SCIENCE CHINA
Information Sciences

• Supplementary File •

DPCN: enabling efficient multi-agent path finding via decentralized-planning-centralized-negotiation Paradigm

Qidong Liu[1], Xiaowen Li[1*], Chuang Zhan[1], Jie Zhang[2], Cheng Long[2] & Mingliang Xu[1]

[1]*School of Computer Science and Artificial Intelligence, Zhengzhou University, Zhengzhou 450000, China*
[2]*College of Computing and Data Science, Nanyang Technological University, Singapore 639798, Singapore*

Appendix A Planning Phase

The design of the Intention Network is inspired by SCRIMP [1], which similarly leverages structured representations of local observations for intention prediction. We consider a partially observable grid world environment, where each agent (AGV) has a limited field of view (FOV) of 3×3 centered around itself. As for agent i, the local observation $\mathbf{o_i^t}$ consists of eight binary matrices that respectively represent four heuristic maps as proposed in [2], the locations of agents, the agent's own goal location (if it falls within the FOV), the projected positions of other observable agents' goals and nearby obstacles.

To clarify the construction of the heuristic maps, we provide an illustrative example in Figure A1. Each heuristic map corresponds to one of the four primitive actions: Up, Down, Left, and Right. A cell in a given map is marked as 1 if taking the corresponding action from that position brings the agent closer to its goal (based on shortest path distance), and 0 otherwise. Since multiple shortest paths may exist, multiple actions can be marked as beneficial, preserving exploration flexibility. For instance, in the example shown, the agent (red) aims to reach its goal (green) located to the east. The Right channel activates all positions where moving right reduces the distance to the goal; similarly, the Up and Down channels activate positions where those actions are progress-making. Obstacles are marked in black. This local guidance is computed using a single-agent pathfinding algorithm (e.g., A*) on the local map, without requiring centralized planning.

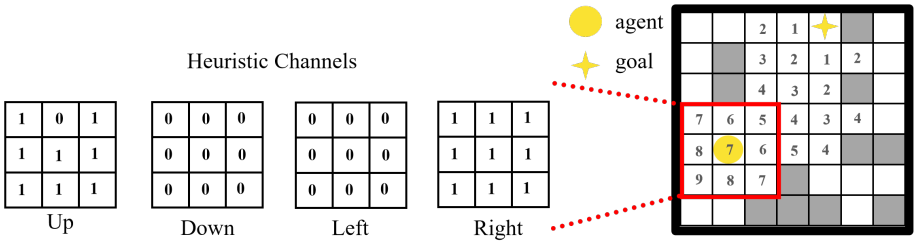

Figure A1 Heuristic channels for the yellow circle agent, with 3×3 FOV. The yellow four-pointed star is the goal, and the gray blocks are obstacles. Four channels indicate whether agent get closer to its goal by taking a certain action at those locations.

In addition, a vector $\mathbf{v_i^t}$ records target information, rewards, and historical action, which assists the model in making decisions. When agents approach the boundaries of the grid world, we introduce virtual obstacles at all positions outside the world's edges to ensure a consistent and comprehensive representation of the environment.

For the individual agent, local observation is used to approximate the state of the environment, and the intention network selects an intention from the set of actions $\mathcal{A}$. The reward structure is shown in Table A1. We adopt the neural network architecture proposed in [1] as the foundation for this learning process. Specifically, as shown in Figure A2, for agent i at timestep t, the intention network f^P encodes the observation $\mathbf{o_i^t}$, the vector $\mathbf{v_i^t}$ and the messages from other agents $\mathbf{m^{t-1}} = (m_1^{t-1}, \ldots, m_n^{t-1})$ to c_i^t which represents the local state of agent i for making decisions. This process is expressed by the following equation:

$$c_i^t = f^P\left(\mathbf{o_i^t}, \mathbf{v_i^t}, \mathbf{m^{t-1}}\right). \tag{A1}$$

Finally, c_i^t is fed into two different linear transformation layers f^{L_1} and f^{L_2} to obtain the message m_i^t and intention $\tilde{a}_i^t$,

$$\begin{aligned} m_i^t &= f^{L_1}(c_i^t), \\ \tilde{a}_i^t &= f^{L_2}(c_i^t). \end{aligned} \tag{A2}$$

The intention network is trained by Proximal Policy Optimization (PPO) [3]. During training, the predictions of state values are updated by minimizing the temporal-difference (TD) [4] error between the predicted value and the discounted returns.

