A Experiment Details

A.1 Non-Cooperative Navigation

Non-Cooperative Navigation is developed based on the Multi-agent Particles Environment (MPE) [10]. n agents are required to maximize their coverage over n landmarks without any explicit cooperation or inter-agent communication mechanism. Instead of being assigned some predetermined landmarks as their destinations, agents are attracted to the immediate closest landmark at each time step. This indicates that an agent's destination is not fixed in an episode and that multiple agents can be attracted to a specific landmark simultaneously. Agents should properly select their intention landmarks, reach and stay at their intended landmarks, and avoid any conflicts with other agents. The length of each episode is 50 steps. Agents and landmarks are randomly initialized within a 2×2 world space. All plots in Non-Cooperative Navigation are averaged over 3 random seeds.

In Non-Cooperative Navigation, there are three different kinds of agents that are controllable by MARL policies and one kind of agent that is controlled by the pre-defined random policy taking random actions at each time step. Table 2 shows the parameters of different kinds of agents; their major differences come from their sizes and acceleration values:

Agent Type	Size	Acceleration
Normal	0.08	1.0
Tiny	0.06	1.1
Bulky	0.10	0.9
Random	0.08	1.0

Table 2: Parameters for Agents used in Non-cooperative Navigation

Scenarios. Two scenarios with different heterogeneity levels are included in this paper:

- **Easy:** 1 Normal agent, 1 Tiny agent, and 1 Bulky agent.
- **Hard:** 1 Normal agent, 1 Tiny agent, 1 Bulky agent, and 1 Random agent.

Note that all agents in the *easy* scenario are controllable. One uncontrollable agent exists along with three controllable agents in the *hard* scenario, which makes this scenario more heterogeneous.

Observation Space. Non-Cooperative Navigation is a fully-observable environment with a continuous observation space for each agent. The observation vector of an agent is composed of state vectors of all entities within the world space, including the states of all agents and landmarks. Here, we denote the state of an entity in Non-Cooperative Navigation as a vector with its ID, current position, and velocity. Within agent i's observation vector, the positions of all entities are their positions with respect to agent i. Agent i's ego state vector locates it at the top of its observation vector and uses its own absolute position in the world space. For those centralized MARL algorithms requiring the global state, the global state is the collection of all entities' state vectors composed of their IDs, absolute positions, and velocities in the world space.

Action Space. Non-Cooperative Navigation has a discrete action space with 5 identical high-level actions, $\{$*idle*, *up*, *down*, *left*, *right*$\}$. Taking action in any direction (*i.e.*, all actions except *idle*) makes this agent accelerate by one step size in that direction. The acceleration step size varies in different kinds of agents.

Reward. Each agent has an individual reward function in Non-Cooperative Navigation. An agent gets a penalty that equals its distance from the closest landmark in the environment at each time step. Notably, multiple agents may get this penalty with respect to their distances to a specific landmark if this landmark is the closest to all of them. If a collision happens between two agents, both will receive a penalty of -5. If an agent reaches the scope with a distance of less than 0.1 to any landmarks, this agent receives a positive reward of 10. We denote this scope as the *rewarding scope*. If all controllable agents reach and stay within the *rewarding scope* without conflicts, they all receive a positive reward of 100.

A.2 Heterogeneous Highway

Heterogeneous Highway is developed based on Highway-env [20], which is a 2D autonomous driving simulator based on PyGame. Traffic scenarios in our environment are designed based on the Highway scenario given by Highway-env with simulated vehicles driving on a multi-lane highway. The objective of vehicles controlled by MARL algorithms is to maintain a collision-free trajectory

with a proper speed between 20 and 30 m/s when driving through heterogeneous traffic. Uncontrollable vehicles are controlled by three different behavior-driven vehicle models modified from models proposed in [61], and we denote them as *Normal*, *Aggressive* and *Conservative* vehicles. Their major differences come from their kinematic features, given in Table 3.

