

Supplemental Materials to Submission: Hierarchical Multiagent Reinforcement Learning for Maritime Traffic Management

1. Synthetic Data Experimental Setup:

The settings of all instances are as follows. We use a graph with 30 edges, 6 source zones and 1 terminal zone as shown in figure 1, traffic enters from left and end at right side, the traffic from four last zones ends at one common terminal zone. For figure 4(c) experiment, we use different maps where we vary number of edges. Each vessel's arrival time at the starting edge is uniformly sampled from $[1, 20]$, each vessel consumes one unit of resource when traversing an edge, $t_{\min}^{zz'}$ and $t_{\max}^{zz'}$ are set to $[1, 30]$, for all experiments delay penalty $w_d = 1$ and horizon = 100. For each setting, we generate 10 instances and average values are reported. All of our experiments are performed on Intel(R) Xeon(R) Gold 6154, 72-Core processor, 3.0 GHz.

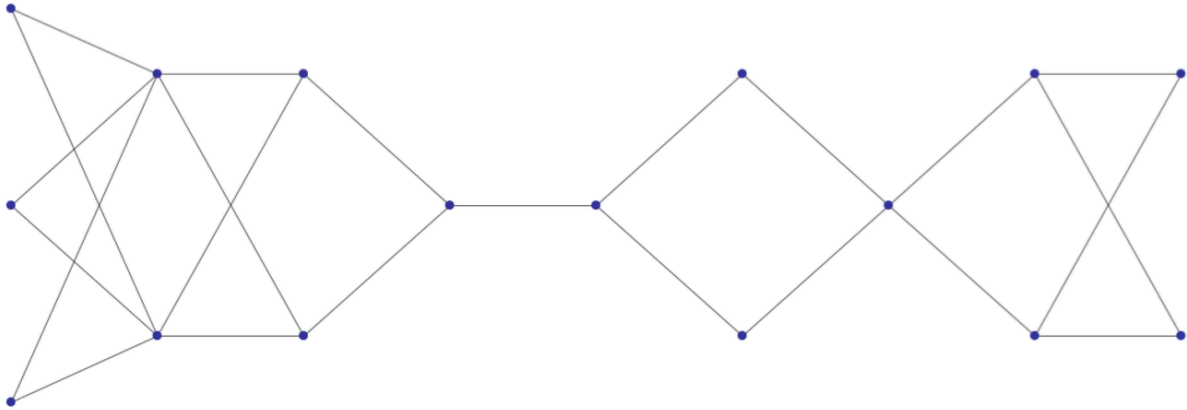


Figure 1: Graph of synthetic data experiments

1.1 Synthetic Graph Generation

For generating synthetic graph, we create series of bipartite like graph. First we select number of columns i.e the vertical line where blue dots(nodes) are shown is considered as a column. Once we select number of columns then we choose maximum number of nodes to be present in 1 column. Then for each column we randomly choose number of nodes between $[1, \text{max-nodes-per-column}]$. Then from each node in each column we make full connection with all others nodes of next column. In the above example number of columns = 9 and max-nodes-per-column = 3.

1.1 Implementation Details:

For all three approaches IMVF-PG, Vessel-PG, Meta-PG we use same neural network architecture, but Vessel-PG differ only in output layer. IMVF-PG and Meta-PG have two output heads one for each $\mu_{\theta zz'}$ and $\pi_{\nu zz'}$. Vessel-PG has only one output head for the deterministic policy and structure is similar to output head structure of $\pi_{\nu zz'}$. We use a one big neural network with each zone intersection $\langle z, z' \rangle$ as one sub-network which are segregated from one another. Input to the big network is $\mathbf{n}_t^{\text{tot}}$, then each zz' (sub-network) receives only the count information of $\mathbf{n}_t^{\text{tot}}(z)$ and neighboring zone count $\mathbf{n}_t^{\text{tot}}(z')$. For each subnetwork, we use 2 hidden layers, each layer with hidden nodes = 4, then from the last hidden layer we have two output heads one for each $\mu_{\theta zz'}$ and $\pi_{\nu zz'}$. For $\mu_{\theta zz'}$ we have output layer which gives probability distribution over meta actions with dimension equals to number of meta actions. For $\pi_{\nu zz'}$ outputs a vector $\langle \beta_{\omega}^{zz'} \rangle_{\omega}$. For each hidden layer, we use \tanh activation and for the output layer $\mu_{\theta zz'}$ head we use softmax and for output head sigmoid activation is used on each unit so that we get the value between $[0, 1]$. Layer-norm [1] is applied before each hidden layer and output layer. We use Adam optimizer [2] with learning rate 1e-3 and entropy penalty 1e-3 is used. All of our model are implemented on pytorch [3].