* Corresponding author (email: xiaowli@gs.zzu.edu.cn)

Table A1 Reward Function Design

Actions	Rewards
Move (Up/Down/Left/Right)	-0.3
Stay (On goal, Away goal)	0.0, -0.3
Collision	-2
Block	-1

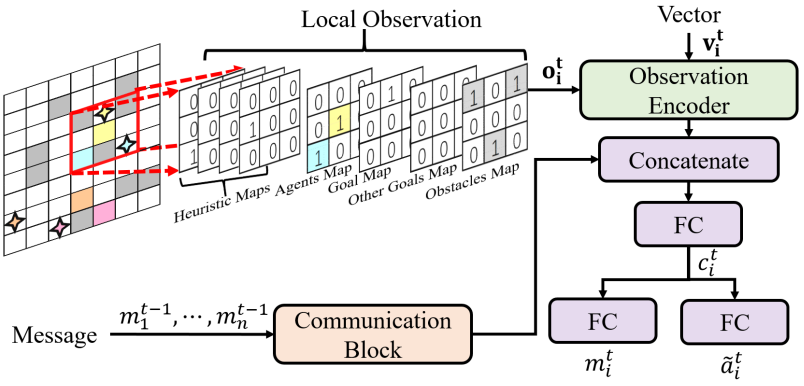

Figure A2 Framework of intention network f^P. It uses local observation information and messages from other agents to determine the intention $\tilde{a}_i^t$.

Appendix B Negotiation Phase

At each timestep, we initially categorize agents whose intentions clash into different conflict sets. Specifically, we simulate the execution of all agents' intentions on an image map reflecting the current state. Define d_i^t to represent the position of the agent i before executing the intention $\tilde{a}_i^t$, and $\tilde{d}_i^{t+1}$ to represent the position of the agent i after executing intention on the image map. Let $\mathbf{\Omega}^t = \{\Omega_1^t, \cdots, \Omega_{K_t}^t\}$ denote the set of all conflict sets on the image map. Thus, at timestep t, for the k-th conflict set $\Omega_k^t \subset \mathcal{I}$, $\forall i \in \Omega_k^t$, we have

$$\begin{cases} \exists_{j \in \Omega_k^t \wedge j \neq i} \left(\tilde{d}_i^{t+1} = \tilde{d}_j^{t+1} \right) \\ \text{or} \\ \exists_{j \in \Omega_k^t \wedge j \neq i} \left[\left(\tilde{d}_i^{t+1} = d_j^t \right) \wedge \left(\tilde{d}_j^{t+1} = d_i^t \right) \right] \end{cases} \tag{B1}$$

In other words, if multiple agents try to reach the same cell, this is identified as a vertex conflict, and the conflict set includes all agents attempting to reach that cell. If two agents intend to swap positions, this is labeled as an edge conflict. Since in our setting, the map is 4-neighborhood, the number of agents in a conflict set does not exceed 5.

During the negotiation phase, the objective is to select a winner from the agents within the conflict set. In this phase, the conflict set is modeled as a super-agent, and the action set comprises the agents within this super-agent. It is worth noting that the elements in different super-agent vary, leading to inconsistent action sets (Challenge 1). To address this challenge, drawing inspiration from Pointer Network [5], which outputs elements directly originated from the input sequence rather than from a predefined vocabulary, we design a learnable network named special-edition pointer network (PNSE). It dynamically establishes a prioritization framework for decision-making within each super-agent.

We select a winner for each conflict set. At current timestep, the non-conflicting agents and the selected winners have the authority to execute their intentions. While the non-winning agents should remain stationary or resample their actions. Specifically, during training, the non-winning agents are encouraged to remain stationary. This strategy prevents the introduction of additional conflicts through switching actions and focuses on resolving the identified conflicts, thereby stabilizing the training process. During testing, to quickly address complex tasks, non-winning agents are allowed to reselect the alternative executable actions. Since the super-agent can independently resolve conflicts based on the PNSE, any new conflicts arising from switching actions can be swiftly and effectively resolved. Once no new conflicts arise, the program would move to the next timestep.

Appendix C Training of PNSE

For training this reinforcement learning task, we model the conflict set Ω_k as a super-agent and employ the policy gradient method to maximize the objective function. At this current timestep t, we consider a game involving K super-agents, each with policies parameterized by $\theta = \{\theta_1, \cdots, \theta_K\}$, and let $\pi = \{\pi_1, \cdots, \pi_K\}$ denote the set of all agents' policies. The gradient of the expected return for super-agent k, $J(\theta_k) = \mathbb{E}[R_k]$, can be expressed as:

$$\nabla_{\theta_k} J(\theta_k) = \mathbb{E}_{v_k \sim \pi_k} \left[\nabla_{\theta_k} log \pi_k(v_k|s_k) Q_k^{\pi_k}(v_k, s_k) \right]. \tag{C1}$$

Here, $\pi_k(v_k|s_k)$ denotes the policy of the PNSE, where s_k represents the state, and v_k represents the action of the super-agent, specially, the winner selected by π_k from Ω_k. The action-value function $Q_k^{\pi_k}(v_k, s_k)$ takes the action v_k and state s_k of super-agent k as input,

and outputs the Q-value for super-agent k. However, the dynamic nature of the super-agents, makes it hard to learn an approximation of the action-value function $Q_k^{\pi_k}$ by temporal-difference learning [4], or to assign a specific policy network π_k for each super-agent k (Challenge 2). To address this challenge, we draw upon the REINFORCE algorithm [6] to approximate $Q_k^{\pi_k}$, and a shared policy network for all agents, denoted as π. Therefore, Equation (C1) can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{v_k \sim \pi} \left[\nabla_\theta log\pi(v_k|s_k) Q_k^\pi(v_k, s_k) \right]. \tag{C2}$$

At timestep t, the Q-value of the whole multi-agent system can be represented as the sum of the discount return of all agents in the environment:

$$Q((v_1, \cdots, v_K), (s_1, \cdots, s_K)) \approx \sum_{t'=t}^{T} \left(\gamma^{t'-t} r_{all}^{t'} \right), \tag{C3}$$

where γ is a discount factor, and T is the time horizon. Furthermore, $r_{all} = \sum_{i=1}^{n} r_i$ is the sum of all agents' rewards in the environment.