Kinematic Parameters	Normal	Aggressive	Conservative
Max Speed (m/s)	40	50	40
Default Speed Range (m/s)	$[23, 25]$	$[35, 40]$	$[23, 25]$
Max Acceleration (m/s^2)	6.0	9.0	5.0
Desired Acceleration (m/s^2)	3.0	6.0	2.0
Desired Deceleration (m/s^2)	-5.0	-9.0	-4.0
Desired Front Distance (m)	$5.0 + l$	0.5	$8.0 + l$
Time Wanted (Before Stop) (s)	1.5	1.2	1.8

Table 3: Kinematics for the behavior-driven vehicle model used in Heterogeneous Highway scenarios. All vehicles are assumed to have the same size l.

The length of each episode is 90 steps. Initially, vehicles are randomly placed throughout the world space with a density of 1. All results in Heterogeneous Highway are averaged over 3 random seeds.

Scenarios. Two scenarios under *mild* and *chaotic* traffic are included in this paper. Each scenario has 5 controllable vehicles and 50 behavior-driven vehicles uniformly distributed over an 8-lane highway. The compositions of different behavior-driven vehicles relate to the heterogeneity of traffic. The *mild* traffic has mostly normal-behaving vehicles, so we consider this scenario more homogeneous. In the *chaotic* traffic scenario, more aggressive vehicles exist, which makes the environment more heterogeneous. Here are the propositions of each kind of behavior-driven vehicle in the *mild* and *chaotic* traffic scenarios:

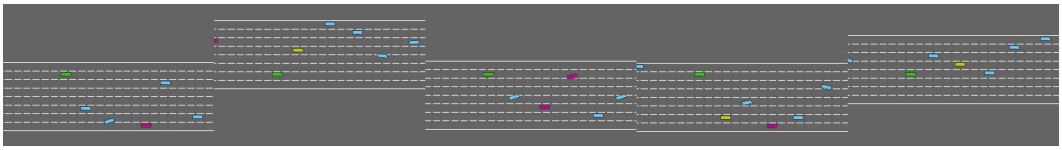
- **Mild:** 80% Normal vehicles + 10% Aggressive vehicles + 10% Conservative vehicles.
- **Chaotic:** 40% Normal vehicles + 30% Aggressive vehicles + 30% Conservative vehicles.

Observation Space. Heterogeneous Highway is a partially-observable environment in that agents can only observe 15 other vehicles within their predefined observation scope. The observation scope for each agent is 100 m in both directions of the x-axis and 20 m in both directions of the y-axis. Each agent has a continuous observation space. The observation vector of an agent is composed of stacked state vectors of all vehicles within its observable scope. Here, we denote a state vector of a vehicle as a vector with its ID, current position, and velocity in the world space. For agent i's observation vector, its ego state vector locates it at the top of its observation vector and uses its own absolute position in the world space. The remaining state vectors are state vectors of vehicles observed by agent i using their positions relative to agent i. The global state for centralized MARL baselines is made up of concatenated state vectors of all controllable and uncontrollable vehicles within the environment.

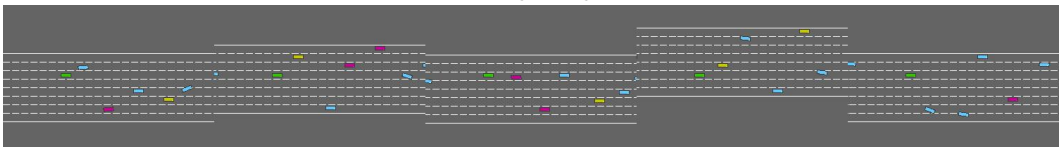
Action Space. The action space for each controllable agent is discrete with 5 distinct actions, {*lane left*, *idle*, *lane right*, *faster*, *slower*}. Vehicles convert their high-level discrete action orders into a sequence of x, y coordinates when taking actions. All vehicles' low-level motion models follow the Kinematic Bicycle Model [62], and their kinematic parameters are given in Table 3.

