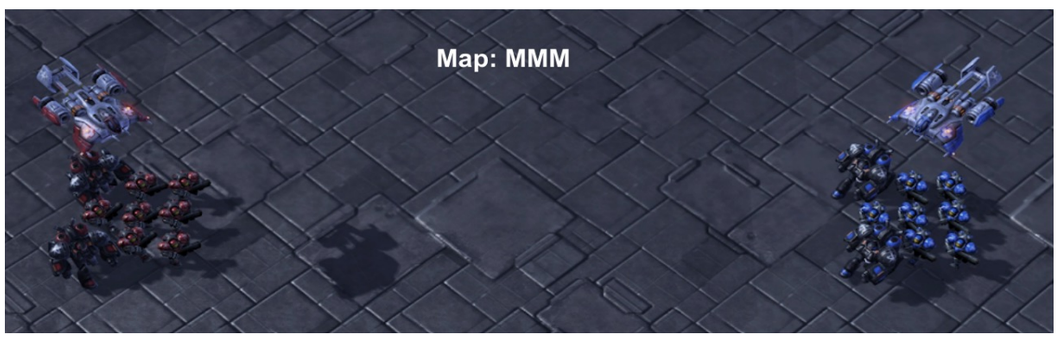
Cooperative multi-agent reinforcement learning (MARL) demands that multiple agents work together towards a common goal. Agents in these tasks often select and complete actions at different times and require asynchronous action execution. However, current methods have actions modeled as primitive operations and synchronized action execution. Hence, this project aims to explore hierarchical reinforcement learning (HRL) methods such as the options framework to achieve temporal abstraction of actions. A novel algorithm named COMA-OC is proposed to train the decentralized agents in StarCraft II's micromanagement scenarios against an enemy army.

COMA-OC Architecture

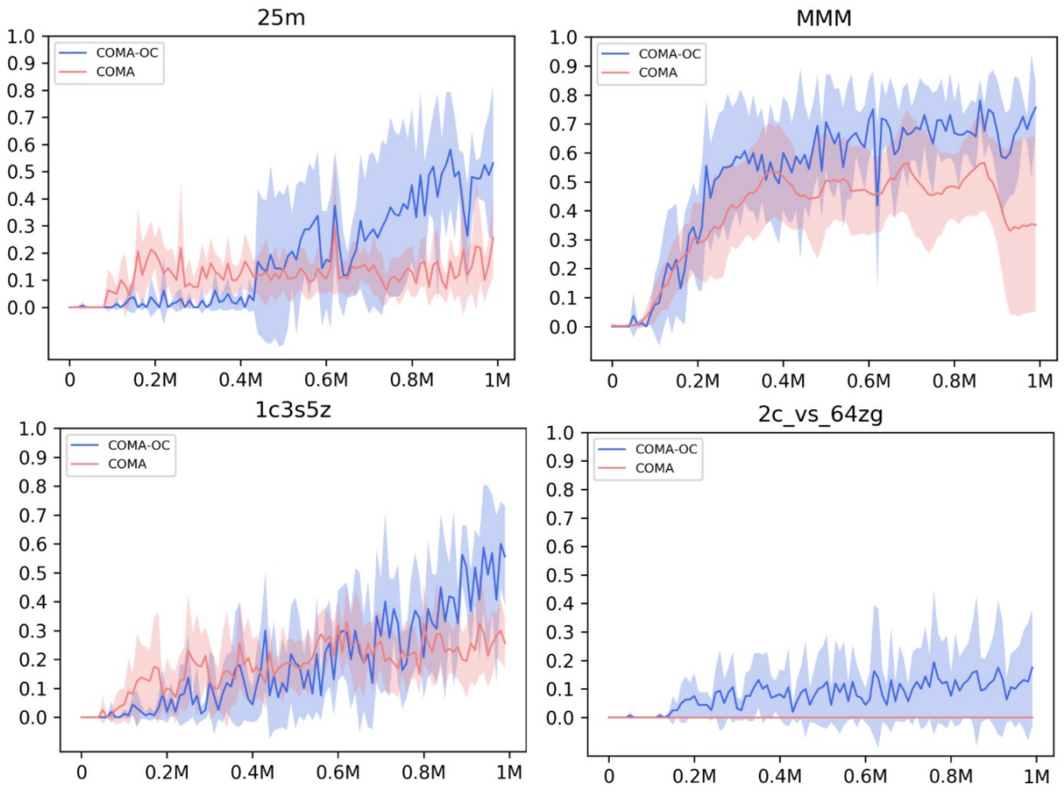


Evaluation Environment

COMA-OC is evaluated in StarCraft II where 2 armies battle each other in a scenario.