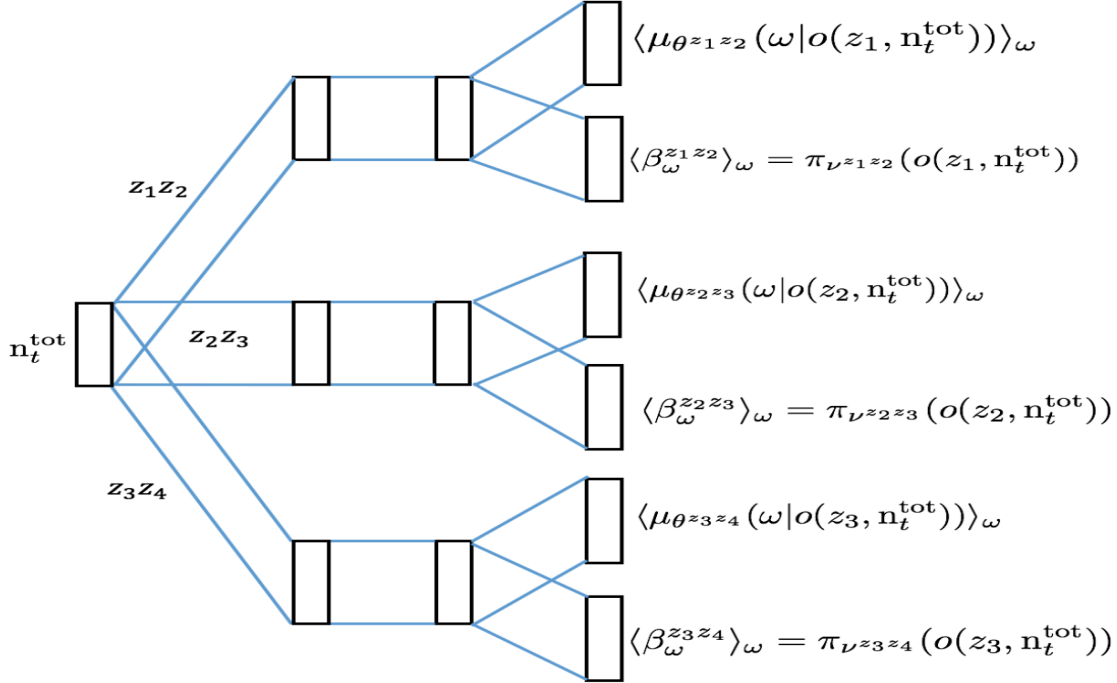


Figure 2: Neural Network Architecture

2 Collective Graphical Model :

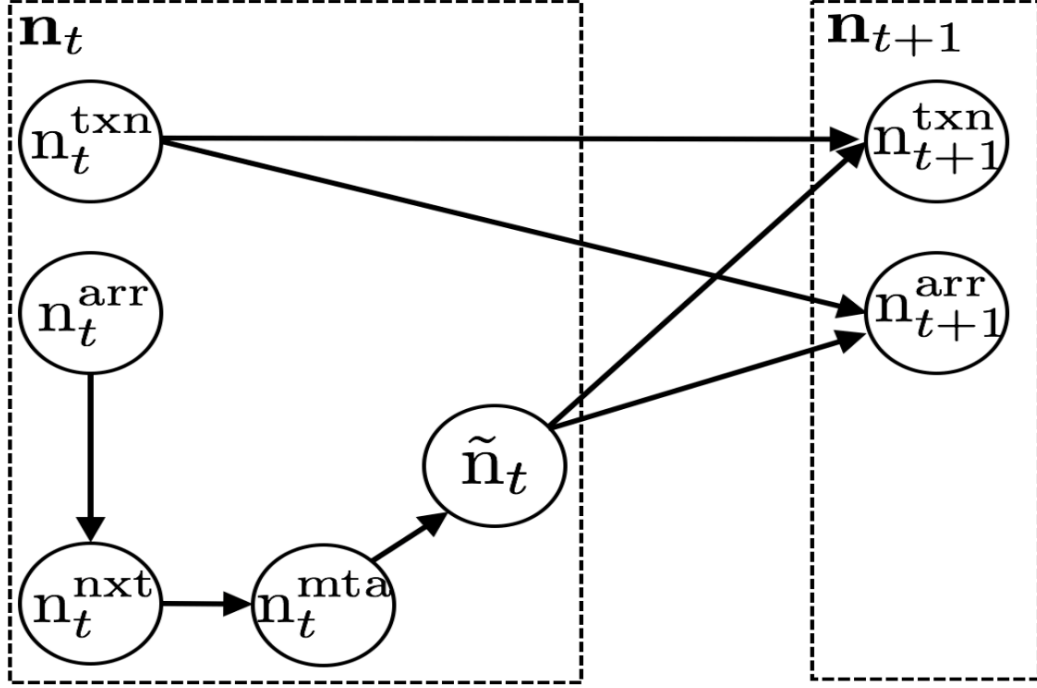


Figure 3: Collective Graphical Model

2. Sampling Process :

Here we describe our stochastic process for generating count $\mathbf{n}_{t+1} = (n_{t+1}^{\text{arr}}, n_{t+1}^{\text{nxt}}, n_{t+1}^{\text{mta}}, \tilde{n}_{t+1}, n_{t+1}^{\text{txn}})$ given \mathbf{n}_t .

$n_{t+1}^{\text{arr}}(z')$: This count represents total number of vessels that arrived at zone z' at time $t+1$. It is computed by sum of vessels that were in transit in previous zone z at time t reached z' at $t+1$ and newly arrived vessels in zone z at time t reaching z' at time $t+1$, note that, there may be many vessels reaching z' at $t+1$ traveling with varying behavior ω .

$$n_{t+1}^{\text{arr}}(z') = \sum_z \left[n_t^{\text{txn}}(z, z', \tau = 1) + \sum_{\omega} \tilde{n}_t(z, z', \omega, \tau = 1) \right] \forall z' \quad (1)$$

$n_{t+1}^{\text{nxt}}(z, \cdot)$: This count represents total number of vessels that newly arrived at zone z at $t + 1$ and moving to next neighboring zone z' . It can be generated from a multinomial distribution with parameters $n_{t+1}^{\text{arr}}(z)$ and $p_{z'} = \alpha(z'|z)\forall z'$

