Supplementary material for Multi-UAV Cooperative Pursuit Strategy with Limited Visual Field in Urban Airspace: A Multi-Agent Reinforcement Learning Approach

S1: The way of judging whether the evader is visible

For the pursuers set P and the evader e, the visibility state can be denoted by ε^t, where $\varepsilon^t = 0$ signifies the evader is outside the pursuers' visual field, otherwise $\varepsilon^t = 1$. Therefore, when $\varepsilon^t = 1$, the following conditions must be satisfied:

$$\begin{cases} \|\mathbf{p}_i^t - \mathbf{p}_e^t\|_2 \leq L \\ |\psi_i^t - \rho_{ie}^t| \leq \dfrac{\varphi}{2} \quad \exists\ i \in P \\ bs_{ie}^t = 0 \end{cases} \tag{S1}$$

where bs_{ie}^t denotes the blockage state between pursuer i and evader e at time t. $bs_{ie}^t = 0$ signifies an unobstructed connection between the pursuer i and evader e, indicating no obstruction by any building, conversely, $bs_i^t = 1$ indicates blockage. Here ρ_{ie}^t represents the relative azimuth between pursuer i and evader e at time t, which can be computed using the following formula:

$$\rho_{ie}^t = atan2\left(\frac{y_e^t - y_i^t}{x_e^t - x_i^t}\right) \tag{S2}$$

where $atan2$ function is an inverse tangent commonly utilized for computing the azimuth angle.

S2: Solution existence analysis based on Apollonius circle

In this problem, both the pursuers and the evader have their own advantages: the pursuers have more numbers, while the evader has a higher maximum speed. Assuming the ratio of the maximum speeds of the pursuers and the evader is $\Lambda = v_{i,max} : v_{e,max}$. The set of points that both the evader e and the pursuer i can reach simultaneously when traveling at their maximum speeds can form an Apollonius circle, as illustrated in the Fig. S1. Given a point $\mathbf{p}_c$ on the circle, the ratio of

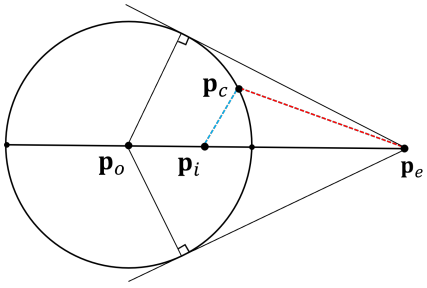

Figure S1. The Apollonius circle. $\mathbf{p}_o$ represents the coordinates of the circle's center.

distances $d_{ic} : d_{ec} = \Lambda$. The radius of the Apollonius circle can be calculated using the following formula:

$$r = \Lambda\sqrt{(x_i - x_e)^2 + (y_i - y_e)^2}/(1 - \Lambda^2) \tag{S3}$$

With a constant maximum speed ratio Λ, the radius is solely dependent on the distance between the pursuer and the evader. Therefore, the characteristics of the Apollonius circle lead to the following theorem: the pursuer can capture the evader

Table S1
THE INPUT INFORMATION OF PURSUERS, GLOBAL STATE, AND THE INPUT INFORMATION OF EVADER UNDER REINFORCEMENT LEARNING STRATEGY