Results



COMA-OC Pseudocode

Algorithm 1 COMA-OC Algorithm

Initialise θ_1^c, $\hat{\theta}_1^c$, θ_1^π, θ_1^β
for each training episode e **do**
  Empty buffer
  for $e_c = 1$ **to** $\frac{\text{BatchSize}}{n}$ **do**
    s_1 = initial state, $t = 0$, $h_0^u = \mathbf{0}$ for each agent u
    Choose ω according to an ϵ-soft policy over options $\pi_\Omega(s_1)$
    while $s_t \neq$ terminal **and** $t < T$ **do**
      $t = t + 1$
      for each agent u **do**
        if ω terminates in s_t according to $\beta_{\omega,\theta^\beta}$ **then**
          Choose new ω according to ϵ-soft$\pi_\Omega(s_t)$
        $h_t^u = \text{Actor}\left(o_t^u, h_{t-1}^u, a_{t-1}^u, u, a, \theta_i\right)$
        Sample a_t^u from $\pi_{\omega,\theta^\pi}(h_t^u, \epsilon(e))$
      Get reward r_t and next state s_{t+1}
  Add episode to buffer
Collate episodes in buffer into single batch
for $t = 1$ **to** T **do** // processing all agents in parallel via single batch
  Batch unroll RNN using states, actions and rewards
  Calculate TD(λ) targets y_t^u using $\hat{\theta}_i^c$ // targets for actions
  Calculate update target g_t **for options:**
  if s is non-terminal **then**
    $g_t^u = r + \gamma(1 - \beta_{\omega,\theta^\beta}(s))Q_\Omega(s,\omega) + \gamma\beta_{\omega,\theta^\beta}(s)\max_{\bar{\omega}} Q_\Omega(s,\bar{\omega})$
for $t = T$ **down to** 1 **do**
  $\Delta Q_1^u(t) = y_t^u - Q_1\left(s_j^u, \mathbf{a}\right)$ // for actions
  $\Delta Q_2^u(t) = g_t^u - Q_2\left(s_j^u, \mathbf{a}\right)$ // for options
  $\Delta\theta^c = \nabla_{\theta^c}((\Delta Q_1^u(t))^2 + (\Delta Q_2^u(t))^2)$ // calculate critic gradient
  $\theta_{i+1}^c = \theta_i^c - \alpha\Delta\theta^c$ // update critic weights
  Every C steps reset $\hat{\theta}_i^c = \theta_i^c$
for $t = T$ **down to** 1 **do**
  $A^u(s_t^u, \mathbf{a}) = Q(s_t^u, \mathbf{a}) - \sum_a Q(s_t^u, a, \mathbf{a}^{-u})\pi(a|h_t^u)$ // calculate COMA
  $\Delta\theta^\pi = \Delta\theta^\pi + \nabla_{\theta^\pi}\log\pi(a|h_t^u)A^u(s_t^u, \mathbf{a})$ // accumulate actor gradients
  $\Delta\theta^\beta = \Delta\theta^\beta + \nabla_{\theta^\beta}\beta(s_t^u)A^u(s_t^u, \mathbf{a})$ // accumulate beta gradients
$\theta_{i+1}^\pi = \theta_i^\pi + \alpha\Delta\theta^\pi$ // update actor weights (pi loss)
$\theta_{i+1}^\beta = \theta_i^\beta + \alpha\Delta\theta^\beta$ // update actor weights (beta loss)

Conclusion and Future Work

Results showed that learning options leads to significant improvements in performance in various scenarios as compared to the baseline method. One possible improvement is to incorporate communication tools between agents to enhance cooperation.