A Details of Environment

We conduct the experiments in a multi-sensor multi-target tracking environment. This environment is developed based on the previous version in HiT-MAC[12]. Differently, we add some new features to make it more complex and realistic.

First, HiT-MAC assumes that all the agents can see all the targets in the environment, which is unreasonable for real applications. Therefore, we change this setting into a Dec-POMDP. An agent can only obtain the information in its local observation. For those targets out of view, the corresponding observation will be a zero vector.

Secondly, we add another kind of objects, obstacles, into the environment. The obstacles are all circles in this 2D plain simulator, varying in the radius. The targets within the observation radius will still be invisible if it is shadowed by an obstacle.

Finally, in the original environment all the targets move in a goal-oriented manner. The targets sample their destinations at the beginning of an episode and navigate themselves to the destinations. Nevertheless, not all targets in real world follow the same action pattern. Therefore, we fill the environment with a mixed-type population of targets. The target can either be goal-oriented or random-walking. When the target is random-walking, it will randomly sample a primitive action to take at each step. In this way, the movement of targets is harder to predict, raising the difficulty of the planning.

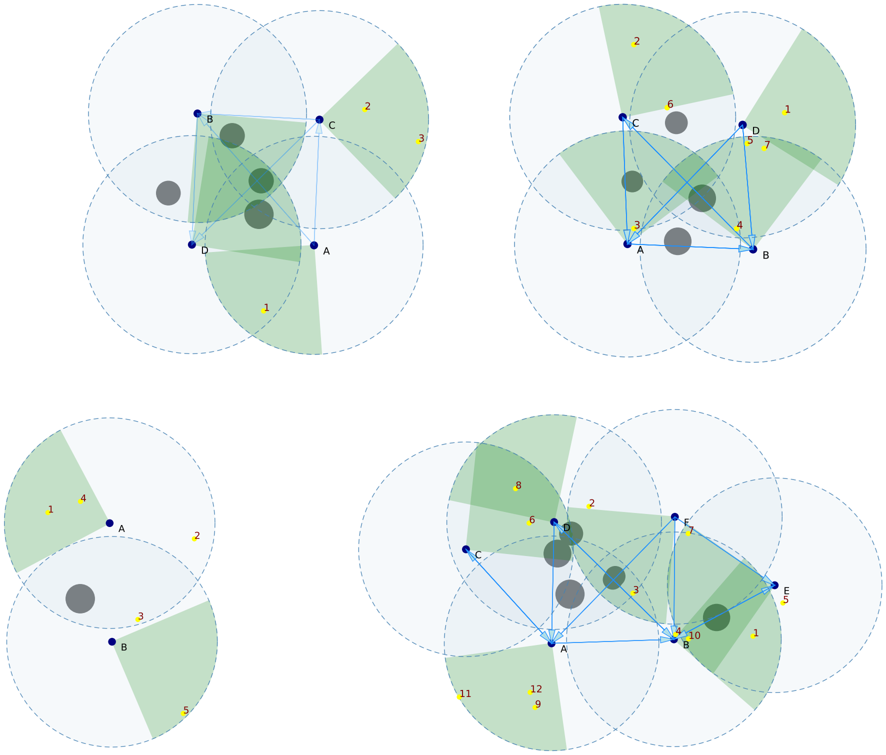


Figure 7: Snapshots of the multi-sensor multi-target tracking environments with different scales. From the upper left figure, in the clockwise direction, they are 4 vs 3, 4 vs 7, 6 vs 12, and 2 vs 5, respectively.

B ToM2C

Network Architecture and Hyper-parameters for ToM2C. The observation encoder consists of 2-layer multilayer perceptron (MLP) and an attention module:$\mathtt{att}_1$. The ToM net consists of a Gated

Recurrent Unit (GRU) and 2-layer MLP. The message sender is a Graph Neural Network (GNN) and the actor consists of one fully connected layer. The critic consists of an attention module att_2 that can handle different number of agents. As mentioned before, the basic RL training algorithm is A2C, and the hyper-parameters are detailed in Tab. 1.