Input	Component	Detailed Information
For pursuers: Observation of **each** pursuer (Take pursuer i, for example)	1. Agent's own state information	The coordinates $\mathbf{p}_i^t$, yaw angle ψ_i^t, velocity $\mathbf{v}_i^t$ and obstacle information $\boldsymbol{\lambda}_i^t$ of pursuer i.
	2. Teammates' state information	The relative coordinates $\Delta\mathbf{p}_{ij}^t$, distance d_{ij}^t, relative yaw angle $\Delta\psi_{ij}^t$, and azimuth angle ρ_{ij}^t of **all allied pursuers**, $j \in \{1, 2, \cdots, N\}$ and $i \neq j$. (Relative to the pursuer i)
	3. Evader's information	- If pursuers are in the pursuit phase: The relative coordinates $\Delta\mathbf{p}_{ie}^t$, distance d_{ie}^t, and azimuth angle ρ_{ie}^t of evader e. (Relative to the pursuer i) - If pursuers are in the reacquisition phase (approach): The relative coordinates $\Delta\mathbf{p}_{i\beta}^t$, distance $d_{i\beta}^t$, and azimuth angle $\rho_{i\beta}^t$ of latest lost point β. (Relative to the pursuer i) - If pursuers are in the search phase or reacquisition phase (search): Replaced by zero
Global state	-	The coordinates $\mathbf{p}^t$, yaw angle ψ^t, velocity $\mathbf{v}^t$, and obstacle information $\boldsymbol{\lambda}^t$ of **all pursuers and the evader**.
For evader: State of evader (Reinforcement learning strategy)	1. Evader's own state information	The coordinates $\mathbf{p}_e^t$, yaw angle ψ_e^t, velocity $\mathbf{v}_e^t$ and obstacle information $\boldsymbol{\lambda}_e^t$ of evader e.
	2. Pursuers' state information	The relative coordinates $\Delta\mathbf{p}_{ei}^t$, distance d_{ei}^t, relative yaw angle $\Delta\psi_{ei}^t$, and azimuth angle $\rho_{ei}^t$of **all pursuers**, $i \in \{1, 2, \cdots, N\}$. (Relative to the evader)

if and only if the evader's path intersects the Apollonius circle and the capture point lies within the circle. We extend this theorem to the scenario involving N pursuers, where the evader can be ensured of capture if the set of Apollonius circles formed between each pursuer and the evader tightly encloses the evader. This tight enclosure of the Apollonius circles must satisfy the condition: $\Lambda \leq \sin(\pi/N)$. Hence, to ensure the existence of capture solutions, $\Lambda \in [\sin(\pi/N),\ 1)$. Given that the initial position of the evader is randomized, and to prevent the evader from infinitely escaping the pursuers, the evader is also constrained by the bounded airspace in this problem. Within the designated airspace, the pursuers can coordinate effectively to establish the conditions for capturing the evader. Additionally, when the evader is near the boundary, the pursuers can use the boundary to limit the evader's movement direction, thereby achieving capture.

S3: The input information of pursuers, global state, and the input information of evader

Please refer to the Table. S1.

S4: Computational complexity analysis of NAGC and comparison with other algorithms

Computational complexity analysis of NAGC:

Actor network: Assuming the input dimension of the Actor network is M_o, with the output dimension is 2. The hidden layer dimension is M_h, and the number of hidden layers is $\mathcal{L}$. Therefore, the computational complexity of Actor network is $\mathcal{O}\left(M_o M_h + \mathcal{L}{M_h}^2 + 2M_h\right)$. Since we have added the NF operation to the Actor network, three independent single-layer networks will fit the corresponding transformation formulas. The input layer dimension is M_h. The output dimensions are $M_a \cdot F$, $M_a \cdot F$, and F where F is the number of flows in the NF and M_a is the action dimension of the agent. The computational complexity of generating the normalization flow mapping formula is $\mathcal{O}\left(2M_h M_a F + M_h F\right)$. Finally, the actions output by the network need to be mapped. The mapping process is matrix multiplication. The computational complexity of this process is $\mathcal{O}\left(2FM_a\right)$.Therefore, ignoring the constant and smaller terms, the computational complexity of Normalizing Flow Actor Network is $\mathcal{O}_{ac}\left(M_o M_h + \mathcal{L}{M_h}^2 + M_h M_a F\right)$.