According to the assumption of VDN [7], the joint action-value function for the system can be additively decomposed into value functions across agents. Therefore, at timestep t, the Q-value of the whole multi-agent system can be represented as the sum of the Q-value of all super-agents, i.e.,

$$Q((v_1, \cdots, v_K), (s_1, \cdots, s_K)) \approx \sum_{i=1}^{K} Q_k^\pi(v_k, s_k). \tag{C4}$$

According to Equation (C3) and Equation (C4), we have

$$\sum_{i=1}^{K} Q_k^\pi(v_k, s_k) \approx \sum_{t'=t}^{T} \left(\gamma^{t'-t} r_{all}^{t'} \right). \tag{C5}$$

We suppose $Q_x^\pi = Q_y^\pi, \forall x, y \in \{1, \cdots, K\}$, then Equation (C2) can be rewritten as follow:

$$\nabla_\theta J(\theta) = \mathbb{E}_{v_k \sim \pi} \left[\nabla_\theta log\pi(v_k|s_k) \frac{1}{K} \sum_{t'=t}^{T} \left(\gamma^{t'-t} r_{all}^{t'} \right) \right]. \tag{C6}$$

This approximation effectively serves as a shared baseline across super-agents, encouraging balanced contribution and stabilizing training.

Finally, the parameters θ can be updated using the gradient ascent rule $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$, where α is the learning rate. Although the above mathematical proof is not entirely rigorous, extensive experimental results demonstrate that this method can effectively work and the trained model can efficiently solve MAPF problem.

Appendix D Experiment

Appendix D.1 Experiment Settings

Our models are trained and tested in classic MAPF environment. During training, the experimental setup is consistent with some other algorithms [8] [1] [9]. The number of agents is consistently fixed at 8. The map size is randomly selected from 10, 25, or 40, with the obstacle density sampled from a triangular distribution ranging from 0% to 50% to simulate various levels of complexity. The initial positions and target positions of the agents are also randomly placed. To ensure the connectivity of the environment, a check is conducted after randomly placing the obstacles, and some of them are appropriately removed to ensure that all problem instance are solvable. We evaluate the trained model across various environmental settings. Notably, the maximum timestep is limited to 256. All experiments were conducted on the same computer equipped with Intel i9-14900k CPU and an NVIDIA GeForce RTX A6000 GPU.

Given the different objectives of the intention network and the PNSE, it is difficult to train them concurrently. Moreover, they are mutually dependent, where the performance of one directly influences the other. Therefore, training them alternately results in convergence challenges for both networks. Consequently, we initially train the intention network independently, and the agent with the maximum ID in the conflict set is selected as the winner. After a period of training, the decision-making ability of the intention network is improved. Subsequently, keeping the parameters of the intention network fixed, we proceed to train the PNSE to focus on enhancing its ability in resolving conflicts. During the experimental process, both networks were trained for 30 million timesteps and successfully converged. The parameters of the training phase are shown in Table D1.

Appendix D.2 Comparison with Other Planners

We compare DPCN with three state-of-the-art RL-based planners, in which the observations of agents are limited. DCC [10] is equipped with a selective communication mechanism to select effective communication targets, and the FOV size is 9×9. PICO [9] is based on a hierarchical communication mechanism, and makes decisions hierarchically based on implicit priorities, where lower-level agents avoid actions conflicting with higher-level agents. Its FOV size is 11×11. SCRIMP [1] employs Transformer architecture to aggregate information from other agents, and combines heuristic rules to establish agents' priorities for resolving conflicts. Its FOV size is 3×3. Their hyperparameters are set according to their official codes or papers. The larger FOV can increase the information used by agents, but also increases the computational burden. However, the smaller FOV imposes higher demands on the algorithm's performance. To reduce the computational burden while demonstrating the superior performance, we set the FOV of DPCN to 3×3. Additionally, we evaluate two centralized planners. The one is ODrM* [11], which coordinates all agents via a central controller. The other is

Table D1 Parameter Settings for DPCN during Training.

Name	Value
Num agent	8
Max timestep for each episode	256
FOV size	3
World size	[10, 40]
Obstacle density	[0%, 50%]
Optimizer	Adam
Learning Rate	1e-5
Discount factor	0.95
Steps for intention network	3e7
Steps for PNSE	3e7
Parallel envs	16

BALANCE [12], which is an LNS-based method, and dynamically adjusts destroy heuristics and neighborhood sizes through a bi-level multi-armed bandit scheme.

