Supplemental Material for "Exploiting Behavior Sequences for Multi-agent Interactive Decision Making through Variational Autoencoder"

Appendix

In this Supplementary Materials, we detail the four operators in reconstructing an incomplete policy tree, and the operators in encoding and decoding an incomplete policy tree. All implementations are included in our I-DID toolkit [1] .

A: [The Four Operators in Reconstructing an Incomplete Policy Tree] We present the pseudo-code to implement the four operators in reconstructing an incomplete policy tree. The operators are detailed in Section "Reconstructing an Incomplete Policy Tree" and outlined in Algorithm "VAE-Enabled Behaviors".As depicted in Alg. 1, the key steps of each operator are as follows:

- **Split Operator:** The *split* operator $\mathcal{S}$ divides a sequence into multiple sub-sequences based on a fixed length, and stores them along with their probabilities in the form of a set.
  - Initially, an empty set H^T is initialized to store policy paths and their associated probabilities (line 4).
  - The given action-observation sequence is then segmented into multiple sub-sequences of fixed length (line 5).
  - These sub-sequences, along with their probabilities, are stored in the set H^T (lines 6-9), providing a structured representation of policy paths.

[1] https://github.com/lamingic/VAE-IDID

- **Union Operator:** The *union* operator $\mathcal{U}$ combines the elements of the given set of policy paths into a set of sets of policy trees based on the grouping of their root node actions and observation pairs.
    - Policy path subsets $H^T_{a\mathbf{o}}$ are defined and their probabilities computed, where each subset begins with a specific action a and observation sequence $\mathbf{o}$ (lines 15-16).
    - For every possible observation sequence $\mathbf{o}$ that follows action a, the corresponding subsets $H^T_{a\mathbf{o}}$ are collected (lines 14-16).
    - Sets $\mathrm{H}^{\mathrm{T}}_{\mathrm{a}}$ are then constructed, encompassing all policy paths that start with a particular action a (line 17).
    - The overall probability $\mathbb{P}(\mathrm{H}^{\mathrm{T}}_{\mathrm{a}})$ of each $\mathrm{H}^{\mathrm{T}}_{\mathrm{a}}$ is determined by summing the probabilities of its constituent subsets $H^T_{a\mathbf{o}}$ (line 18).
    - Combining the sets $\mathrm{H}^{\mathrm{T}}_{\mathrm{a}}$ and their probabilities for all actions $a \in A$ yields the comprehensive set of policy paths H^{T} (lines 13-19).
    - The comprehensive set H^{T} represents all potential policy trees of depth T that can be derived from the original interaction sequence h^L (line 20).
- **Roulette Operator:** The *roulette* operator $\mathcal{R}$ randomly selects an element from a given set.
    - Given a set A of elements (a, p) where p is the probability associated with a, a random number p_r between 0 and 1 is generated (line 24).
    - A running sum p_c is maintained to track the cumulative probabilities encountered while iterating through the set A (line 25).
    - As soon as p_c exceeds or equals p_r, the corresponding element a is returned, effectively selecting an element based on its associated probability (lines 26-31).
- **Graphing Operator:**The *graphing* operator $\mathcal{G}$ converts the set of paths from the policy tree into a tree structure within a graph model.
    - An edge set E is constructed to store the sub-paths within the policy tree (lines 35-41).

  - Duplicates are removed from E, ensuring that edges with the same time slice, node value (action), and edge weight (observation) are not represented multiple times (line 42).
  - Unique nodes from the deduplicated edge set E are then extracted and compiled into a node set V (line 44).
  - Finally, a directed graph $G(E, V)$ is generated and plotted, representing the policy tree constructed from the edge and node sets (line 45).

B: [The Operators in encoding and decoding an Incomplete Policy Tree] The ZZOH encoder and decoder operators are described in Section "Zig-Zag One-Hot encoding & decoding" and outlined in Algorithm "VAE-Enabled Behaviors".The operators are used to perform a zigzag transformation on the actions of a given policy tree, encoding them into a binary format using one-hot encoding to form a column vector. Conversely, it can also decode a given column vector from one-hot encoding back into actions, subsequently reconstructing the policy tree. As depicted in Alg. 2, the key steps of each operator are as follows:

- **ZZOH Encoder Operator:**
  - Initially, we extract the action sequence from the given policy tree $\mathcal{H}^T$ (lines 3-12).
  - Subsequently, for each action in the sequence, we generate a corresponding one-hot binary code representation p (lines 13-16). This one-hot encoding is designed specifically for policy trees, utilizing the Zig-Zag encoding technique.
- **ZZOH Decoder Operator:**
  - Initially, given a one-hot binary code representation $\mathbf{x}$, we decode it back into an action sequence p (lines 20-26). The decoding process reverses the Zig-Zag encoding, accurately reconstructing the original action sequence.
  - Subsequently, using the decoded action sequence p, we reconstruct the policy tree $\mathcal{H}^T$ (lines 27-40). This reconstruction ensures that the generated tree closely matches the original tree, capturing its structure and behavior.