Table 1: Hyper-parameters for ToM2C

Hyper-parameters	#	Description
GRU hidden units	32	the # of hidden units for GRU
att_1 hidden units	64	the # of hidden units for att_1
att_2 hidden units	192	the # of hidden units for att_1
max steps	3M	maximum environment steps sampled in workers
episode length	100	maximum time steps per episode
discount factor	0.9	discount factor for rewards
entropy weight	0.005	parameter for entropy regularization
learning rate	1e-3	learning rate for all networks
workers	6	the # of workers for sampling
update frequency	20	the network updates every # steps in A2C
ToM Frozen	5	the ToM net is frozen for every # times of RL training
gamma rate	0.002	the increasing rate of discounting factor γ

Training Strategy. There are two training strategies adopted to accelerate training and stabilize the result. As mentioned in Sec.3.5, one is to increase episode length L and γ factor gradually during training, the other one is to split the optimization of the ToM and RL model.

In this paper, we propose this curriculum learning strategy that gradually increases episode length L and discounting factor γ. Usually, the discounting factor γ is set larger than 0.9 to encourage long-term planning in RL algorithms. Furthermore, the length of an episode is usually determined by the environment. We notice that if using the default hyper-parameters, the agents performs sample inefficient and unstable while learning. In our experiments, we set $L = 20$ and $\gamma = 0.1$ initially. After 2000 episodes of warm up, the γ factor will be updated according to a pre-set rate β. Each time the network is optimized through reinforcement learning, $\gamma = \gamma * (1 + \beta)$, where $\beta = 0.002$ in this paper. Simultaneously, the episode length L is updated with γ. In fact, $L = \lfloor \frac{\gamma+0.1}{0.2} \rfloor \times 20$. In the end, $\gamma = 0.9$ and $L = 100$. By doing so, the agents learn short-term plan first, and then adapt to a longer horizon. We find in experiments that such strategy accelerates the training process, leading to a faster convergence and a better performance.

Furthermore, we separate the optimizing of the ToM and RL model in implementation. Before the training process starts, the parameters of our model are split into two parts: θ^{ToM} and θ^{other}. Each part is optimized by an individual optimizer. Since we adopt A2C as the basic RL training algorithm, we collect trajectories data from different worker processes and send them to the training process when all the running episodes end. After that, θ^{other} is optimized with regard to the A2C loss. Meanwhile, the trajectories data for ToM training are saved instead of being used for training ToM net immediately. In this way, the ToM net is 'frozen'. θ^{ToM} will be optimized with regard to ToM loss after θ^{other} has been optimized for T_F times. Here we choose $T_F = 5$. Just like the discussion before, the separation of ToM and RL training avoids the nested loop of influence among the ToM net and the policy network.

The environment and model are implemented in Python. The model is built on PyTorch and is trained on a machine with 7 Nvidia GPUs (Titan Xp) and 72 Intel CPU Cores.

C Baselines

Heuristic Search Method. To evaluate the performance of our ToM2C model, we choose to implement a heuristic search policy to serve as a reference. This search policy is applied to select low-level sensor action(Stay,Turn Left/Right). At each step, the policy searches all the 3^n possibilities of combination of actions, where n is the number of sensors. The goal is to find the action combination that minimizes the angle distance of targets to sensors. Specifically, we denote the angle distance

Table 2: Coverage Rate in 4 sensors vs 5 targets scenario

Methods	Coverage Rate(%)↑
A2C	38.44± 0.54
HiT-MAC	61.48± 1.45
I2C	66.29± 1.40
ToM2C-ToM	67.66± 0.63
ToM2C-Comm	71.61± 0.31
ToM2C(Ours)	**75.38± 0.57**

Table 3: Communication Statistics

Methods	Communication Edges	Communication Bandwidth↓
I2C	7.16	257.76
HiT-MAC	8.0	164
FC	12.0	60
ToM2C w/o CR	9.36	46.81
ToM2C(Ours)	**6.02**	**30.08**

of target j to sensor i as α_{ij}. Then the objective is to minimize $\sum_{j=1}^{m} \min_i\{\alpha_{ij}\}$. It is obvious that such searching policy only considers one step, thus not the optimal policy. However, we show that this naive heuristic search can reach 80% target coverage. As a result, it can serve as a reference 'upper bound' that evaluates all the MARL baselines.