The performance of these planners is evaluated from four aspects. Success rate (**SR**) measures the planner's ability to successfully resolve the problem within the predefined timesteps. Execution time (**ET**) measures the time required to resolve the instance, which includes the time for planning and negotiation, reflecting its performance and responsiveness in real time. Episode length (**EL**) measures the timesteps required to successfully solve the instance, reflecting the coordination of the RL-based planners. Total move (**TM**) [9] represents the total distance moved by all agents, specifically referring to the non-stationary actions taken by the agents. Following the experimental setup by Wang et al. [1], we test those planners on different environmental settings, where each map size is equipped with different team scales and three static obstacle densities of 0%, 15% and 30%. For maps smaller than 100×100, the predefined timestep of RL-based algorithms is limited to 256. But for the map size reached 100×100, the predefined timestep is increased to 512. The predefined time of ODrM* is set to 5 minutes. For each environment setting, each planner is required to solve 200 randomly generated instances. SR and ET are calculated as the mean values from these 200 instances, while EL and TM are averaged exclusively over the successfully completed instances.

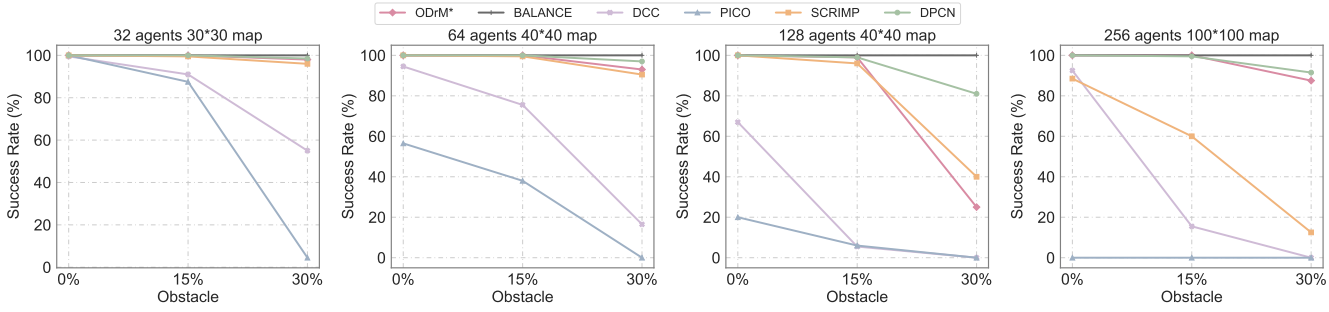

Figure D1 Success rate of different algorithms. The x-axis denotes the obstacle ratio in the environment, and the y-axis represents the success rate. Higher success rates indicate that the algorithm is more effective in guiding agents to reach their goals under crowded or obstacle-rich settings.

The result of SR is shown in Figure D1. Among all planners, the performance of DPCN is outstanding. ODrM*, by utilizing the centralized planner to globally coordinate all agents, performs excellently when agents are sparsely distributed. However, as the distribution of agents becomes denser, the probability of conflicts increases, which imposes a heavier computational burden on the central controller, leading to lower SR for ODrM* in scenarios with dense obstacles and agents. BALANCE consistently achieves high SRs, owing to its unique solution strategy. It first generates an initial feasible solution by leveraging several classical centralized planning methods, and then iteratively refines this solution. In other words, the primary objective of BALANCE is not merely to find a feasible solution, but to progressively improve it toward near-optimality. In contrast, other algorithms are primarily designed to obtain a feasible solution that satisfies the constraints. As the complexity of environment increases, the SR of SCRIMP, DCC, and PICO all decline. Specifically, SCRIMP employs heuristic rules to determine the priority of agents. However, rules struggle to deal with complex scenarios due to increased numbers and complexities of conflicts, which affects SR in environments with large-scale teams. DCC adopts a blunt conflict handling strategy involving remaining conflicting agents stationary, which leads to prolonged stagnation in high-density environments (e.g. 30% obstacle density), thereby preventing them from effectively completing tasks. PICO introduces a hierarchical decision-making mechanism that is theoretically feasible. However, in complex scenarios, low-level agents may not be able to execute optimal actions continuously, making it challenging to complete tasks. In contrast, although agents in DPCN make decisions independently, with effective coordination mechanisms and superior conflict resolution capabilities, the DPCN maintains a high SR in various scenarios. Notably, in environments with high obstacle density, DPCN not only significantly outperforms other RL-based planners but also surpasses ODrM*, further demonstrating the superior performance of PNSE in conflict resolution.

The results of the ET are shown in Figure D2. It is worth noting that, due to the different termination conditions of ODrM* and BALANCE with other RL-based planners, we don't report their ET. The hierarchical decision-making mechanism of PICO results in a