Critic network: Assuming the MLP part of the Critic network before integrating with the GAT has an input dimension of M_{oa}, hidden layer dimension of M_h, and output dimension is M_{pr}, the computational complexity of this process is $\mathcal{O}\left(M_{oa}M_h + M_hM_{pr}\right)$. For the GAT network, let the number of nodes be $\mathcal{J}$, each node's input feature dimension be M_{pi}, each node's output feature dimension be M_{po}, and the number of edges in the graph structure be $\mathcal{R}$. The computational complexity for the feature mapping of the nodes is $\mathcal{O}\left(\mathcal{J}M_{pi}M_{po}\right)$, and for the edge mapping to calculate the attention coefficients, it is $\mathcal{O}\left(2\mathcal{R}M_{po}\right)$. Therefore, the computational complexity of the GAT network is $\mathcal{O}\left(\mathcal{J}M_{pi}M_{po} + 2\mathcal{R}M_{po}\right)$. Ignoring the differences in input and output dimensions between GAT-O and GAT-T, as well as constants, the computational complexity for a GAT network with K attention heads is $\mathcal{O}\left(KM_{po}(\mathcal{J}M_{pi} + \mathcal{R})\right)$. After merging the outputs of both parts, the combined output enters a new MLP network. The input dimension is $M_{pr} + K\mathcal{J}M_{po}$, output dimension is 1, hidden layer dimension is M_h, and the number of hidden layers is $\mathcal{L}$, the computational complexity of this process is $\mathcal{O}\left(\left(M_{pr} + K\mathcal{J}M_{po}\right)M_h + \mathcal{L}{M_h}^2 + M_h\right)$. Therefore, Ignoring smaller terms and constants, the simplified computational complexity of Graph Attention Critic Network is $\mathcal{O}_{cr}\left(M_h\left(M_{oa} + \mathcal{L}M_h + M_{pr}\right) + KM_{po}(\mathcal{J}(M_{pi} + M_h) + \mathcal{R})\right)$.

Decentralized Execution: During the decentralized execution phase of the algorithm, only the Actor network is used. Assuming the total number of interaction steps with the environment is T and the number of agents is N, the computational complexity of the whole decentralized execution phase can be is $\mathcal{O}_{de} = TN\mathcal{O}_{ac}$.

Centralized Training: During centralized training phase, both the Actor and Critic networks are used. With the adoption of the double Q network method, the training process involves using the Actor network $2N$ times and the Critic network $4 + 2N$ times for each training, where N is the number of agents. Assuming the total number of steps is T and the batch size is D, ignoring the constants, the simplified computational complexity is $\mathcal{O}_{ct} = TDN\left(\mathcal{O}_{ac} + \mathcal{O}_{cr}\right)$.

Computational complexity comparison and analysis:

Furthermore, we compared the computational load of different algorithms based on the same experimental parameters, and the results calculated by whole interaction steps are shown in Table S2. From the perspective of decentralized execution, NAGC has an advantage. Although it adds normalizing flows to the Actor network, it removes a hidden layer from the Actor network. However, removing a hidden layer from other algorithms' Actor networks hampers effective learning convergence in our experiments. MAPPO and HAPPO require additional use of a Critic network during decentralized execution, increasing their computational load. In centralized training, the computational load varies significantly due to the different architectures of each algorithm. On-policy algorithms like MAPPO and HAPPO do not sample batches of data from an experience pool like off-policy algorithms but instead train using recently collected data and discard it afterward. Therefore, their computational load is roughly lower by a factor approximately equal to the batch size compared to off-policy algorithms. FACMAC's value decomposition architecture results in a smaller input dimension for the Critic network. NAGC and HASAC have relatively similar structures, but NAGC's GAT module increases the computational load during training. However, the increase is minor and brings significant performance improvements. In practical reinforcement learning applications, offline training times are generally long, with much of the time spent on simulation platform iterations. Therefore, the additional training time is acceptable, and we usually focus more on execution time during actual usage. In this regard, NAGC has a certain advantage. Based on repeated tests, we found that, the Actor network of NAGC only takes 2.158 ms to complete a decision-making for each agent, providing UAVs with high real-time decision-making capability.

Table S2
THE COMPUTATION LOAD OF ALL ALGORITHMS

Computation Load	NAGC	HASAC	FACMAC	HAPPO	MAPPO
Decentralized Execution	1.124×10^{10}	1.546×10^{10}	1.546×10^{10}	2.256×10^{10}	2.256×10^{10}
Centralized Training	4.045×10^{13}	3.450×10^{13}	2.647×10^{13}	1.437×10^{11}	1.128×10^{11}

S5: The center coordinates of all buildings

Please refer to the Table. S3

Table S3
THE CENTER COORDINATES OF ALL BUILDINGS

Coordinate (m)	Coordinate (m)	Coordinate (m)	Coordinate (m)
1: [375, 375]	2: [325, 125]	3: [175, 275]	4: [125, 75]
5: [-125, -125]	6: [-175, -375]	7: [-325, -125]	8: [-375, -425]
9: [375, -375]	10: [275, -125]	11: [125, -175]	12: [25, -325]
13: [-75, 425]	14: [-75, 125]	15: [-325, 375]	16: [-375, 125]

S6: The three kinds of evasion strategies

In this experiment, three distinct evasion strategies for the evader were established:

Random Move Strategy: The evader maneuvers at maximum speed, while ensuring that the yaw angular velocity at each moment adheres to a random value constrained by the UAV dynamics equation.