Reward. For distributed MARL algorithms, each agent receives an individual reward, while for centralized MARL algorithms, all agents receive a global reward by summing their individual rewards together. Once an agent collides with other vehicles, this agent gets a -1 penalty. Agents are encouraged to keep right, and an agent gets a linear reward from 0 to 0.1 with respect to its distance to the rightmost lane. Agents are encouraged to keep a speed within the rewarding speed range of 20 to 30 m/s. At each time step, an agent is rewarded with respect to its speed within the reward speed range. If an agent can reach a speed of 30 or higher at this time step, it gets a reward of 0.4. If an agent keeps a speed of 20 or lower at this time step, it gets a reward of 0.

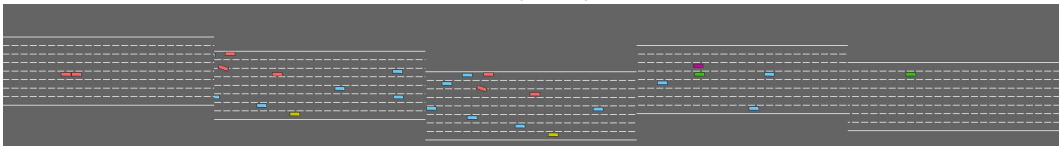
B Visual Results



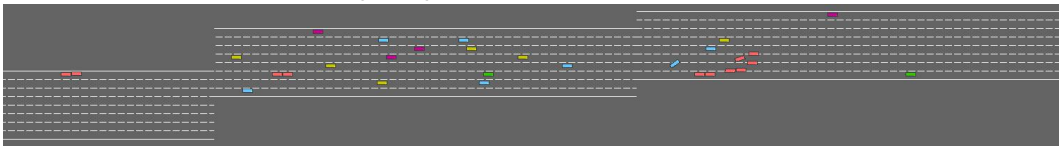
(a) *(Easy)* **iPLAN**: All 5 agents (green) are successful.



(b) *(hard)* **iPLAN**: All 5 agents (green) are successful.

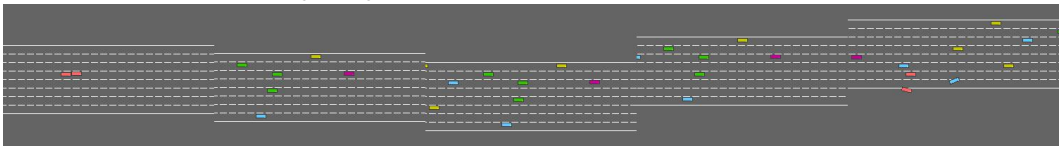


(c) *(Easy)* **MAPPO**: 2 agents (green) are successful. The first 3 crash (red vehicles).



(d) *(hard)* **MAPPO**: 2 agents (green) are successful. The first, second, and fourth crash (red vehicles).

(e) *(Easy)* **QMIX**: 3 agents (green) are successful. The first and the last crash (red vehicles).



(f) *(hard)* **QMIX**: 4 agents (green) are successful. The third vehicle crashes (red vehicle).

Figure 3: **Qualitative results on Heterogeneous Highway:** We visually compare the performance of iPLAN with QMIX and MAPPO. Each baseline is tested with multiple learning agents shown in green, and each figure above shows 5 such learning agents from their respective viewpoints. In each figure, we show cases when the green agents succeeded versus when they crashed. **Conclusion:** All 5 agents succeed using iPLAN as shown in Figures 3a and 3b whereas on average 2 or more agents crash using QMIX or MAPPO.

C Implementation Details

Behavioral Incentive Inference. The encoder of the behavioral incentive inference module uses a 1-layer GRU network with a size of 32 and generates an 8-length vector as the latent representation of the behavioral incentive. The decoder uses another 1-layer GRU network with a size of 64 to reconstruct the state sequences, with a dropout rate of 0.1. The truncated length t_h of the observation history is 10 in Heterogeneous Highway, and 5 in Non-Cooperative Navigation. The learning rate for behavioral incentive inference is 1×10^{-4}.