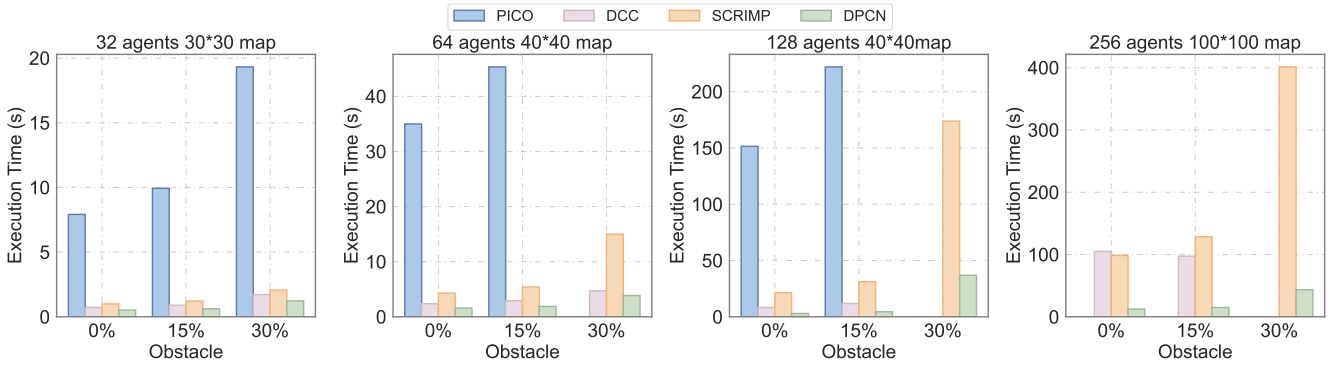

Figure D2 Execution time of different algorithms.The x-axis denotes the obstacle ratio in the environment, and the y-axis shows the execution time. A smaller execution time indicates higher efficiency, while a missing bar means the planner fails to complete the tasks.

longer decision time at each timestep, thereby increasing ET. In contrast, the blunt conflict handling strategy of DCC enables agents to make decisions quickly, thereby reducing the ET. Both SCRIMP and DPCN handle conflicts based on priority. However, SCRIMP relies on heuristic rules to calculate priorities, which significantly increases ET in complex scenarios. In contrast, DPCN uses the PNSE to calculate priority, which reduces time overhead. In addition, the experimental results show that as problems become more complex, the ET of DPCN is significantly better than that of DCC. Although DPCN may not make decisions as quickly as DCC at each timestep, its efficient conflict handling strategy accelerates the overall completion of tasks, resulting in DPCN demonstrating optimal performance in ET.

The results of EL and TM are shown in Table D2 and Table D3, respectively, reflecting the coordination capabilities of different algorithms and the quality of the obtained paths. As mentioned above, DCC and PICO lack effective conflict handling mechanisms, leading to increased timestep and movement costs to complete the tasks. When dealing with simple problems, the performance of SCRIMP and DPCN is comparable. In scenarios with 32 agents, SCRIMP even slightly outperforms DPCN in TM, as heuristic rules can effectively handle conflicts in simple scenarios. However, as the number of agents grows and the complexity of scenarios increases, the advantages of DPCN become increasingly apparent. This is because heuristic rules struggle to effectively handle complex conflict situations and may even lead to deadlocks due to inappropriate adjustment of actions. In contrast, DPCN benefits from its centralized negotiation mechanism which uses the trained PNSE to efficiently resolve conflicts based on current environment state. This learning-based conflict resolution mechanism enhances the coordination ability of DPCN, thereby enabling it to complete tasks quickly and reduce unnecessary movements. As a result, DPCN demonstrates superior performance in complex environments. Notably, in certain scenarios, DPCN outperforms both the centralized planner ODrM* and BALANCE in terms of path length (EL) and task completion time (TM). ODrM* is specifically designed to optimize EL and can compute bounded-optimal solutions, while BALANCE aims to iteratively converge toward near-optimality. Nevertheless, DPCN demonstrates superior performance in several cases. Under low obstacle densities (0% and 15%), its performance is comparable to that of ODrM* and closely approaches that of BALANCE, thanks to its effective conflict resolution mechanism. In high-density scenarios (30%), ODrM* and BALANCE slightly outperform DPCN due to their centralized coordination; however, DPCN still achieves the best performance among all reinforcement learning-based methods. This clearly validates the effectiveness and robustness of DPCN.

Table D2 Episode length and standard deviation of different algorithms. "-" represents unavailable data, as the planner cannot complete the tasks. The results of the best-performing RL-based algorithms in each map setting are highlighted in **bold**.

Methods	32 agents in 30 × 30 map			64 agents in 40 × 40 map			128 agents in 40 × 40 map			256 agents in 100 × 100 map		
	0%	15%	30%	0%	15%	30%	0%	15%	30%	0%	15%	30%
ODrM*	43.32 ±6.42	43.72 ±6.51	55.20 ±9.36	60.60 ±5.61	61.82 ±5.91	74.33 ±	65.28 ±4.59	66.64 ±6.2	81.04 ±9.18	197.85 ±9.62	201.68 ±13.29	212.16 ±14.47
BALANCE	40.08 ±4.15	45.21 ±4.87	61.33 ±11.84	61.20 ±5.54	57.33 ±5.7	81.46 ±14.18	70.57 ±6.99	63.86 ±7.1	106.80 ±27.19	165.85 ±9.25	166.33 ±8.66	178.87 ±11.13
DCC	46.79 ±7.73	55.12 ±26.15	86.24 ±33.84	70.22 ±19.68	84.05 ±28.61	131.48 ±39.93	101.41 ±33.93	134.00 ±41.41	-	178.15 ±16.76	205.84 ±23.95	-
PICO	55.36 ±9.86	66.91 ±18.54	106.22 ±30.2	116.33 ±22.5	142.62 ±25.21	-	159.88 ±39.78	203.33 ±37.89	-	-	-	-
SCRIMP	42.74 ±5.52	44.79 ±5.36	63.79 ±20.29	61.90 ±6.4	65.48 ±13.68	91.70 ±47.73	67.22 ±5.37	75.56 ±9.41	150.66 ±92.74	295.01 ±79.86	265.21 ±66.1	345.28 ±84.46
DPCN	**41.60** ±4.64	**44.32** ±5.02	**58.98** ±12.45	**59.49** ±5.71	**62.04** ±8.44	**84.56** ±20.93	**63.91** ±4.46	**67.09** ±5.19	**121.50** ±36.24	**164.80** ±9.57	**169.42** ±22.33	**201.52** ±43.84