Repulsive Force Strategy: Under this strategy, the evader operates at maximum speed and seeks escape by navigating towards the direction of the resultant repulsive force. This force is derived from the collective contribution of all pursuers. Assuming the coordinate points of the three pursuers are denoted as $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$, respectively, and the coordinate point of the evader is $\mathbf{p}_e$. The repulsive forces exerted by the pursuers on the evader are represented as vectors $\mathbf{g}_1 = \mathbf{p}_e - \mathbf{p}_1$, $\mathbf{g}_2 = \mathbf{p}_e - \mathbf{p}_2$, and $\mathbf{g}_3 = \mathbf{p}_e - \mathbf{p}_3$. The weight w_g of these repulsive forces are inversely proportional to the distance between the pursuers and the evader, as follows:

$$\begin{aligned} w_{g1} : w_{g1} : w_{g1} &= 1/d_{1e} : 1/d_{2e} : 1/d_{3e} \\ w_{g1} + w_{g1} + w_{g1} &= 1 \end{aligned} \tag{S4}$$

This implies that the nearer a pursuer is to the evader, the stronger the repulsive force exerted by that pursuer. Consequently, the resultant repulsive force exerted by all pursuers on the evader is calculated as $\mathbf{g}_{jt} = w_{g1} \cdot \mathbf{g}_1 + w_{g2} \cdot \mathbf{g}_2 + w_{g3} \cdot \mathbf{g}_3$. The evader will move in the direction of the calculated repulsive force at the current moment. At the same time, it also needs to follow the maximum angular velocity constrainted by the UAV dynamics equation. If the yaw angle cannot achieve the repulsive force direction even with the maximum angular velocity, the evader will aim to rotate as closely as possible towards the direction.

Reinforcement Learning Strategy: Under this strategy, it is assumed that the evader possesses real-time awareness of the pursuers' overall information. Therefore, the evader's observation consists of two parts: (1) Evader's own state information: This includes the evader's coordinates $\mathbf{p}_e^t$, yaw angle ψ_e^t, velocity $\mathbf{v}_e^t$, and obstacle information $\boldsymbol{\lambda}_e^t$. (2) Pursuers' state information: This includes the relative coordinates $\Delta\mathbf{p}_{ei}^t$, distance d_{ei}^t, relative yaw angle $\Delta\psi_{ei}^t$, and azimuth angles ρ_{ei}^t of all pursuers, where $i \in \{1, 2, 3\}$. The reward for the evader comprises distance reward r_{e1}^t, yaw angle reward r_{e2}^t, and collision warning