Instant Incentive Inference. The encoder of the instant incentive inference module uses a GAT with a hidden-layer size of 32 and a 1-layer GRU with a hidden-layer size of 32. The decoder uses another 32-size GRU to predict the trajectory, with a dropout of 0.1. The trajectory prediction length t_p is 5 in Heterogeneous Highway, and 2 in Non-Cooperative Navigation. The learning rate for instant incentive inference is 2×10^{-5}.

IPPO Controller. The input of the PPO controller for an agent is the flattened vector of its observation of all entities' (vehicles in Heterogeneous Highway; other agents and landmarks in Non-Cooperative Navigation) states and the inference of all other agents' (or other vehicles') behavioral incentive and instant incentive. The PPO controller has a buffer size of 256 and a learning rate of 5×10^{-4} for its actor and critic. All fully-connected and recurrent layers in the actor and critic of PPO have a dimension of 64.

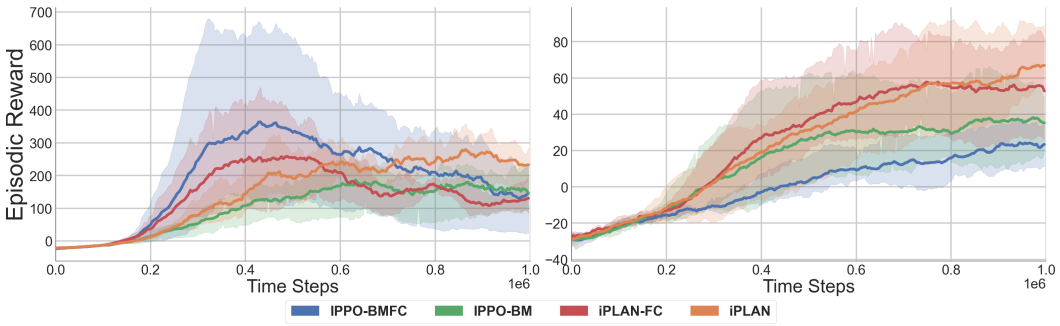


Figure 4: **Non-Cooperative Navigation with recurrent and fully-connected behavioral incentive inference modules:** Comparing the episodic reward in the (left) *easy* and (right) *hard* scenarios. **Conclusion:** iPLAN (orange) performs better than others in the *easy* scenario. IPPO-BM (green) outperforms IPPO-BMFC (blue) in the *hard* scenario.

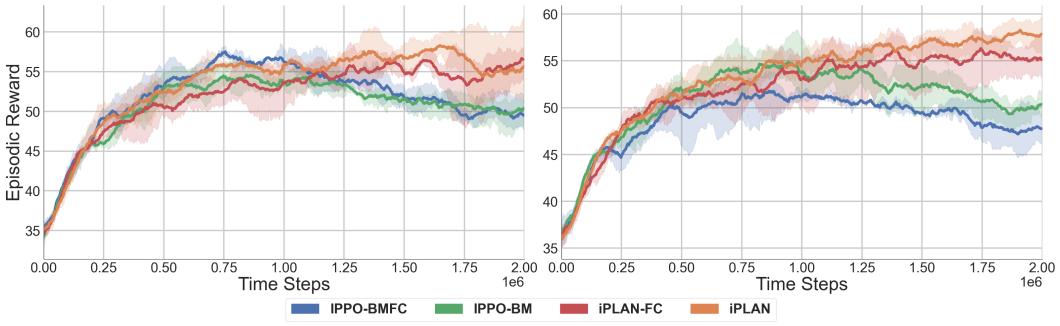


Figure 5: **Heterogeneous Highway with recurrent and fully-connected behavioral incentive inference modules:** Comparing the episodic reward in the (left) *mild* and (right) *chaotic* traffic scenarios. **Conclusion:** Approaches using recurrent behavioral incentive inference modules, including iPLAN (orange) and IPPO-BM (green), outperform those using fully-connected behavioral incentive inference modules.