Table D3 Total move and standard deviation of different algorithms. "-" represents unavailable data, as the planner cannot complete the tasks. The results of the best-performing RL-based algorithms in each map setting are highlighted in **bold**.

Methods	32 agents in 30 × 30 map			64 agents in 40 × 40 map			128 agents in 40 × 40 map			256 agents in 100 × 100 map		
	0%	15%	30%	0%	15%	30%	0%	15%	30%	0%	15%	30%
ODrM*	712.94 ±76.71	735.61 ±72.75	933.40 ±125.24	1867.80 ±110.61	1943.51 ±108.56	2386.06 ±	3802.60 ±155.79	3995.41 ±175.35	4904.58 ±224.07	21093.51 ±548.89	21679.13 ±544.88	24569.32 ±658.85
BALANCE	586.79 ±40.38	659.64 ±61.15	985.71 ±148.14	1786.69 ±82.8	1886.96 ±99.58	2570.97 ±267.4	3737.03 ±42.98	4019.61 ±187.19	6209.21 ±913.35	18455.12 ±695.14	19335.38 ±583.95	22848.38 ±853.95
DCC	686.32 ±59.44	741.86 ±78.28	974.24 ±178.32	1854.90 ±126.54	2015.78 ±160.13	2522.77 ±292.3	4174.20 ±211.6	4781.19 ±302.02	-	17939.63 ±604.07	19155.70 ±600.54	-
PICO	748.40 ±73.54	838.80 ±104.2	1056.00 ±206.73)	2135.32 ±149.87	2415.76 ±210.99	-	4753.16 ±296.35	6223.18 ±398.91	-	-	-	-
SCRIMP	**650.39** ±61.24	**691.56** ±54.88	**931.01** ±177.39	1750.72 ±115.62	1858.84 ±118.37	2542.16 ±508.96	3608.58 ±152.94	3925.62 ±182.6	6891.81 ±1469.06	17950.78 ±601.94	18254.86 ±497.43	22820.11 ±1275.64
DPCN	**650.67** ±57.39	**692.81** ±59.82	**941.55** ±113	**1748.93** ±107.09	**1854.17** ±110.27	**2456.18** ±278.23	**3606.90** ±152.48	**3916.51** ±179.28	**6016.00** ±890	**17367.72** ±536.24	**17987** ±554.14	**21763.91** ±914.58

Appendix D.3 Generalization of the MAPF Benchmark

To evaluate the generalization capability of DPCN across diverse structured map, we also evaluates its performance on the MAPF benchmark [13], with each map containing 25 problem instances. We selected three representative structured maps. The first map is a multi-room environment, with the size of 32 × 32, and the size of the room is 3 × 3. The second map is a chantry-like environment, with the size of 162 × 141. The third map represents a complex warehouse environment, with the size of 170 × 84, where the aisle width between shelves is only 2 cells. We select ODrM* and SCRIMP as the benchmark algorithms for comparison, as prior experimental results indicate that their generalization ability and robustness surpass those of DCC and PICO. On the first map, the predefined timestep for RL-based algorithms is set to 256. However, for the other two maps, which are of a larger size, this value is increased to 1024.