MARL Baselines. The code of HiT-MAC and I2C are from their official repositories. We follow the default hyper-parameters in their code, except that we change the learning rate, discounting factor γ and episode length to be the same as ToM2C. Moreover, HiT-MAC is a hierarchical method, so we simply train the high-level coordinator and use the same rule-based low-level policy utilized in ToM2C. On the other hand, I2C is not a hierarchical method and it is not target-oriented. As a result, we concatenate all the target information into one vector as the observation for I2C. The action space is modified as the set of choice of all the targets, so the space size is 2^m, where m is the number of targets. In this way, the output action of I2C agent is the selection of goal targets, same as HiT-MAC and ToM2C. Once the goal target is selected, the primitive actions will be chosen by the rule-based policy.

D Quantitative Results

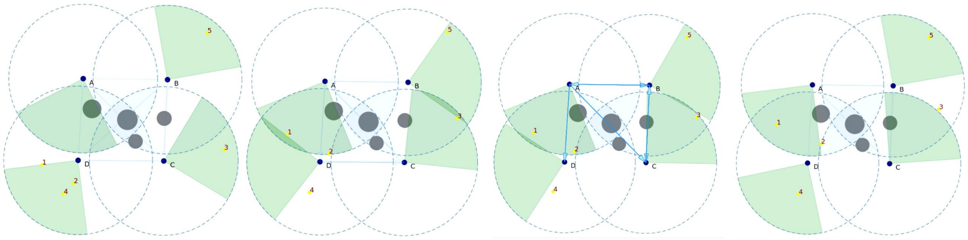
We list the **coverage rate** achieved by different methods in Tab. 2. The mean and standard deviation is computed based on the data collected in 1000 episodes.

Apart from coverage rate, we analyze the communication efficiency of different methods. There are 2 metrics introduced in this paper. Communication edges refers to the count of directed communication pairs. One edge from i to j means that agent i sends a message to agent j. Communication bandwidth refers to the total volume of messages. As we explained in the experiment section, it is the volume of messages that has decisive effect on the cost of communication. Since the messages are all float-type vectors, we use the length of message instead of the number of bits to represent the volume of a single message. For I2C, the message from agent i is the local observation o_i, containing the information of all the targets. For HiT-MAC, the communication happens between the executors and coordinator. The executors send its local observation to the coordinator, and the coordinator returns the goal assignment. For fully connected communication (FC), ToM2C w/o CR and ToM2C, the message is simply the inferred goals of the receiver. ToM2C w/o CR means that the trained ToM2C model is not further optimized to reduce communication.

The experiment is conducted in the 4 sensors 5 targets scenario. As is shown in Tab. 3, our method achieves the lowest communication cost both in communication edges and bandwidth. Moreover, even FC beats I2C and HiT-MAC in communication bandwidth. It is because FC only sends the inferred goals, which is much simpler than the raw observation.

E Demo Sequence

For a better understanding of the learned behavior, we render the environment and show a typical demo sequence in Fig. 8. It consists of 4 consecutive keyframes in one episode. The arrows between sensors indicate communication connection. Note that communication only happens every 10 steps. In step 16, sensor *D* can track target 1, 2 and 4. However when it comes to step 22, sensor *D* can no longer track all the three targets, so it starts to hesitate about which targets to track. Then in step 24, *A* sends a message to *D*, and *D* inferred that *A* would track target 1 and 2. Therefore, it re-plans its own goal to be target 4. In the end, we can see that sensor *D* really abandons target 1 and 2, and focuses on target 4.



Step 16 **Step 22** **Step 24** **Step 29**

Figure 8: An exemplar sequence in 4 sensors and 5 targets environment. The gray circle indicates the obstacle. The arrows are rendered as solid only when the communication happens, and transparent at other times.