Table S4
COMPARISON RESULTS OF FIVE ALGORITHMS IN EACH EVALUATION METRIC

Evasion Strategy	Algorithms	Reward	SCR (%)	CDR (%)	Mission Time (s)	Search Time (s)	Lost Time (s)
Random Move Strategy	NAGC	**53.59±1.70**	**88.80±4.66**	**1.60±0.08**	**71.74±9.74**	36.21±9.76	8.11±2.11
	HASAC	51.05±3.50	85.40±2.94	6.40±4.63	72.13±7.82	**33.98±3.53**	**4.24±1.30**
	FACMAC	49.99±2.08	82.00±2.83	8.40±3.88	79.94±4.32	36.55±5.19	8.31±2.81
	HAPPO	46.85±3.43	78.80±4.49	6.00±5.22	88.51±4.99	44.69±7.13	5.79±0.54
	MAPPO	48.06±4.02	80.40±6.86	6.40±2.56	87.53±9.49	51.09±15.53	13.84±3.37
Repulsive Force Strategy	NAGC	**56.93±5.66**	**80.80±5.60**	**8.00±1.56**	**98.47±7.53**	**47.41±8.65**	18.02±2.54
	HASAC	49.18±3.66	71.20±7.33	15.60±5.57	100.67±12.93	50.08±5.56	13.52±2.92
	FACMAC	51.27±4.80	67.20±6.52	12.00±4.73	110.72±9.94	54.91±8.27	13.78±1.66
	HAPPO	53.48±5.15	67.20±4.83	9.20±1.60	116.52±11.55	51.43±11.02	22.82±7.01
	MAPPO	44.74±6.23	62.80±6.88	8.00±2.19	128.70±3.90	75.86±8.77	**12.28±2.95**
Reinforcement Learning Strategy	NAGC	**53.84±4.41**	**79.20±7.33**	6.40±3.20	**86.11±5.60**	**43.26±6.78**	**9.05±2.09**
	HASAC	48.51±4.39	76.40±7.74	8.60±3.20	88.74±9.32	51.00±14.34	16.43±2.69
	FACMAC	45.75±8.90	66.00±12.71	16.00±7.16	109.94±11.43	60.14±16.02	14.92±3.92
	HAPPO	49.67±2.45	64.00±5.51	8.00±3.35	116.74±8.12	55.94±9.54	25.13±3.37
	MAPPO	46.94±7.22	64.40±6.12	**4.00±2.19**	114.70±10.40	57.85±4.93	21.16±8.42

reward r_{e3}^t. The calculation of r_{e1}^t depends on the relative distance between the evader and all pursuers. The calculation of r_{e1}^t adheres to the following formula:

$$r_{e1}^t = \begin{cases} \sum_{i=1}^{3} (d_{ei}^t / \mathcal{C}_7), \ d_{ei}^t \geq d_{ei}^{t-1} \\ -\sum_{i=1}^{3} (d_{ei}^t / \mathcal{C}_7), \ d_{ei}^t < d_{ei}^{t-1} \end{cases} \tag{S5}$$

where $\mathcal{C}_7$ is a constant.

The yaw angle reward r_{e2}^t serves as an incentive for the evader to maintain flight in a direction that keeps it distant from the pursuers. It can be computed using the following formula:

$$r_{e2}^t = \sum_{i=1}^{3} \mathcal{C}_8 \cos\left(\psi_e^t - \rho_{ei}^t\right) \tag{S6}$$

where $\mathcal{C}_8$ is a negative constant differ from $\mathcal{C}_5$. The computation process for collision warning reward r_{e3}^t is similar to the pursuers', except that r_{e3}^t utilizes the radar ray length obtained by the evader itself. Consequently, the evader's reward r_e^t can be got by adding three parts together.

The evasion UAV's action involve the acceleration and angular velocity, which are akin to those of the pursuers. However, the evader possesses a higher speed limit. The action serves as the output of the evader's policy. As a commonly used baseline with solid performance in continuous control, Deterministic Deep Policy Gradient (DDPG) is a classic single-agent DRL algorithm. We selected it as the training algorithm for the evader's reinforcement learning strategy. Following 5×10^5 steps of training, the trained neural network serves as the reinforcement learning strategy for the evader.

Among the three evasion strategies, in order to entirely avoid collisions, obstacle avoidance rule was established for the evader. If the evader approaches within 25 meters of an obstacle, it will maneuver to deviate from the obstacle's direction as much as possible. It will then maintain a safe distance of over 25 meters from the obstacle along that altered direction to ensure avoidance of collision. This rule is also used for preventing evader from leaving the designated airspace.

S7: Comparison results of five algorithms in each evaluation metric

Please refer to the Table. S4.