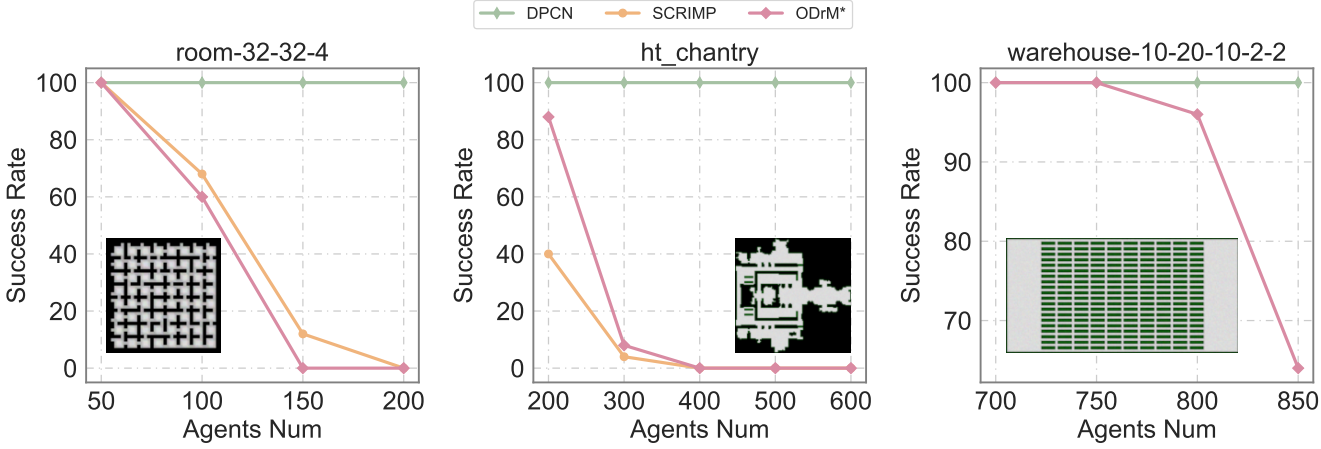

Figure D3 The SR of all algorithms in MAPF benchmark. The x-axis denotes the number of agents, and the y-axis represents the success rate. A higher success rate indicates better scalability. Results show that as the number of agents increases, overall success rates drop, and on the third map, SCRIMP fails to handle large-scale teams.

The results of the SR are shown in the Figure D3. In the multi-room scenario, although the map is connected, the complex distribution of obstacles demands algorithms with strong coordination capabilities. Due to frequent conflicts among agents, the central planner of ODrM* faces the excessive computational burden, resulting in poor performance in coordinating multi-agents. Its SR drops to 0% as the team size increases to 150. SCRIMP coordinates conflicts based on the priority which is calculates by heuristic rules, achieving a slightly higher success rate than ODrM*. DPCN, benefiting from superior coordination and conflict resolution, enables 200 agents to successfully complete tasks in this highly structured and complex environment. In the chantry scenario, we progressively increase the team scale until the GPU memory is insufficient to support further computations. The results indicate that ODrM* and SCRIMP struggle with large-scale tasks, and their SR drops to 0% when the team scale reaches 400 agents. In contrast, DPCN demonstrates exceptional performance, successfully completing tasks involving 600 agents. In the warehouse environment, large-scale teams place a significant computational burden on SCRIMP's heuristic-based conflict resolution mechanism, hindering its ability to handle large-scale teams effectively. Given that the map size of the warehouse scenario is larger than the other two scenarios, the agents are relatively sparsely distributed when the numbers are equal, allowing ODrM* to perform better on this type of map, maintaining a 100% SR even with 750 agents. In contrast, DPCN can successfully coordinate tasks involving 850 agents. Overall, the experimental results on the

benchmark maps demonstrate that DPCN has excellent generalization capabilities, can handle various complex structured environments, and efficiently coordinates large-scale teams through its effective conflict resolution capabilities.

Appendix D.4 Ablation Results

To further analyze the importance and effectiveness of PNSE in our proposed model, we conduct ablation studies. Specifically, we remove the PNSE, creating the variant "DPCN-without-PNSE", which selects the agent with the largest ID within the conflict set as the winner. Additionally, we design a variant to get agents' priorities based on heuristic rule, denoted as "DPCN-with-HR", which divides the number of agents in its FOV by the Euclidean distance from its target location [14]. Aside from the mechanisms for selecting the winner, all other aspects of both variants remain consistent with the original DPCN.

The performance of these three methods is shown in Figure D4. When faced with small-scale teams (8 and 16 agents), all three methods perform well. However, as the team scale increases, "DPCN-with-HR" outperforms "DPCN-without-PNSE", demonstrating the superiority of the dynamic priority mechanism over the fixed-priority mechanism, as the latter may lead to deadlock situations. DPCN outperforms both variants, highlighting the effectiveness and robustness of the learning-based priority. Although "DPCN-with-HR" employs dynamic priorities, it only considers limited factors, leading to suboptimal performance in tasks with large-scale teams. DPCN's EL performance is also superior to both variants, indicating that effective conflict resolution not only enhances the planner's ability to complete tasks but also improves the quality of the solutions.

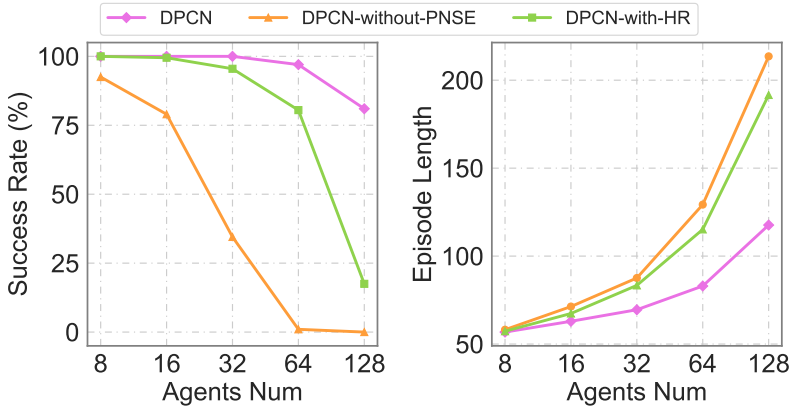

Figure D4 The SR and EL in ablation study for PNSE. A higher SR indicates better task completion, while a lower EL reflects more efficient paths. The map size is 40×40, and the static obstacle density is 30%.