D Supplementary Experiments: Behavioral Incentive Inference Module

D.1 Choice of Behavioral Incentive Inference Module

During our design process for the behavioral incentive inference module, we experimented with different architectures in the encoder-decoder framework. Specifically, we tested the usage of a recurrent layer and a fully-connected layer. While the latter design has been utilized in prior works for similar tasks [16, 17, 47], we want to address the temporal relationship presented in the historical observation sequences. To evaluate the performance of these two designs, we conduct experiments on the comparison between iPLAN and an alternative approach that uses a fully-connected behavioral incentive inference module.

In this module, we take the flattened historical observation sequence as input and employed a 3-layer fully-connected network with a hidden layer dimension of 64 as the encoder. This encoder generates an 8-length latent representation of the behavioral incentive. Additionally, we use another 3-layer fully-connected network with the same hidden layer dimension as the decoder to reconstruct the state sequences for opponents. The learning rate for this alternative behavioral incentive inference module is set to 1×10^{-4}.

We depict the episodic rewards over both environments in Figure 4 and Figure 5. In these figures, the approach employing the fully-connected network in the behavioral incentive inference module is denoted as iPLAN-FC, and the same notation applies to IPPO-BMFC. The results indicate that incorporating the recurrent layer improves the performance of the behavioral incentive inference module. Specifically, our approach (iPLAN, orange curve) demonstrates better performance than iPLAN-FC (red curve). Similarly, IPPO-BM (green curve) outperforms IPPO-BMFC (blue curve) in general.

D.2 Soft Updating Policy

Another important aspect to consider in our behavioral incentive inference module design is the updating policy for behavioral incentives. Drawing inspiration from previous works [16, 17, 47],

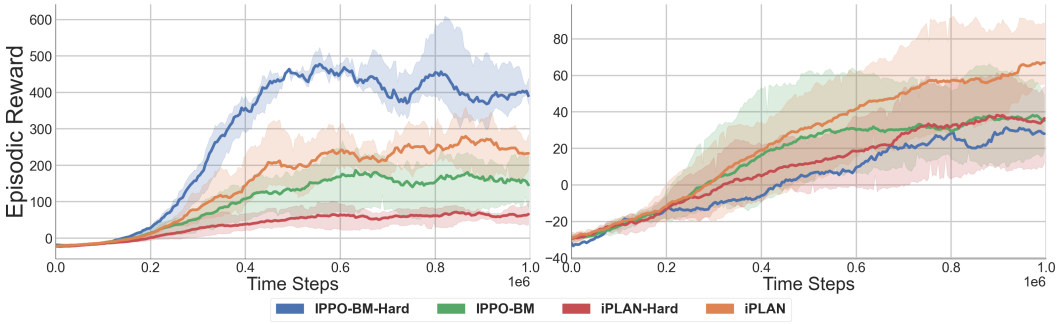


Figure 6: **Non-Cooperative Navigation with and without soft-updating policy:** Comparing the episodic reward in the (left) *easy* and (right) *hard* scenarios. **Conclusion:** IPPO-BM-Hard (blue) performs the best in the *easy* scenario and the worst in the *hard* scenario. iPLAN (orange) has a better performance in general.

