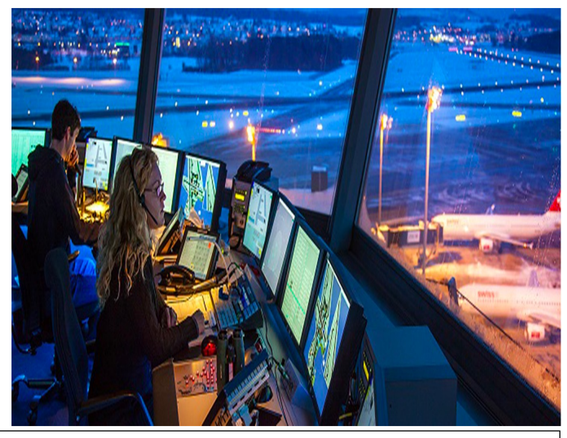
A Deep Ensemble Method for Multi-Agent Reinforcement Learning: A Case Study on Air Traffic Control

Supriyo Ghosh[1], Sean Laguna[1], Shiau Hong Lim[1], Laura Wynter[1] and Hasan Poonawala[2]

[1]IBM Research AI, Singapore; [2]Amazon Web Services (AWS), UK

Motivation

- **Air traffic control (ATC)**
  - Monitor current state of aircrafts and recommend real-time decisions.
  - Heavy traffic volume might lead to (human) operational errors.
  - Need to optimize a complex objective (minimize congestion, conflicts, arrival delay and fuel cost).
  - A **sequential** decision-making problem involving **multiple actors** influencing each other.



Our Contributions:

- Modelled ATC problem within a multi-agent reinforcement learning (MARL) framework.
- Solved the MARL problem with a model-based Kernel RL and a model-free deep RL methods.
- Proposed a **general-purpose novel** deep ensemble MARL method to combine the power of deep RL and kernel RL.
- Demonstrated the efficacy of ensemble MARL method on a real-world dataset consisting of ~1600 active aircrafts.

Deep Ensemble MARL

- Existing ensemble methods are either not feasible for model-based methods or unable to take multi-agent interactions in account.
- We train a separate deep neural network that efficiently learns to arbitrate between decisions of pre-trained kernel and deep MARL by considering multi-agent interactions in environment.

Ensemble Multi-agent RL:

Inputs: Kernel model $\tilde{K}$, PPO model $\tilde{\pi}(\tilde{\theta})$

1.Initialize ensemble policy to $\pi(\theta_0)$

2.For each episode k, run line 3-7

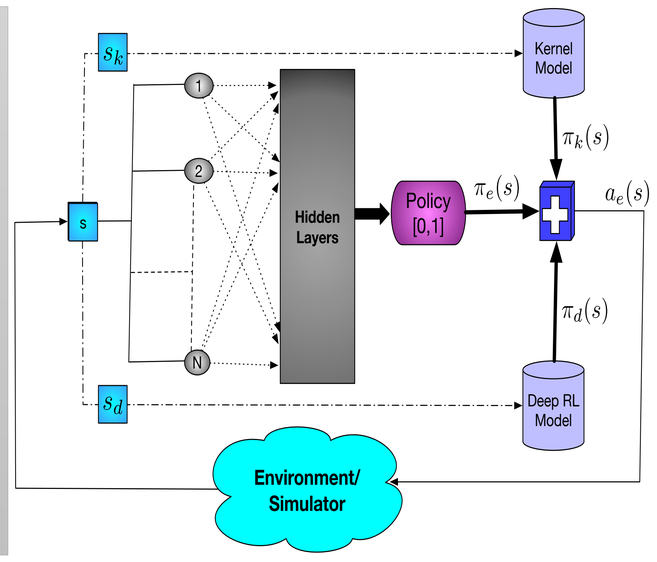
3.For each time t and agent i, sample ensemble action a_t^i using policy $\pi(\theta_k)$ for observation s_t^i

4.If a_t^i is 0 then get action $\tilde{a}_t^i$ from $\tilde{K}$, otherwise get $\tilde{a}_t^i$ using $\tilde{\pi}(\tilde{\theta})$

5.Execute joint action $\tilde{\boldsymbol{a}}_t = (\tilde{a}_t^1, \ldots, \tilde{a}_t^{N_t})$

6.Store transitions $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ in buffer D

7.For M rounds, update θ_k with minibatch of transitions from D to optimize the PPO objective



MARL Formulation for ATC

- Single Agent RL: $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$

States | Actions | Transition function | Reward function

Learn a policy π to maximize long term rewards:

