PACT: PREDICTIVE ACTION COORDINATION WITH TRAFFIC AWARENESS FOR LIFELONG MAPF

Anonymous authors
Paper under double-blind review

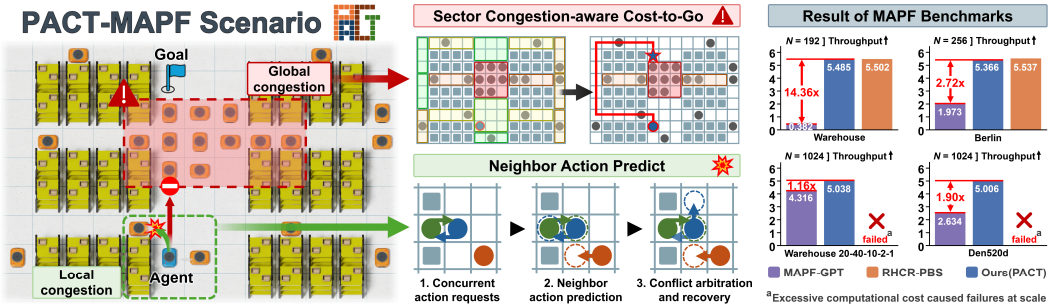


Figure 1: Overview of PACT. Neighbor Action Predictor (NAP) anticipates higher-priority neighbors' actions for confidence- and priority-guided local conflict resolution, while sector-level Traffic Guidance converts aggregate occupancy into congestion-aware cost-to-go fields for long-range routing. Together, these mechanisms improve throughput over MAPF-GPT (Andreychuk et al., 2025) across dense benchmarks and approach centralized RHCR-PBS (Li et al., 2021) performance, which becomes computationally prohibitive at the largest scale.

ABSTRACT

Learning-based policies offer scalable decentralized multi-agent path finding (MAPF) but struggle to maintain throughput in crowded lifelong settings. Two information gaps cause this: uncertainty about neighbors' concurrent decisions and a lack of traffic awareness beyond the local view. Relying solely on local conflict resolution or global traffic guidance fails to simultaneously resolve incompatible actions and prevent bottlenecks. We propose **PACT** (Predictive Action Coordination with Traffic Awareness) to bridge both gaps. PACT avoids exchanging action proposals by using the **Neighbor Action Predictor (NAP)** to infer neighboring agents' decisions from partial observations for local coordination. Complementarily, coarse sector-level occupancy information provides long-range awareness of congestion beyond each agent's local field of view. This enables agents to coordinate immediate interactions and bypass emerging bottlenecks without peer-to-peer communication. Across diverse MAPF benchmarks, PACT achieves over a **13×** macro-average per-map throughput ratio relative to MAPF-GPT in highly congested settings. Furthermore, it reaches over **90%** of the throughput of the centralized RHCR-PBS planner. These results demonstrate that combining inferred intentions with lightweight global traffic awareness substantially narrows the performance gap between learned policies and centralized planning.

1 INTRODUCTION

Decentralized multi-agent path finding (MAPF) allows robot teams to navigate shared environments through local decisions, with applications in warehouse automation and logistics (Stern et al., 2019; Wurman et al., 2008). In lifelong settings, agents receive new goals upon task completion, making sustained throughput the primary objective (Ma et al., 2017). Learned policies support scalable execution from fixed-size observations (Sartoretti et al., 2019; Damani et al., 2021), yet they still struggle to match centralized throughput in crowded environments (Andreychuk et al., 2025; Veera-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

paneni et al., 2025). As density increases, conflicting local movements can propagate delays through narrow passages and turn brief interactions into persistent congestion.

Two information gaps contribute to this limitation. First, *local observations do not reveal neighbors' concurrent decisions*. At intersections, individually reasonable actions can conflict because positions alone do not indicate who will move or yield. Communication-based approaches address this ambiguity by exchanging intentions, but incur coordination overhead (Ma et al., 2021a;b; Liu et al., 2026). The challenge is therefore to anticipate nearby decisions without peer-to-peer exchange.

Second, *traffic conditions beyond the field of view remain hidden*. Routes that appear attractive locally can funnel agents into distant bottlenecks (Zhang et al., 2024; Zang et al., 2025), intensifying congestion over repeated task assignments. Local conflict resolution cannot prevent this concentration, while long-range guidance alone cannot reconcile incompatible neighboring actions. Sustaining throughput therefore requires both local action coordination and long-range traffic awareness.