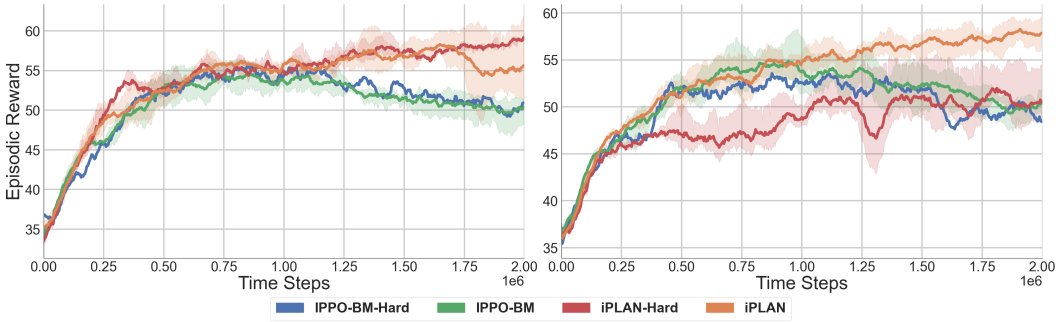


Figure 7: **Heterogeneous Highway with and without soft-updating policy:** Comparing the episodic reward in the (left) *mild* and (right) *chaotic* traffic scenarios. **Conclusion:** iPLAN (orange) that uses soft-updating policy for behavioral incentive inference module greatly outperforms its alternative approach iPLAN-Hard (red) that uses a hard-updating policy.

we divide the behavioral incentive inference within an episode into multiple sub-episodes. We aim to update the behavioral incentive inferences at the end of each sub-episode. This updating policy is referred to as the *hard-updating policy*, in contrast to the *soft-updating policy*, which treats the behavioral incentive inference as a converging procedure and iteratively updates the behavioral incentive inferences.

In our experiments, we evaluate the performance of iPLAN and an alternative method, iPLAN-Hard, which employs a hard-updating policy. In iPLAN-Hard, the behavioral incentive inference module updates the behavior incentives at specific time intervals (e.g., $t = 10, 20, 30, \ldots$), while the behavior incentive inferences remain unchanged between these updating points (*i.e.*, between $t = 10$ and $t = 20$). All other hyperparameters used in the behavioral incentive inference module remain the same.

Figure 6 and Figure 7 illustrate the results obtained with different behavior incentive updating policies. In Non-Cooperative Navigation, IPPO-BM-Hard achieves the best performance in the *easy* scenario but performs the worst in the *hard* scenario. This significant gap between scenarios may stem from its inability to capture heterogeneity, considering that all agents in the *easy* scenario are controllable. On the other hand, iPLAN exhibits overall better performance, ranking second in the *easy* scenario and first in the *hard* scenario. This outcome demonstrates that the soft-updating policy helps address heterogeneity and stabilize agents' strategies.

In Heterogeneous Highway, iPLAN-Hard denotes the approach that uses a hard-updating policy for behavioral incentives, and the same notation applies to IPPO-BM-Hard. The results reveal that despite the difference in updating policies, their performances remain relatively close in *mild* traffic for both comparison pairs (iPLAN *v.s.* iPLAN-Hard, IPPO-BM *v.s.* IPPO-BM-Hard). However, in *chaotic* traffic, where instant incentive inference is not available, the use of the soft-updating policy leads to a substantial improvement for iPLAN. As agents become more reliant on their inference of others' behaviors and intentions in a highly heterogeneous environment, the reliability and flexibility of their behavioral incentive inference become crucial, enabling them to gain a better understanding of their surroundings.

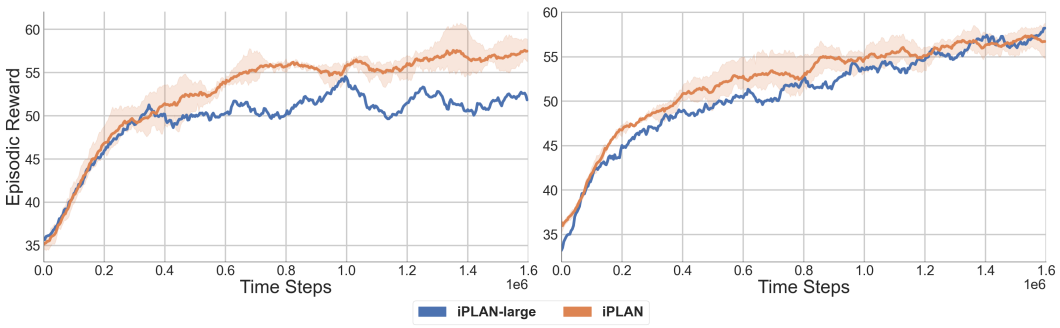


Figure 8: **Heterogeneous Highway with different learning rates for instant incentive inference module:** Comparing the episodic reward in the (left) *mild* and (right) *chaotic* traffic scenarios (with 1.6M training time steps). **Conclusion:** Using a smaller learning rate in instant incentive inference (iPLAN, orange) has a better performance in the *mild* traffic

