Supplementary material for "Privacy-Aware Deep RL for Sequential Coalition Formation Decisions under Uncertainty"

I. Coalition Formation Protocol

Algorithm 1 A DRL-enabling coalition formation protocol

Require: Initialized set of agents, A

Initialize the list of Tasks
for each episode **do**
  Reset the environment
  while no terminal condition has been met **do**
    An agent i is randomly selected as the proposer
    while proposer has made $< p$ proposals **do**
      Proposer i uses DRL to make a proposal
      Each responder agent uses DRL to respond
      if proposal is accepted **then**
        Coalition C is formed to complete task $\mathcal{T}_C$.
        The reward/penalty R for completing $\mathcal{T}_C$ is distributed equally among the participants.
      else
        Another proposal is made from proposer i.
      end if
    end while
  end while
end for

II. Proof of Theorem 1

Here, we provide a formal proof that agents of different social groups do not share sensitive information with the use of the PAMH-DQN. However, the social groups can communicate the *shared part* of the PAHM-DQN architecture. In this proof, we show that the agents cannot retrieve the sensitive information only by knowing the weights of the shared layers. Recall that, by Definition 1, sensitive information is defined as coalition task pairs $\mathcal{P}$ and the corresponding reward values R, $d^* = \{\mathcal{P}, R(\mathbf{t_C}, \mathcal{T}_C)\}$. For ease of notation, we will use $d^*[0]$ to denote a coalition task pair, and $d^*[1]$ to denote the corresponding reward.

Theorem 1. *PAMH-DQN does not compromise privacy among social groups.*

Proof. **Forward Pass:** PAMH-DQN consists of a number of shared layers C and K heads; each head is utilized only by the corresponding social group. During the forward pass of the network, the neuron weights (both of the shared layers and the head-specific ones) are not updated and hence, sensitive information does not leak. In addition, during the forward pass between the shared network and the heads, the information is multiplied by the mask. In this way, the heads of different social groups do not have access to sensitive information of other groups and thus cannot perform any kind of calculations that use sensitive information of others.

Let $y = d^*[1] + \gamma max_a Q(s', a)$ be the target of the NN, and h_k^l is the output of the l layer of the k^{th} head. The output of a shared layer is defined as h_{shared}^l. Finally, we define L as the last layer of each head and M as the last layer of the shared network. Hence, $h_k^L, \forall k$ is the output of the last layer of each head and h_{shared}^M is the last layer of the shared network. We also define as $I_o^{(k)}$ the input to each head for that specific observation o; this input is derived from the last layer of the shared network. That is, $I^{(k)}$ can be written as:

$$I^{(k)} = m_o^k \cdot h_{\text{shared}}^M, \forall k$$

where m_o^k is the mask for observation o, i.e. a K dimension one-hot vector that only the k^{th} bit is 1. Hence, during the forward pass, the information from that shared layer is only passed to the k^{th} head. Note that k may also represent an array, as a single experience can influence multiple heads/social groups. However, for notational simplicity, we will treat k as a scalar throughout the remainder of the proof[1].

Backpropagation: During the backpropagation, each social group's head is trained on different data. That is, the gradients g_o^k calculated in the last layer of each head k are:

$$g_o^k = m_o^k \left(y - h_k^l\right) \nabla_{w_k} h_k^l.$$

Thus, during the backpropagation of sensitive data belonging to social group k, the gradient of all heads $i \neq k$ will be zero. This can be also seen from the formula for upgrading the weights of another head i:

$$\frac{\partial L}{\partial W_i^{(l)}} = \frac{\partial L}{\partial h_i^{(L)}} = \frac{\partial h_i^{(L)}}{\partial h_i^{(l)}} = \frac{\partial h_i^{(l)}}{\partial W_i^{(l)}} = 0, \forall i \neq k$$

Hence, in the case of backpropagating gradients generated by data from a social group k, only the weights of k^{th}head are updated; the other heads are not affected by that data in no means possible.

Updating the shared layers: We now consider the update

[1]In our implementation, during the forward pass, we directly connect the output of the shared layer to the input of the appropriate head k. Hence, we do not perform any computations in the non-appropriate heads

of the shared layers. During the backpropagation step, the gradient for a *shared layer* l, is calculated as (chain rule):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l_{\text{shared}}} = \frac{\partial \mathcal{L}}{\partial h^L_k} \cdot \frac{\partial h^L_k}{\partial \mathbf{h}^l_{\text{shared}}} \cdot \frac{\partial \mathbf{h}^l_{\text{shared}}}{\partial \mathbf{W}^l_{\text{shared}}}$$

where $\mathbf{W}^l_{\text{shared}}$ are the weights of the shared layer l and $\mathcal{L}$ is the Huber loss calculated as:

$$\mathcal{L}(y, h^L_k) = \begin{cases} \frac{1}{2}(y - h^L_k)^2 & \text{if } |y - h^L_k| \leq \delta, \\ \delta\left(|y - h^L_k| - \frac{\delta}{2}\right) & \text{otherwise,} \end{cases}$$

The Huber loss $\mathcal{L}(y, h^L_k)$ yields gradients:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial h^L_k} &= \begin{cases} h^L_k - y \\ \delta \cdot \operatorname{sgn}(y - h^L_k) \end{cases} \\ &= \begin{cases} Q^L_l(s,a) - d^*[1] - \gamma \max_{a'} Q(s', a'), & \text{if } |y - h^L_k| \leq \delta \\ \delta \cdot \operatorname{sgn}(d^*[1] + \gamma \max_{a'} Q(s', a') - Q^L_l(s,a)), & \text{otherwise} \end{cases} \end{aligned}$$

where the $sgn(x) = -[x < 0] + [x > 0]$. Hence sgn is non-bijective, since different sgn inputs can be paired to the same sgn outputs—for instance, $sgn(+2) = sgn(+3) = +1$ and $sgn(-1) = sgn(-15) = -1$. *Notice this is enough to guarantee that* $\frac{\partial \mathcal{L}}{\partial h^L_k}$ *and therefore gradients* $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l}$ *are non-bijective.*

In addition, for interest, we note that the reward function (and thus the Q-function) is non-bijective since we guarantee in our implementation that our characteristic function v is both stochastic and non-injective. For example, assume $v(\mathcal{T}_C, A, B) = v(\mathcal{T}_{C'}, E, F) \sim \mathcal{N}(30, 20)$ where A,B,E,F are distinct types of agents participating in (two distinct) coalitions tackling tasks $\mathcal{T}_C$ and $\mathcal{T}_{C'}$; and $\mathcal{N}$ is a gaussian distribution with $\mu = 30$ and $\sigma^2 = 20$. Since it is possible that $v(\mathcal{T}_C, A, B) = v(\mathcal{T}_{C'}, E, F) = c$, we cannot infer which coalition-task pair $d^*[0]$ produces a reward equal to c.

Since gradients $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l}$ are non-bijective and therefore unable to uniquely identify y or h^L_k, no sensitive information can be inferred from the weight updates (and hence the layers) of the shared network. $\square$