Table S5
GENERALIZATION RESULTS OF DIFFERENT ALGORITHMS - PART A

Scenarios	Algorithms	Reward	SCR (%)	CDR (%)	Mission Time (s)	Search Time (s)	Lost Time (s)
Scenario 1	NAGC	**50.39±0.50**	**74.40±2.40**	8.40±1.20	**94.19±4.44**	**55.17±3.57**	11.67±0.62
	HASAC	48.00±1.29	70.80±1.20	10.40±1.60	103.44±0.79	62.96±4.20	**9.88±0.71**
	FACMAC	47.53±1.80	70.40±3.20	13.20±1.20	104.90±2.12	60.38±3.38	12.00±0.34
	HAPPO	48.39±0.89	62.00±0.40	8.80±0.80	117.97±1.79	61.78±0.74	22.43±0.27
	MAPPO	48.70±1.60	65.60±0.80	**7.60±1.20**	117.64±1.30	60.48±0.63	21.02±1.16
Scenario 2	NAGC	**52.77±0.21**	**84.40±0.40**	**5.60±0.80**	**85.50±0.24**	**47.09±0.78**	**12.16±0.92**
	HASAC	48.61±2.00	70.40±5.60	10.00±2.80	99.93±7.39	57.29±7.28	15.46±3.88
	FACMAC	43.63±2.03	64.80±3.20	19.60±2.80	111.51±1.21	54.57±3.31	12.61±0.96
	HAPPO	44.62±1.40	51.60±0.40	15.20±4.00	132.57±2.31	68.04±0.23	33.79±2.75
	MAPPO	42.14±1.54	64.80±0.20	10.80±3.60	120.36±1.52	73.22±7.50	15.40±0.96
Scenario 3	NAGC	**50.37±0.30**	**76.80±0.80**	6.00±1.20	**91.97±0.90**	53.38±4.09	12.13±1.17
	HASAC	49.96±2.06	70.40±6.40	12.40±1.20	99.62±8.24	50.55±5.89	**11.28±0.23**
	FACMAC	46.18±1.09	65.60±4.80	17.20±2.00	114.81±4.98	52.74±3.17	15.56±1.54
	HAPPO	47.81±3.50	55.20±8.80	6.80±1.20	126.90±5.95	**48.64±2.38**	38.70±4.76
	MAPPO	50.18±1.38	74.40±3.20	**4.80±0.80**	107.93±3.92	54.42±4.23	17.77±1.87
Scenario 4	NAGC	**52.55±2.17**	**83.60±3.60**	**4.80±1.60**	**81.64±4.08**	**39.81±7.57**	**11.83±1.34**
	HASAC	49.90±5.08	75.20±7.20	10.00±5.20	91.66±8.10	48.62±6.36	12.37±1.65
	FACMAC	44.27±3.35	64.00±6.40	14.40±4.00	111.72±6.75	58.08±4.95	14.22±0.10
	HAPPO	57.87±0.24	48.80±4.00	17.60±2.40	135.90±5.48	47.88±0.68	34.75±0.89
	MAPPO	44.00±1.59	53.60±2.40	23.60±1.20	120.26±0.98	45.70±0.22	22.75±0.40
Scenario 5	NAGC	**62.99±0.98**	**82.00±1.20**	**7.60±2.80**	**96.23±0.40**	**34.37±2.90**	19.91±0.62
	HASAC	52.15±1.64	61.6±0.20	20.00±1.60	110.20±2.10	34.61±1.41	15.39±0.35
	FACMAC	44.05±0.25	68.00±0.80	12.80±2.40	98.20±3.39	47.96±3.49	**13.45±1.15**
	HAPPO	57.87±0.24	48.80±0.40	17.60±2.40	135.90±5.48	47.88±0.68	34.75±0.89
	MAPPO	44.00±1.59	53.60±2.40	23.60±1.20	120.26±0.98	45.70±0.22	22.75±0.40
Scenario 6	NAGC	**65.31±0.28**	**95.60±2.00**	**1.60±0.80**	**51.96±2.78**	**13.58±0.19**	**7.69±0.35**
	HASAC	63.22±1.05	73.60±1.60	12.40±1.20	99.72±0.68	22.46±2.02	12.89±0.84
	FACMAC	60.60±1.85	82.80±0.40	4.40±0.40	69.72±0.86	31.51±1.92	10.57±2.37
	HAPPO	52.20±1.52	54.40±0.80	9.20±1.20	134.79±2.60	45.87±5.32	34.22±4.35
	MAPPO	56.63±2.66	77.20±1.20	8.80±2.40	92.06±3.06	26.26±3.83	8.37±1.31

Table S6
GENERALIZATION RESULTS OF DIFFERENT ALGORITHMS - PART B

Scenarios	Algorithms	Metric 1	Metric 2 (s)	Scenes	Algorithms	Success Rate (%)	Mission Time (s)
Scenario 7	NAGC	**4.46±0.71**	**29.27±1.08**	*Scenario 8*	NAGC	**80.80±3.20**	**113.50±0.92**
	HASAC	4.37±0.70	30.90±1.08		HASAC	62.00±0.40	122.92±1.32
	FACMAC	3.86±0.94	36.72±2.14		FACMAC	61.20±2.80	128.79±1.62
	HAPPO	4.37±0.72	35.30±0.82		HAPPO	42.00±2.80	146.90±2.64
	MAPPO	3.54±0.98	40.01±1.17		MAPPO	80.40±0.40	116.30±1.99

S8: Generalization results of different algorithms

Generalization results from *Scenario 1 - 6*, please refer to the Table. S5.