We propose **PACT** (**P**redictive **A**ction **C**oordination with **T**raffic Awareness) to address both gaps while retaining decentralized action decoding. Its core principle is to **predict rather than transmit action proposals**. A *Neighbor Action Predictor* (NAP) leverages the shared policy to estimate higher-priority neighbors' initial-action distributions from locally available information. These predictions guide initial decisions and support priority-based conflict refinement. This avoids peer-to-peer action exchange and runtime CS-PIBT rounds (Veerapaneni et al., 2024; Okumura et al., 2022), while residual conflicts remain subject to the environment's collision handler (Skrynnik et al., 2025).

PACT complements local coordination with **event-triggered aggregate traffic guidance**. Anonymous sector-occupancy counts from a shared source reveal distant congestion without exposing individual routes. PACT incorporates this information into the pretrained MAPF-GPT policy's existing cost-to-go input (Andreychuk et al., 2025), discouraging routes through crowded regions without increasing input length. As shown in Fig. 1, PACT improves MAPF-GPT throughput by $13.92\times$ on Warehouse with 192 agents and $2.72\times$ on Berlin with 256 agents, while approaching centralized RHCR-PBS (Li et al., 2021). It further maintains strong throughput even at the very large scale of 1024 agents. These results demonstrate the complementary roles of predictive coordination and traffic awareness in scalable, high-throughput navigation.

Our main contributions are threefold. (i) First, we develop prediction-guided coordination that combines inferred neighbor actions with confidence- and priority-based selective recovery, without peer-to-peer action exchange. (ii) Second, we introduce event-triggered sector-occupancy guidance that conveys distant congestion through existing fixed-length policy inputs. (iii) Third, we demonstrate substantial throughput gains in dense lifelong MAPF, with PACT achieving over $13\times$ the per-map throughput of MAPF-GPT on average and over 90% of centralized RHCR-PBS throughput.

2 Related Work

Learning-based decentralized MAPF. PRIMAL and PRIMAL2 (Sartoretti et al., 2019; Damani et al., 2021) train decentralized policies from partial observations through reinforcement and imitation learning. Follower (Skrynnik et al., 2024) extends this direction to lifelong MAPF, with each agent planning from information available to itself. Communication-based methods such as DCC and SCRIMP (Ma et al., 2021b; Wang et al., 2023) instead exchange learned information among neighboring agents, while graph-based policies coordinate through message passing (Li et al., 2020). MAPF-GPT (Andreychuk et al., 2025) learns local policies through large-scale expert imitation without inter-agent communication during execution. PACT retains decentralized action selection but augments local observations with inferred neighbor actions and aggregate traffic information, without peer-to-peer action exchange.

Runtime shielding and predictive coordination. PIBT (Okumura et al., 2022) coordinates agents through priority inheritance. CS-PIBT (Veerapaneni et al., 2024) adapts this mechanism into a runtime collision shield, converting jointly available policy preferences into a collision-free joint action. SSIL and SILLM (Veerapaneni et al., 2025; Jiang et al., 2025) combine learned policies with this shield at every timestep. Agent modeling offers a complementary perspective by inferring others' behavior from locally available information. SOM (Raileanu et al., 2018) applies an agent's own policy from another agent's viewpoint, while LIAM (Papoudakis et al., 2021) learns latent

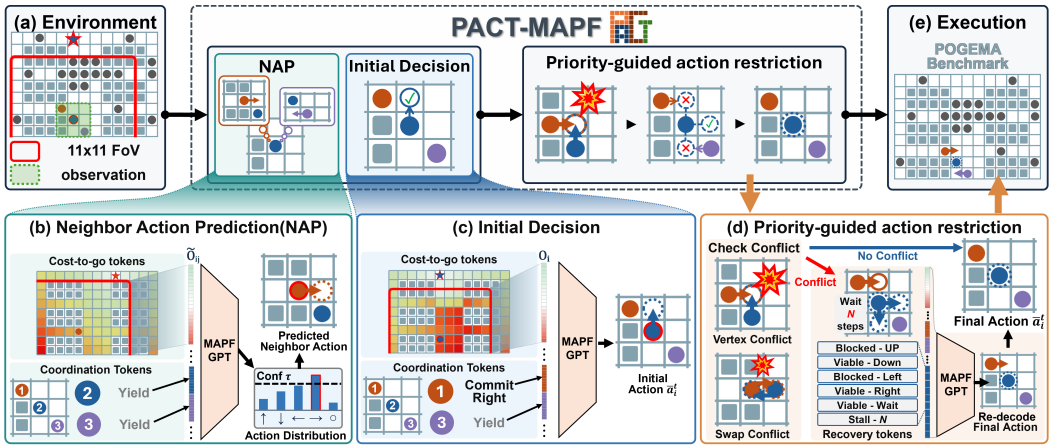


Figure 2: Overview of a single PACT decision step. (a) The ego agent observes its local 11×11 FoV. (b) NAP predicts higher-priority neighbors' actions from target-centered inputs. (c) These predictions condition the ego agent's decision through coordination tokens. (d) Confidence-qualified predictions restrict conflicting actions, with at most one masked recovery before execution.

representations of other agents under partial observability. PACT connects this perspective to local MAPF coordination by predicting neighbors' initial-action distributions. Predictions provide policy context and selectively constrain conflicting actions according to confidence and priority, avoiding a full runtime CS-PIBT round.

