A APPENDIX

Our code [2] and dataset [3] are available in our site.

A.1 Algorithms for Data Generations

We employed a multi-agent simulator [47] for simulating an evacuation scenario from the actual drawing design of the National Theatre, Tokyo. The building consisted of three halls containing an opera house, a playhouse, and a pit. In more detail, the opera house consisted of four floors whose number of seats are 868, 354, 292, and 200, respectively. The playhouse contained two floors whose number of seats are 851 and 187. The pit contained 358 seats. In total, our simulator consisted of $3,210$ seats (spaces) and $L = 7$ areas.

To consider different initial locations of crowds, we selected the occupancy rates of seats on each floor from 30%, 60%, and 90%, which means that we generated $3^7 = 2,187$ covariates. When we set the rate to 50%, we randomly assigned agents to 50% of seats in a floor by the uniform distribution. We set $d_l = 1$, and $x_{l,i} = 1$ if the i-th seat in the i-th floor was occupied by an agent or $x_{l,i} = 0$ otherwise. We show a pseudo code for data generations in Algorithm 1, where $P_i = \{p_1, \ldots, p_L\}$ is the population of all floors for a simulation.

A.2 Factual Treatment Selection

We define a probabilistic policy (the propensity scores) for selecting a factual guidance $\boldsymbol{a}$. For k-th guide location, we set p_k as the populations of occupied seats in the nearest area $c_k \in C$. We randomly sampled guide locations without replacement by using a multinomial distribution whose parameters were set as:

$$p_k = \frac{\sum_{i=1}^{N_{c_k}} X_i^{c_k}}{X_{\max}^{c(k)}},$$
$$\bar{p}_k = \exp(\beta * p_k) / \sum_{k=1}^{K} \exp(\beta * p_k), \tag{12}$$

where $X_{\max}^{l}$ is the maximum number of crowds in the l-th area. We set $\beta = 1$ in our experiments. We show a pseudo code for our factual selection procedure in Algorithm 2

A.3 Models and Configurations

We implemented Mult. NN, Mult. NN w/ MMD, Mult. Cum. NN, and Mult. Cum. NN w/MMD using Pytorch. We set the hyperparameters for them as: (1) the number of hidden units of SGCN $\in \{20, 40\}$ and MLPs $\in \{200, 400\}$; (2) the regularization weight for MMD $\lambda \in \{10^{-7}, 10^{-6}, 10^{-5}\}$. We set the batch size to 32 and $\lambda_3 = 10^{-8}$. We used the Adam optimizer with initial learning rate 0.01, gradient clipping, and decayed the learning rate by 50 every 50 epochs, where the maximum epoch was 500. To keep the output greater than 0, we used the rectified linear unit.

For Ridge regressions, the ℓ_2 regularization weight was selected from $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$. We set the hyperparameters of random forest regressions and the extreme tree regression as: (1) the number of estimators $\{2, 5, 10, 50, 100, 200\}$; (2) the maximum depth of trees $\{2, 5, 10, 50, 100\}$. For MLP regressions, we set the hyper parameters as: (1) the number of hidden layers $\{2, 3\}$; (2) the number of hidden units $\{50, 100\}$. We used the Adam optimizer with initial learning rate 0.01.

We selected the hyperparameters of gradient boosting regressions from: (1) the proportion of sub samples $\{0.1, 0.2, 1.0\}$; (2) the maximum depth of trees $\{2, 4, 6\}$; (3) the weight of the ℓ_2 regularizer $\{10^{-2}, 10^{-1}, 10^{0}\}$.

We employed the implementations provided by Scikit-learn and XGBoost libraries in our experiments.

A.4 Computation Time Comparison

We implemented our proposed method with Pytorch. We run our method on a GPU server with AMD EPYC 7413 Processor (2.65GHz) and NVIDIA RTX A5000. We used the original Java code of a multi-agent simulator and ran it on a CPU server with Intel Xeon Gold 6148 Processor (2.4 GHz).

Algorithm 1 An algorithm for a data generation

```
Require: $A_K, P_i$
  $Y = \{\}$
  while $l \le L$ do
    $p_l = P_i[l]$
    while $n \le N_l$ do
      $q_{l,n} \sim \mathrm{Unif}(0,1)$
      if $q_{l,n} \le p_l$ then
        $x_{l,n} = 1$
      else
        $x_{l,n} = 0$
      end if
    end while
  end while
  for $\boldsymbol{a} \in A_K$ do
    Run a multi-agent simulator [47] and get $\boldsymbol{y}$.
    $Y = Y \bigcup \boldsymbol{a}$
  end for
  return Y
```

Algorithm 2 An algorithm for a factual treatment selection

```
Require: $X, X_{\max}, K', \beta$, C
  $Z = \{\}$
  $P = \{p_1, \ldots, p_K\}$
  while $k \le K$ do
    $p_k = \frac{\sum_{i=1}^{N_{c_k}} X_i^{c_k}}{X_{\max}^{c_k}}$
  end while
  while $k \le K$ do
    $p'_k = \exp(\beta * p_k) / \sum_{k=1}^{K} \exp(\beta * p_k)$
  end while
  while $k \le K'$ do
    $Z_k \sim \mathrm{Multinomial}(P)$
    $Z = Z \bigcup Z_k$
    $P = P \setminus Z_k$
  end while
  return Z
```

[2]https://github.com/SIMON202202/SIMON
[3]https://1drv.ms/u/s!AvkPhNiV_FS7ah_SCkYugU1Qc4g?e=HSQfZM