Generalization results from *Scenario 7 - 8*, please refer to the Table. S6.

S9: The influence of GAT on motion trajectories

From the motion trajectories of NAGC and NGAT during the search phase. We observed that NAGC forms a more orderly circular motion trajectory, allowing for more efficient and continuous airspace search, whereas trajectory of NGAT appears more disordered and fluctuating, with limited spatial coordination among UAVs. Additionally, we calculated the trajectory smoothness for NAGC and NGAT. The trajectory smoothness metric uses the mean squared error between the points of the spline-fitted trajectory and the actual trajectory points, resulting in a trajectory smoothness metric of 0.530 for NAGC and 0.648 for NGAT. The fluctuation degree of NAGC's trajectory is significantly lower than that of NGAT, further validating that incorporating GAT brings more stable and orderly movement patterns for the agents.

S10: Pursuit-evasion scenarios under different kinds of environments

In Fig. S2, we depict the trajectories of two kinds of pursuit UAVs capturing the evasion UAV under NAGC algorithm. The environment is the same as training environment.

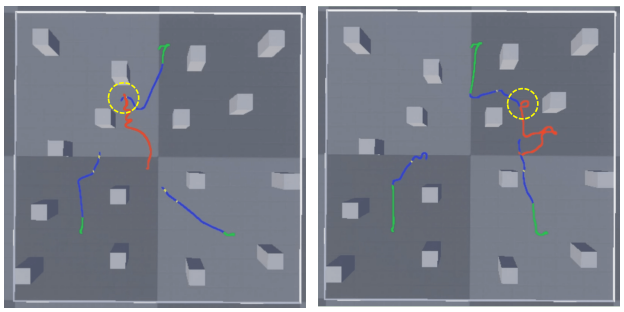

Figure S2. Trajectories of two kinds of pursuit UAVs capturing the evasion UAV under NAGC algorithm. We use the yellow circles to mark where the captures take place. The meanings of the different colored lines have already been explained in the main documant.

We tested the NAGC's performance under communication loss. A pursuit UAV located in the bottom left place of the map experienced communication disruptions with the other two pursuit UAVs between 60 to 120 seconds. During this disruption, status information could not be exchanged, and when the pursuers detected the evader, information about the evader could not be shared. We substituted the unavailable information with a zero input. Generalization testing of this scenario was conducted under the NAGC algorithm, as illustrated in Fig. S3. Note that we represent the motion trajectory of the pursuer under communication loss with **white lines**. The test results are as follows:

(1) When communication is lost, the disconnected UAV can still perform obstacle avoidance and search behaviors without exhibiting chaotic movement, marked by purple circles in Fig. S3a and S3b.

(2) In the absence of information from the disconnected UAV, the remaining two UAVs can continue their pursuit of the evader, marked by yellow circles in Fig. S3a and S3b.

(3) When the disconnected UAV detects the evader, it will attempt to capture it, even though it cannot share this information with the other pursuers, as shown in Fig. S3c.

(4) When all UAVs detect the evader, they will all execute the pursuit, as shown in Fig. S3d.