Algorithm 1: The Four Operators in Reconstructing an Incomplete Policy Tree

```
Function $\mathcal{S}$( $h^L$, T)   ◁ split operator: $H^T \leftarrow \mathcal{S}_T h^L$:
    $a^1o^1, a^2o^2 ... a^t, o^t, ... a^L o^L \leftarrow h^L$
    $H^T \leftarrow \emptyset$
    $\{h^T\} \leftarrow \{a^{(l-1)T+1} o^{(l-1)T+1}, \ldots, a^{lT} o^{lT}\}_{l=1}^{\lfloor \frac{L}{T} \rfloor}$
    for $h^{\mathrm{T}} \in \{h^{\mathrm{T}}\}$ do
        $\mathbb{P}(h^T) \leftarrow \#(h_a^T)T/L$  ◁ $\#(h_a^{\mathrm{T}})$ indicates the number of times $h^{\mathrm{T}}$
          appears in the sequence count the times of $h^L$
        $H^T \leftarrow H^T \bigcup (h^T, \mathbb{P}_{h^T})$
    end
return $H^T$
Function $\mathcal{U}$( $H^{\mathrm{T}}$)  ◁ union operator:$\mathrm{H}^T \leftarrow \mathcal{U} H^T$:
    for $a \in A$ do
        for $\boldsymbol{o} \in \Omega^{\mathrm{T}-1}$ do
            $H_{\mathbf{ao}}^T \leftarrow \bigcup_{(h_{\mathbf{ao}}^T, \mathbb{P}(h_{\mathbf{ao}}^T)) \in H^T} (h_{\mathbf{ao}}^T, \mathbb{P}(h_{\mathbf{ao}}^T))$
              $\mathbb{P}(H_{\mathbf{ao}}^T) \leftarrow \sum_{(h_{\mathbf{ao}}^T, \mathbb{P}(h_{\mathbf{ao}}^T)) \in H^T} \mathbb{P}(h_{\mathbf{ao}}^T)$
        end
        $\mathrm{H}_{\mathrm{a}}^T \leftarrow \bigcup_{\mathbf{o} \in \Omega^{T-1}} (H_{\mathbf{ao}}^T, \mathbb{P}(H_{\mathbf{ao}}^T))$
        $\mathbb{P}(\mathrm{H}_{\mathrm{a}}^T) \leftarrow \sum_{\mathbf{o} \in \Omega^{T-1}} \mathbb{P}(H_{\mathbf{ao}}^T)$
    end
    $\mathrm{H}^T \leftarrow \bigcup_{\mathrm{a} \in \mathrm{A}} (\mathrm{H}_{\mathrm{a}}^T, \mathbb{P}(\mathrm{H}_{\mathrm{a}}^T))$
return $\mathrm{H}^{\boldsymbol{T}}$
Function $\mathcal{R}$( $A$)  ◁ roulette operator:$a \leftarrow \mathcal{R}A$:
    $p_r \leftarrow random()$, $p_c \leftarrow 0$
    for $(a, p) \in A$ do
        $p_c \leftarrow p_c + p$
        if $p_r <= p_c$ then
            return $a$
        end
    end
return $a$
Function $\mathcal{G}$( $\mathcal{H}_a^T$)  ◁ graphing operator:$Tree \leftarrow \mathcal{G}\mathcal{H}_a^T$:
    $E \leftarrow \emptyset$
    for $(\mathrm{h}^{\mathrm{T}}, \mathbb{P}(\mathrm{h}^{\mathrm{T}})) \in \mathcal{H}^T$ do
        $a^1o^1, a^2o^2 ... a^t, o^t, ... a^T o^T \leftarrow h^T$
        for $t \in \{1, 2, \cdots T-1\}$ do
            $E \leftarrow E \bigcup (o^t : a^t, a^{t+1})$
        end
    end
    $E \leftarrow Deduplicate(E)$  ◁ deduplicate the edge with same time slice,
      same node value(action) and edge weight (observation)
    $V \leftarrow Node(E))$
    $Tree \leftarrow G(E, V)$
return $Tree$
```