Congestion-aware lifelong planning. Resolving immediate conflicts alone cannot prevent repeated traffic concentration at distant bottlenecks. RHCR (Li et al., 2021) uses centralized rolling-horizon planning, commonly with Priority-Based Search (Ma et al., 2019) to coordinate agents over each planning window. LLLG (Arita & Okumura, 2026) extends configuration-based search to lifelong MAPF through local spatiotemporal guidance within receding-horizon planning. GGO and TFO (Zhang et al., 2024; Chen et al., 2024) shape motion through optimized or traffic-dependent guidance costs. Online GGO (Zang et al., 2025) adapts guidance to current traffic, while SILLM supplies learned policies with traffic-aware guide paths. Rather than constructing joint plans or guide paths, PACT converts anonymous sector-occupancy counts directly into the pretrained policy's cost-to-go input, providing long-range congestion awareness without individual route information.

3 Background

Lifelong MAPF. We consider n homogeneous agents navigating a shared four-connected graph $\mathcal{G} = (V, E)$. At each timestep, agent i chooses an action $a_i \in \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}, \text{WAIT}\}$ toward its goal g_i. Terrain-feasible actions target valid destinations $d_i(a) \in V$. Vertex and swap conflicts (Stern et al., 2019) occur when agents target the same cell or cross paths, respectively. Agents continuously receive new goals upon completion (Ma et al., 2017; Li et al., 2021) over a fixed horizon T. The objective is to maximize throughput, defined as the total number of completed goals per timestep.

Decentralized decision making. Following the MAPF-GPT interface (Andreychuk et al., 2025; Skrynnik et al., 2025) , agents share policy parameters but act independently. Agent i receives a local 11×11 observation Z_i detailing nearby agents and action histories. While the static map is public, neighbors' exact inputs and concurrent decisions remain hidden. Letting O_i denote the encoded input, the policy outputs an action distribution $\pi(\cdot \mid O_i)$. PACT augments these local observations with aggregate traffic guidance.

Pretrained action policy. PACT adapts a pretrained MAPF-GPT policy π_0 (Andreychuk et al., 2025) into π while retaining its 256-token interface: 121 cost-to-go, 10 ego, and up to twelve 10-token neighbor blocks. Each neighbor's greedy-direction field is repurposed to encode a priority-conditioned predicted action, with COMMIT/YIELD indicating the neighbor's priority role. Other

fields are preserved or modified for recovery (Appendix A.4), while the cost-to-go tokens incorporate the traffic guidance introduced in Section 4.3.

4 PACT: Predictive Action Coordination with Traffic Awareness

4.1 Prediction-Guided Action Coordination

PACT coordinates local interactions through deterministic priorities and predicted neighbor actions (Algorithm 1, Figure 2). Priority (Ma et al., 2019; Okumura et al., 2022) determines which agent adapts, while NAP anticipates higher-priority motions. Initial inputs encode sampled predictions and pairwise roles alongside the Traffic Guidance introduced in Section 4.3. Given this input O_i and retained hypotheses $\mathcal{H}_i$ (Section 4.2), agent i decodes an initial action a_i^0 and filters conflicting candidates. An admissible initial action is preserved; otherwise, PACT performs masked recovery or returns WAIT. The recovery input O_i^{rec} encodes action-wise availability and stall level ℓ_i.

Deterministic priority assignment. Agent i's coordination neighborhood is

$$\mathcal{N}_i = \{j \neq i : \|x_j - x_i\|_1 \leq 2\}. \quad (1)$$

Agents prioritize neighbors with fewer terrain-feasible moves:

$$m_i = \sum_{a \in \mathcal{A}_{\mathrm{move}}} \mathbf{1}[d_i(a) \in V], \quad (2)$$

where $\mathcal{A}_{\mathrm{move}} = \mathcal{A} \backslash \{\mathrm{WAIT}\}$. When computing m_i, we also exclude moves into cells currently occupied by other agents. Smaller m_i indicates fewer spatial alternatives and higher priority. A public hash and agent ID break ties deterministically, ensuring consistent pairwise relations. We write $i \succ j$ when i has higher priority.