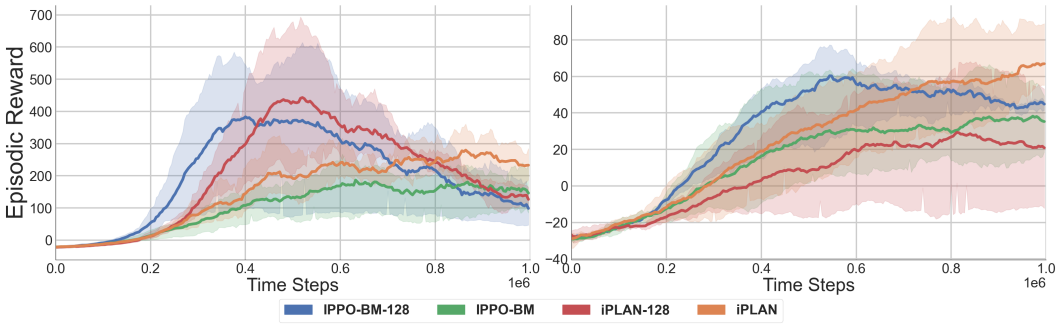


Figure 9: **Non-Cooperative Navigation with different hidden layer dimensions for behavioral incentive inference module:** Comparing the episodic reward in the (left) *easy* and (right) *hard* scenarios. **Conclusion:** Approaches like IPPO-BM-128 (blue) and iPLAN-128 (red) that use a larger hidden layer dimension for behavioral incentive inference do not address the heterogeneity well and suffer from the overfitting problem.

E Supplementary Experiments: Hyper-Parameter Study

E.1 Learning Rate in Instant Incentive Inference

Figure 8 compares the episodic rewards when using different learning rates for instant incentive inference. iPLAN (orange curve) uses a learning rate of 2×10^{-5} and iPLAN-large (blue curve) uses a learning rate of 1×10^{-4}. The result shows that using a smaller learning rate in instant incentive inference has a better performance in practice.

E.2 Hidden Layer Dimension in Behavioral Incentive Inference

Figure 9 presents a comparison of the effect of hidden layer dimensions used in behavior incentive inference. In this figure, we denote the alternative approach iPLAN that utilizes a hidden layer dimension of 128 as iPLAN-128, and the same notation applies to the alternative approach IPPO-BM-128 of IPPO-BM.

In the *easy* scenario, both IPPO-BM-128 (blue curve) and iPLAN-128 (red curve) exhibit significantly better performance than their counterparts using a hidden layer dimension of 64 in the first half of training. However, their episodic rewards experience a substantial decline in the second half, resulting in a lower ultimate episodic reward compared to iPLAN. This observation suggests that these models are overfitting in the *easy* scenario.

In the *hard* scenario, iPLAN (orange curve) outperforms iPLAN-128 (red curve) and IPPO-BM-128 (blue curve), as the episodic reward of IPPO-BM-128 begins to decrease when iPLAN's curve is still increasing. This phenomenon demonstrates that using a larger hidden layer dimension does not necessarily lead to performance improvement, as it can exacerbate the overfitting problem. Additionally, a larger hidden layer dimension may not effectively address the heterogeneity in a more complex and heterogeneous environment, such as the *hard* scenario.

Overall, the results indicate that carefully selecting the hidden layer dimension is crucial. While a larger dimension may offer some benefits, it can also lead to overfitting and failure in addressing the challenges posed by heterogeneity in certain scenarios.