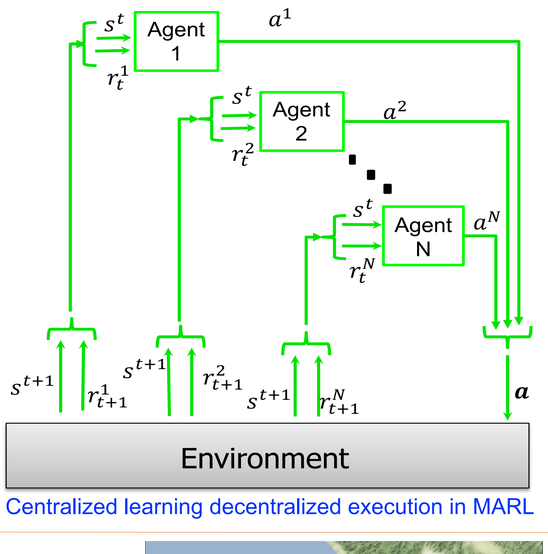
$$Q^*(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a \right]$$

- Multi-agent RL: $< \mathcal{S}, \underbrace{\mathcal{O}_1 \ldots \mathcal{O}_N}, \underbrace{\mathcal{A}_1 \ldots \mathcal{A}_N}, \mathcal{P}, \mathcal{R}, \gamma >$

Observation: $o_i : \mathcal{S} \rightarrow \mathcal{O}_i$

Transition: $\mathcal{P} : \mathcal{S} \times \mathcal{A}_1 \times \ldots \mathcal{A}_N \rightarrow \mathcal{S}$

Reward: $r_i : \mathcal{O}_i \times \mathcal{A}_i \rightarrow \mathbb{R}$



Centralized learning decentralized execution in MARL

MARL Components for ATC Problem:

- **State Space**
  - Local features: Aircraft's location, speed, direction, timeliness
  - Neighborhood features: N nearest aircrafts' relative velocity and relative direction
  - Extended feature: Coarse and fine grid image information
- **Action Space (deviate speed by δ)**

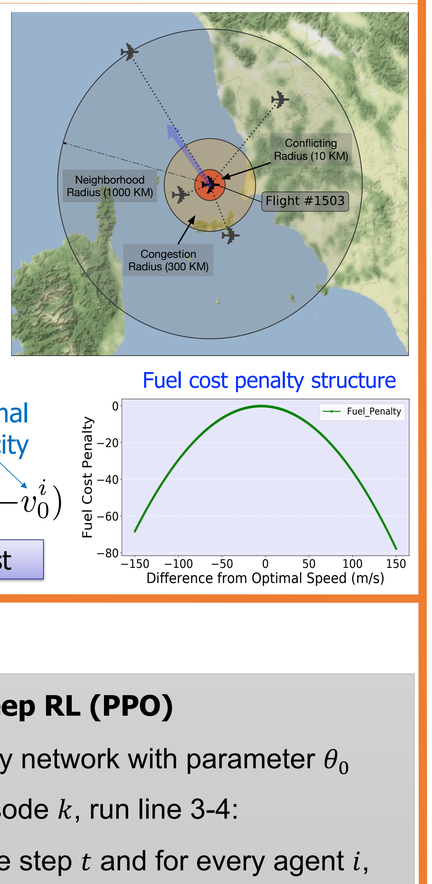
$$A_t = \{\underbrace{\max(v_{min}, (v_{t-1} - \delta))}, v_{t-1}, \underbrace{\min(v_{max}, (v_{t-1} + \delta))}\}$$

- **Reward Function**

Penalty Weights | Conflicting Radius | Congestion Radius | Expected Schedule | Optimal Velocity

$$r_t^i = \alpha \cdot I(o_t^i, R^s) + \beta \cdot I(o_t^i, R^c, N^c) + \gamma \cdot \max(0, \tilde{d}_t^i - d_t^i) + \delta \cdot F(v_t^i - v_0^i)$$

Conflicting Cost | Congestion Cost | Delay Penalty | Fuel Cost



Fuel cost penalty structure

Kernel and Deep MARL for ATC

Model Based Kernel RL

1. Inputs: $S^a = \{s_k^a, r_k^a, \hat{s}_k^a | k = 1, \ldots, n_a\} \forall a \in \mathcal{A}$
2. Generate m representative states with K-means clustering: $\bar{S} = \{\bar{s}_1, \ldots, \bar{s}_m\}$
3. Define Gaussian kernels $\kappa_\tau(\bar{s}, s)$, $\bar{\kappa}_{\bar{\tau}}(\hat{s}, \bar{s})$ using Euclidean distance from original to rep states:

$$\kappa_\tau(\bar{s}, s) = \phi\left(\frac{||\bar{s} - s||}{\tau}\right); \kappa_\tau(\bar{s}, s_i) = \frac{\kappa_\tau(\bar{s}, s_i)}{\sum_{j=1}^m \kappa_\tau(\bar{s}, s_j)}$$