$$n_{t+1}^{\text{nxt}}(z, \cdot) | n_{t+1}^{\text{arr}}(z) \sim \text{Mul}(n_{t+1}^{\text{arr}}(z), p_{z'} \forall z')$$

$n_{t+1}^{\text{mta}}(z, z', \cdot)$: Next we generate behavior count from $n_{t+1}^{\text{nxt}}(z, z')$ i.e newly arrived vessels at z moving to z' and choosing a behavior ω from policy $\omega \sim \mu_{\theta}^{zz'}(\omega | o(z, n_{t+1}^{\text{tot}}))$. We can generate the count from a multinomial distribution with parameters $n_{t+1}^{\text{nxt}}(z, z')$ and $p_{\omega} = \mu_{\theta}^{zz'}(\omega | o(z, n_{t+1}^{\text{tot}})), \forall \omega$

$$n_{t+1}^{\text{mta}}(z, z', \cdot) | n_{t+1}^{\text{nxt}}(z, z') \sim \text{Mul}(n_{t+1}^{\text{nxt}}(z, z'), p_{\omega} \forall \omega)$$

$\tilde{n}_{t+1}(z, z', \omega, \cdot)$: Now, we generate arrival time count from $n_{t+1}^{\text{mta}}(z, z', \omega)$. Since all newly arrived vessels at z moving to z' with behavior ω follow a travel time distribution p^{nav} , we can generate the count from a multinomial distribution with parameters $n_{t+1}^{\text{mta}}(z, z', \omega)$ and $p_{\tau} = p^{\text{nav}}(\tau | z, z'; \beta_{\omega}^{zz'} = \pi_{\nu^{zz}}(o(z, n_{t+1}^{\text{tot}}))) \forall \tau$

$$\tilde{n}_{t+1}(z, z', \omega, \cdot) | n_{t+1}^{\text{mta}}(z, z', \omega) \sim \text{Mul}(n_{t+1}^{\text{mta}}(z, z', \omega), p_{\tau} \forall \tau)$$