Algorithm 2: ZZOH Encoder & Decoder Operators

```
$\triangleleft$ *ZZOH encoder operator*
**Function** $\mathcal{Z}( \ \mathcal{H}^T)$  $\triangleleft$ $\mathbf{x} \leftarrow \mathcal{Z}\mathcal{H}^T$**:**
    $\triangleleft$ *generate action sequence of tree* $\mathcal{H}^T$
    $\mathbf{x} \in \mathbb{R}^{(|A_j|+1)\frac{|\Omega|^T-1}{|\Omega|-1}}$
    $p \leftarrow \{a_l | a_l \leftarrow 0, \forall l \in \{1, 2, \cdots \frac{|\Omega|^T-1}{|\Omega|-1}\}\}$
    **for** $(h^T, i) \in \mathcal{H}^T$ **do**
        $a^1 o^1, a^2 o^2 ... a^t, o^t, ... a^T o^T \leftarrow h^T$
        **for** $t \in \{1, 2, \cdots T\}$ **do**
            $l \leftarrow \frac{|\Omega|^{(t-1)}-1}{|\Omega|-1} + \lfloor \frac{i-1}{|\Omega|^{(t-1)}} \rfloor + 1$
            $p[l] \leftarrow a^t$
        **end**
    **end**
    $\triangleleft$ *generate one-hot binary code of action sequence* $p$
    **for** $l \in \{1, 2, \cdots \frac{|\Omega|^T-1}{|\Omega|-1}\}$ **do**
        $\mathbf{x}[(l-1)(|A_j|+1)+1 : (|A_j|+1)l] \leftarrow \tilde{A}_j^c[p[l]]$
    **end**
**return** $\mathbf{x}$
$\triangleleft$ *ZZOH decoder operator*
**Function** $\mathcal{Z}( \ \boldsymbol{x})$  $\triangleleft$ $\mathcal{H}^T \leftarrow \mathcal{Z}\mathbf{x}$**:**
    $\triangleleft$ *generate action sequence* $p$ *of one-hot binary code* $\boldsymbol{x}$
    $p \leftarrow \{a_l | a_l \leftarrow 0, \forall l \in \{1, 2, \cdots \frac{|\Omega|^T-1}{|\Omega|-1}\}\}$
    **for** $l \in \{1, 2, \cdots \frac{|\Omega|^T-1}{|\Omega|-1}\}$ **do**
        $\mathbf{v} \leftarrow \tilde{A}_j^c \cdot \mathbf{x}[(l-1)(|A_j|+1)+1 : (|A_j|+1)l]$
        $k \leftarrow \text{argmax}_{k \in \{1,2,\cdots|A_j|+1\}} \mathbf{v}[k]$
        $p[l] \leftarrow \tilde{A}_j[k]$
    **end**
    $\triangleleft$ *generate policy tree from action sequence* $p$
    $\mathcal{H}^T \leftarrow \bigcup h^T$
    **for** $(h^T, i) \in \mathcal{H}^T$ **do**
        $a^1 o^1, a^2 o^2 ... a^t, o^t, ... a^T o^T \leftarrow h^T$
        **for** $t \in \{1, 2, \cdots T\}$ **do**
            $l \leftarrow \frac{|\Omega|^{(t-1)}-1}{|\Omega|-1} + \lfloor \frac{i-1}{|\Omega|^{(t-1)}} \rfloor + 1$
            $a^t \leftarrow p[l]$
            **if** $t \neq T$ **then**
                $k \leftarrow \lfloor \frac{(i-1)|\Omega|^t}{|\Omega|^{(T-1)}} \rfloor + 1$
                $o^{t+1} \leftarrow o_k$
            **end**
        **end**
        $h^T \leftarrow a^1 o^1, a^2 o^2 ... a^t, o^t, ... a^T o^T$
    **end**
**return** $\mathcal{H}^T$
```

The One-hot encoding operator is described in Section ”Zig-Zag One-Hot encoding & decoding” and outlined in Algorithm ”VAE-Enabled Behaviors”.The *one-hot* operator $\mathcal{I}$ used to convert the summary vector output by the VAE network into a binary encoding format that matches the one-hot encoding of the policy tree. As depicted in Alg. 3, the key steps of each operator are as follows: Initialize a vector $\mathbf{x}$ to represent the one-hot encoding of the policy tree (line 3). Then, for each node in the policy tree (lines 4-8).Locate the index with the maximum value in the corresponding sub-binary representation (line 6). Represent that sub-binary encoding using the enlarged action space $\tilde{A}_j$ (line 7).

Algorithm 3: *One-Hot* Encoding Operator

```
1  ◁ One-hot encoding
2  Function $\mathcal{I}(\tilde{\boldsymbol{x}})$   ◁ $\mathbf{x} \leftarrow \mathcal{I}(\tilde{\mathbf{x}})$:
3      $\mathbf{x} \in \mathbb{R}^{(|A_j|+1)\frac{|\Omega|^T-1}{|\Omega|-1}}$
4      for $l \in \{1, 2, \cdots \frac{|\Omega|^T-1}{|\Omega|-1}\}$ do
5          $\mathbf{v} \leftarrow \tilde{\mathbf{x}}[(l-1)(|A_j|+1)+1 : (|A_j|+1)l]$
6          $k \leftarrow \mathrm{argmax}_{k\in\{1,2,\cdots|A_j|+1\}} \mathbf{v}[k]$
7          $\mathbf{x}[(l-1)(|A_j|+1)+1 : (|A_j|+1)l] \leftarrow \tilde{A}_j[k]$
8      end
9  return $\mathbf{x}$
```