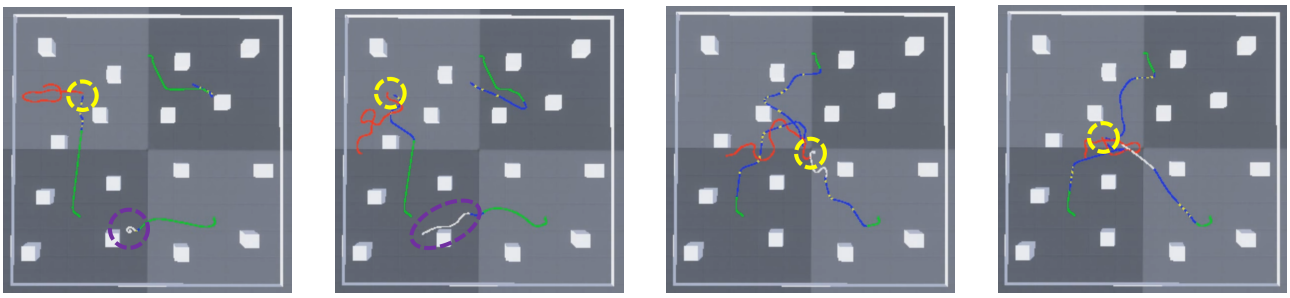

Figure S3. Scenario where one pursuer lose communication with other two. We use the red line to denote the trajectory of evader, use the white line to denote the trajectory of pursuer in losing communication. Note that the track only turns white when the communication is disconnected. We use yellow circle to denote the place of capture event, and use purple circle to highlight the behaviors of the pursuer in losing communication.

We also tested the pursuit-evasion in a very complex maze environment, as shown in Fig. S4. The NAGC algorithm we designed can also achieve the successful capture in this environment, though the evader sometimes hide in remote corner.

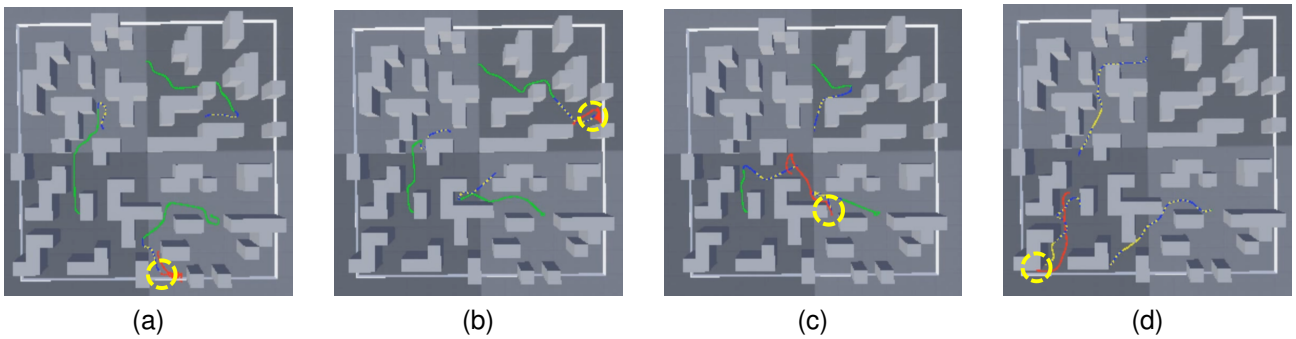

Figure S4. Scenario where pursuers capture the evader in maze. We use the red line to denote the trajectory of evader, use yellow circle to denote the place of capture event.

We also set up a scenario where the evader is moving clockwise at a high speed on the edge of 5 meters close to the building (length and width are 200 meters). NAGC algorithm can effectively generalize to this scenario and capture the evader, as shown in Fig. S5.

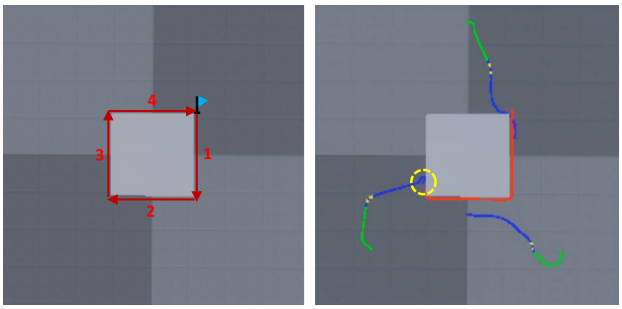

Figure S5. New scenario where evader moves close to building. In (a), the blue flag is the start point of evader, the red arrow line is the moving trajectory of evader, and numbers mean the order of movements. In (b), we use the red line to denote the trajectory of evader, and lines in other colors to denote the trajectory of pursuers, and we use yellow circle to denote the place of capture event.