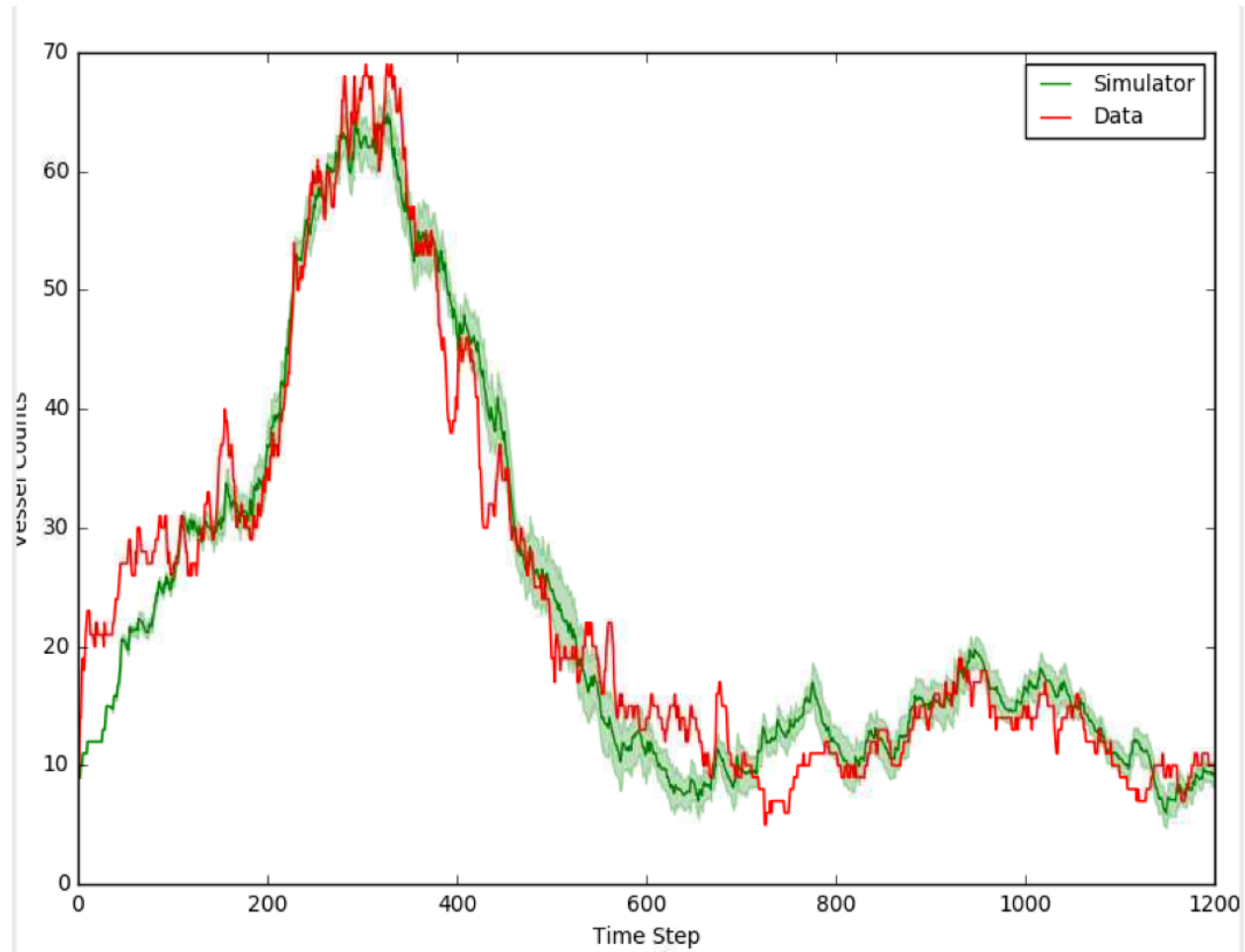
where, $\tau = t_{\min}^{zz'} + \tilde{\Delta}$, $\tilde{\Delta} \in [0, t_{\max}^{zz'} - t_{\min}^{zz'}]$

Now, we compute transit count $n_{t+1}^{\text{txn}}(z, z', \tau)$, vessels at zone z moving to next zone z' that have not reached at time $t + 1$ i.e $\tau > 1$

$$n_{t+1}^{\text{txn}}(z, z', \tau) = n_t^{\text{txn}}(z, z', \tau) + \sum_{\omega} \tilde{n}_t(z, z', \omega, \tau) \quad \forall z, z', \tau > 1 \quad (1)$$

With this generative model for count, we can compute the policy gradient by sampling at the count level $\mathbf{n}_{1:H}$ instead of sampling individual vessel trajectories.

3. Simulator Accuracy :



References

- [1] Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.
- [2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [3] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.