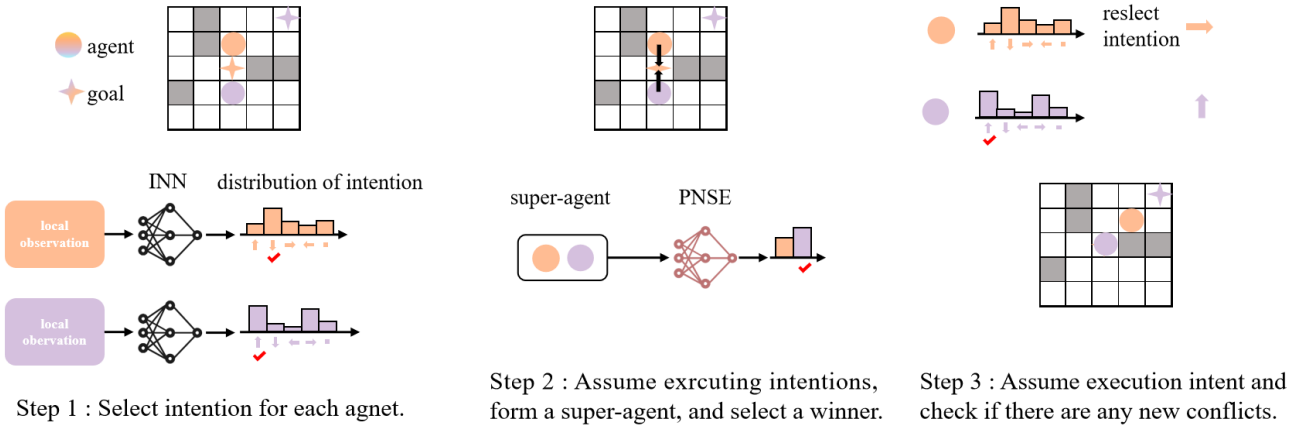

Figure D5 A case study illustrating the intention negotiation process in PNSE. Agents (circles) select initial intentions based on local observations via the INN (left). Conflicting agents are grouped into a super-agent, which uses the PNSE network to select a "winner" (middle). The winner's intention is preserved, while the other resample new intention. The final conflict-free intentions are executed synchronously (right).

Appendix D.5 Case Study

Figure D5 illustrates a case study, where circles represent agents, four-pointed stars denote goal positions, and different colors distinguish among agents. Initially, each agent independently selects an initial intention based on its local observation through the INN network. However, at this stage, agents do not fully account for the potential impact of their intentions on other agents or the overall environment.

Subsequently, the system simulates the execution of these initial intentions and identifies any resulting conflicts. Agents involved in conflicts are dynamically grouped into a "super-agent." The super-agent then employs the PNSE network to select one member as the "winner", whose intention is preserved. The non-winners resample new intentions according to their intention probability distributions. After all agents have adjusted their intentions, the system checks again for conflicts. If no conflicts remain, the final intentions are adopted as the actual actions. Once all agents have determined their actions, they execute them synchronously in the environment and proceed to the next time step.

References

1 Wang Y, Xiang B, Huang S, et al. Scrimp: Scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding. In: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, 2023. 2598–2600
2 Ma Z, Luo Y, Ma H. Distributed heuristic multi-agent path finding with communication. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021. 8699–8705
3 Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv: Learning, 2017
4 Sutton R S. Learning to predict by the methods of temporal differences. Mach. Learn., 1988, 3: 9–44
5 Vinyals O, Fortunato M, Jaitly N. Pointer networks. Advances in neural information processing systems, 2015, 28
6 Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 1992: 229–256
7 Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems2018, AAMAS'18. 2085–2087
8 Sartoretti G, Kerr J, Shi Y, et al. Primal: Pathfinding via reinforcement and imitation multi-agent learning. IEEE Robotics and Automation Letters, 2019, 4: 2378–2385
9 Li W, Chen H, Jin B, et al. Multi-agent path finding with prioritized communication learning. In: 2022 International Conference on Robotics and Automation (ICRA). IEEE2022. 10695–10701
10 Ma Z, Luo Y, Pan J. Learning selective communication for multi-agent path finding. IEEE Robotics and Automation Letters, 2021, 7: 1455–1462
11 Ferner C, Wagner G, Choset H. Odrm*: Optimal multirobot path planning in low-dimensional search spaces. In: 2013 IEEE International Conference on Robotics and Automation, 2013. 3854–3859
12 Phan T, Huang T, Dilkina B, et al. Adaptive anytime multi-agent path finding using bandit-based large neighborhood search. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2024, volume 38. 17514–17522
13 Stern R, Sturtevant N R, Felner A, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. Symposium on Combinatorial Search (SoCS), 2019: 151–158
14 Gao J, Li Y, Ye Z, et al. PCE: Multi-agent path finding via priority-aware communication & experience learning. IEEE Transactions on Intelligent Vehicles, 2024: 1–13