4. Compute $D^a : d_{ij}^a = \bar{\kappa}_{\bar{\tau}}(\hat{s}_i^a, \bar{s}_j)$
5. Compute $K^a : k_{ij}^a = \kappa_\tau^a(\bar{s}_i, s_j^a)$
6. Compute transition probability: $P^a = K^a D^a$
7. Compute reward $r^a : r_i^a = \sum k_{ij}^a r_j^a$
8. Solve MDP $\{\bar{S}, \mathcal{A}, P^a, r^a, \gamma = .99\}$ & obtain Q^*

- **Advantages**
  - Performs well in the dense neighborhood of sample training data.
  - Strong theoretical bounds on training data.
- **Limitations**
  - Extrapolates poorly to unknown situations.

Model Free Deep RL (PPO)

1. Initialize policy network with parameter θ_0
2. For each episode k, run line 3-4:
3. For every time step t and for every agent i, collect transition samples $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ by simulating policy $\pi(\theta_k)$ and store them in a replay buffer D.
4. After every episode k, Update θ_k with minibatch of transitions from replay buffer D for M rounds to optimize the PPO objective:

$$\mathbb{E}_t\left[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)\right]$$

$r_t(\theta)$ is ratio between $\pi_\theta(a_t|s_t)$, $\pi_{\theta_{old}}(a_t|s_t)$

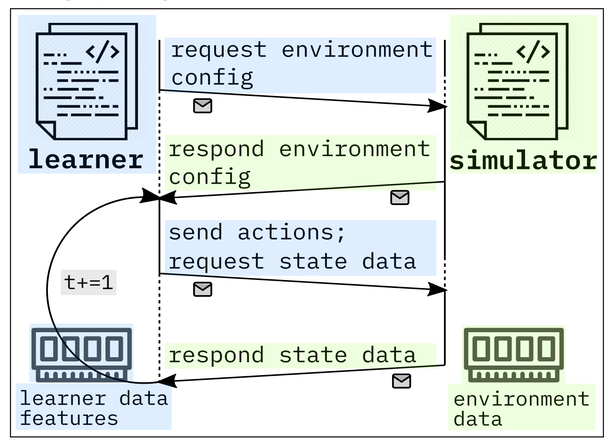
$A_t := R_t - V(s_t)$ is advantage function

- **Advantages**
  - Flexible and generalizes well to unknown situations.
  - Can deal with richer state space information.
- **Limitations**
  - Can be brittle even in dense neighborhood of training data.

Experimental Results

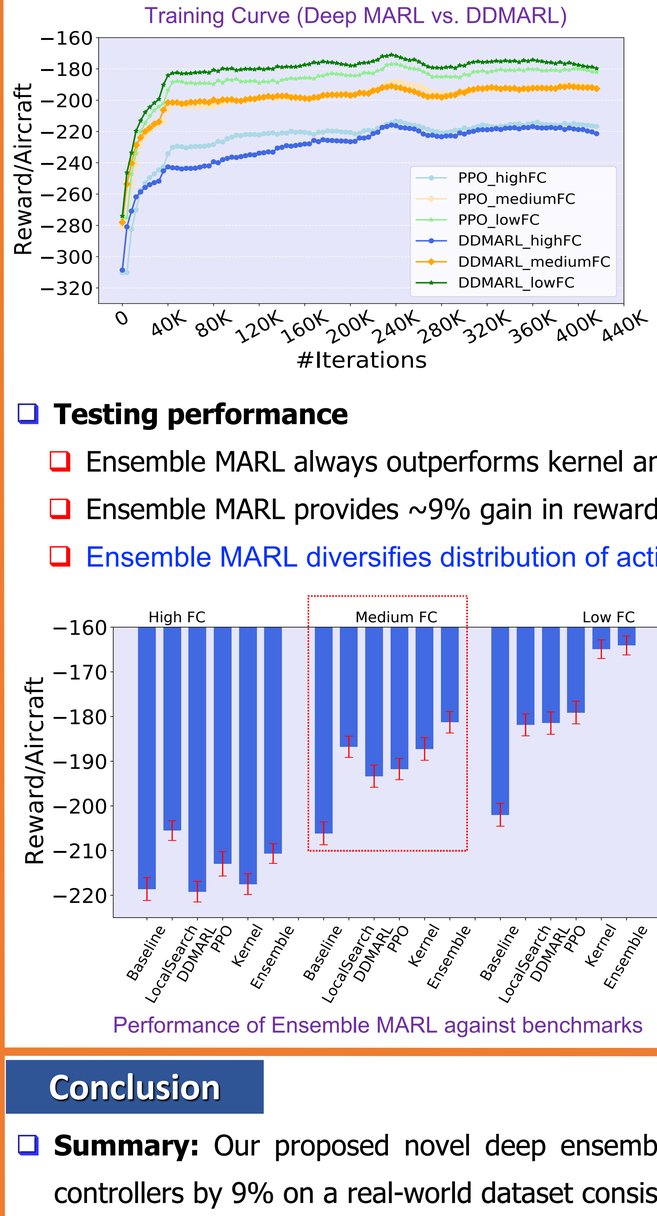
- **Datasets (from Southern Europe)**
  - 24-hours schedule for ~1600 active flights.
  - 300 training days and 30 testing days of schedule.
  - 3 fuel cost settings considered: low, medium (from Airbus) and high.
- **Air traffic simulator**
  - An open-source simulator developed by **Eurocontrol.**
  - We develop a message passing adapter between the simulator and our RL agent.