The role of neighbor j in agent i's coordination input is

$$c_{ij} = \begin{cases} \mathrm{COMMIT}, & j \succ i, \\ \mathrm{YIELD}, & i \succ j. \end{cases} \quad (3)$$

Thus, COMMIT marks a higher-priority neighbor whose supported prediction can restrict i's actions. YIELD marks a lower-priority neighbor expected to adapt; it imposes no restriction on i and does not prescribe WAIT.

Algorithm 1: Prediction-Guided Action Coordination

Require: Initial input O_i, hypotheses $\mathcal{H}_i$, stall level ℓ_i
Ensure: Final action request a_i
Decode $a_i^0 \sim \pi(\cdot \mid O_i)$
$\mathcal{U}_i \leftarrow \{a \in \mathcal{A} : d_i(a) \in V\}$
for each $(j, \tilde{a}_{ij}) \in \mathcal{H}_i$ **do**
    Remove from $\mathcal{U}_i$ all actions conflicting with $\tilde{a}_{ij}$
end for
if $a_i^0 \in \mathcal{U}_i$ **then**
    return a_i^0
else if $\mathcal{U}_i = \emptyset$ **then**
    return WAIT
else
    Construct O_i^{rec} from $O_i, \mathcal{U}_i$, and ℓ_i
    Decode $a_i \sim \pi(\cdot \mid O_i^{\mathrm{rec}})$ masked to $\mathcal{U}_i$
    return a_i
end if

Priority-guided action restriction. PACT initializes $\mathcal{U}_i = \{a \in \mathcal{A} : d_i(a) \in V\}$ and screens every candidate against the retained hypotheses $\mathcal{H}_i$. Each hypothesis contains a confidence-qualified prediction $\tilde{a}_{ij}$ of a higher-priority neighbor ($c_{ij} = \mathrm{COMMIT}$; Section 4.2). Candidates producing vertex or swap conflicts with these predictions are removed. Lower-priority neighbors do not restrict $\mathcal{U}_i$, since they independently adapt their own actions.

Selective masked action selection. If $a_i^0 \in \mathcal{U}_i$, PACT retains the initial decision without further inference. Otherwise, it returns WAIT when $\mathcal{U}_i = \emptyset$, or re-decodes the policy with its distribution masked to the remaining actions:

$$a_i \sim \pi(\cdot \mid O_i^{\mathrm{rec}}), \qquad a_i \in \mathcal{U}_i. \quad (4)$$

Recovery therefore requires both a blocked initial action and a nonempty admissible set, with at most one additional policy evaluation. The stall level provides context rather than triggering recovery.

Effect of priority-guided prediction. Priority determines *whose motion takes precedence*, while NAP identifies *which actions are incompatible*. Preserving admissible decisions and selectively refining others limits re-decoding without peer-to-peer action exchange or full runtime CS-PIBT. Predictions do not guarantee collision-free joint requests; residual conflicts remain subject to the environment's collision handler.

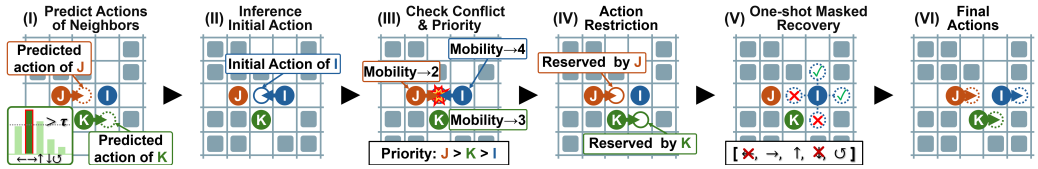


Figure 3: PACT's one-step prediction-guided coordination. NAP predicts neighbors J and K, while priority-guided screening restricts conflicts; a blocked initial action triggers one masked recovery. Panels illustrate the coordination logic rather than the exact computation order.

4.2 Neighbor Action Prediction

Target-centered observations. The neighborhood $\mathcal{N}_i$ contains at most twelve agents and covers all direct one-step conflict partners; NAP predicts those with higher priority. For target j, observer i constructs $\widetilde{O}_{ij}$ from raw, pre-truncation observations, placing j in the ego block (Raileanu et al., 2018) with target-relative descriptors. Any context agent within Manhattan distance three of j lies within distance five of i and thus inside its 11×11 FoV. Position coverage does not reveal private goals, complete histories, or the target's traffic cache. Reconstruction therefore uses only observer-visible features, replacing private traffic guidance with static shortest-path guidance from the public map and visible goal information. Static greedy-direction fields avoid recursive prediction. When i appears in the reconstructed context, it contributes only visible-agent descriptors (Appendix A.2).