Massage-passing architecture between learner and simulator

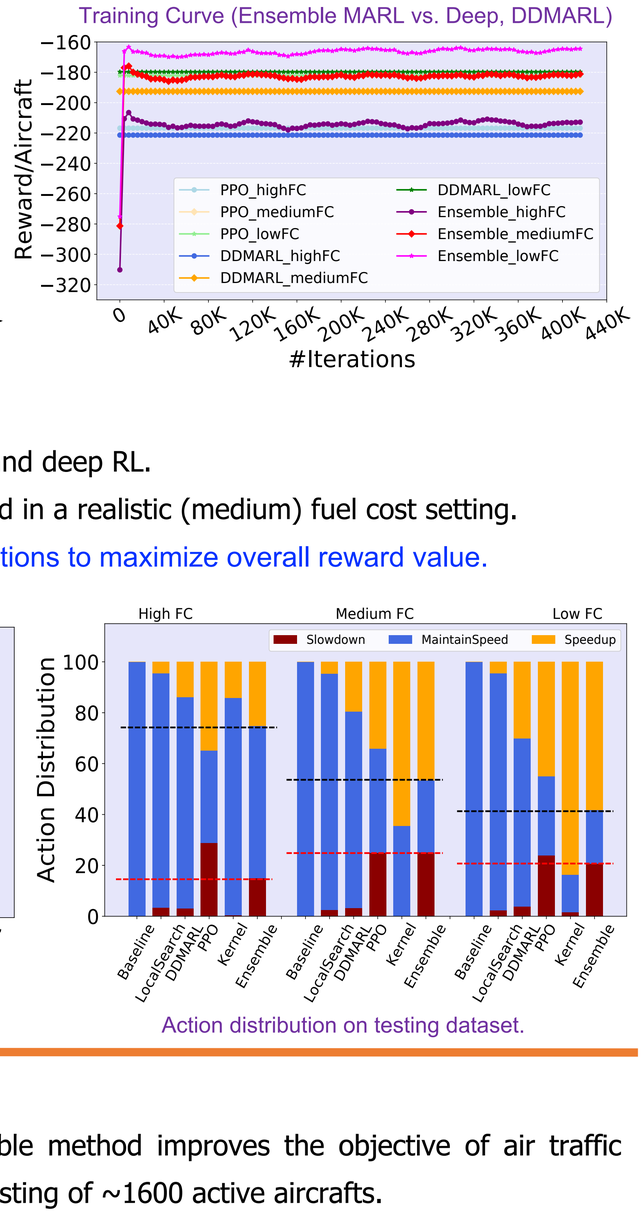


- **Benchmark Approaches**
  - Baseline: simulate default schedule (no penalty for fuel & delay).
  - Local search: Each aircraft chooses a myopic best action.
  - DDMARL (Brittain et. al., 2019): Only consider penalty for conflicting situations.
- **Training performance**
  - Training performance of our deep MARL is at par with DDMARL.
  - Our ensemble MARL has better training performance than both deep MARL and DDMARL across the board.

Training Curve (Deep MARL vs. DDMARL)



Training Curve (Ensemble MARL vs. Deep, DDMARL)



- **Testing performance**
  - Ensemble MARL always outperforms kernel and deep RL.
  - Ensemble MARL provides ~9% gain in reward in a realistic (medium) fuel cost setting.
  - Ensemble MARL diversifies distribution of actions to maximize overall reward value.

Performance of Ensemble MARL against benchmarks

Action distribution on testing dataset.

Conclusion

- **Summary:** Our proposed novel deep ensemble method improves the objective of air traffic controllers by 9% on a real-world dataset consisting of ~1600 active aircrafts.
- **Future Work:** (1) Extend action space to incorporate additional controls such as directional and altitude changes; (2) Extend state space to handle take-off and landing scenarios; (3) Extend ensemble MARL to combine power of multiple methods.