Predicting initial action distributions. For target j's complete initial input O_j, the target and predicted distributions are

$$Q_j = \pi(\cdot \mid O_j), \qquad P_{ij} = f(\cdot \mid \widetilde{O}_{ij}), \tag{5}$$

where f denotes NAP. The teacher distribution Q_j supplies distillation supervision but is unavailable to observer i during deployment. NAP approximates initial, pre-refinement decisions rather than realized motions, without recovering private inputs. Decoding, distillation, and confidence gating use temperature-scaled probabilities.

Anticipating stochastic decisions. PACT samples $\tilde{a}_{ij}$ from P_{ij} using a reproducible draw keyed by the public episode seed, target ID, and timestep. Target j uses the same key for its initial request. Matching distributions, temperature, action ordering, and masks therefore yield identical predicted and actual initial actions. This couples predictions with sampled initial requests rather than only their most probable actions. Recovery uses a separate decision-stage key. Identical reconstructed inputs yield identical predicted distributions under deterministic inference.

Confidence-gated hypotheses. Predictions from reconstructed observations should not all impose hard restrictions. PACT adjusts the required predictive support using the ego agent's movement-flexibility score:

$$F_i = \sum_{a \in \mathcal{A}_{\text{move}}} w_i(a), \quad w_i(a) = \begin{cases} 0, & d_i(a) \notin V, \\ 1 - P_{ij}(\text{WAIT}), & d_i(a) = x_j,\ j \succ i, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

Unlike m_i, F_i additionally accounts for predicted neighbor motion. Using the same temperature-scaled probabilities, the confidence threshold and retained hypotheses are

$$\tau_i = \tau_{\min} + \lambda F_i, \qquad \mathcal{H}_i = \{(j, \tilde{a}_{ij}) : j \in \mathcal{N}_i,\ j \succ i,\ P_{ij}(\tilde{a}_{ij}) > \tau_i\}. \tag{7}$$

We set $\tau_{\min} = 0.2$ and $\lambda = 0.1$. Higher flexibility requires stronger predictive support for restriction. After keyed sampling, each higher-priority neighbor contributes either its sampled action or no hypothesis; no alternative is substituted. Only retained hypotheses impose hard restrictions, while all sampled predictions remain available as coordination context.

One-step example. Fig. 3 illustrates prediction, restriction, and recovery. With $J \succ K \succ I$, confidence-qualified predictions of J and K remove conflicting actions from I's admissible set. Because the initial action is blocked but alternatives remain, I performs one masked re-decoding. The final panel shows the next state, separating predicted neighbor actions from realized motions.

4.3 EVENT-TRIGGERED TRAFFIC GUIDANCE

To expose distant congestion, PACT partitions traversable cells into nonempty sectors $\mathcal{S}$ and tracks anonymous occupancy. Fig. 4 illustrates how sector occupancy is converted into congestion-aware transition costs and incorporated into the cost-to-go field. Partitions are precomputed for each map–agent-count configuration and fixed throughout an episode. Warehouse uses manually defined topological sectors; other maps use Infomap (Rosvall & Bergstrom, 2008), which partitions the traversable graph by compressing random-walk information flow. Its preferred module count equals the agent count, targeting approximately one agent per sector on average. Agent i caches

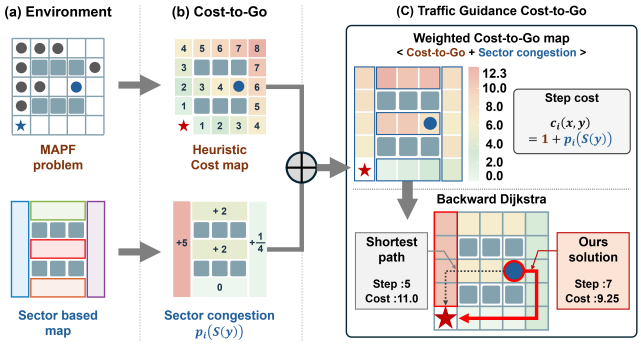


Figure 4: Sector-based Traffic Guidance. (a) At each refresh event, sector occupancies are computed from the current agent distribution and cached by the ego agent until its next refresh. (c) Backward Dijkstra constructs a congestion-aware cost-to-go field using $c_i(x, y)$, favoring less congested routes.

$$K_i = (n_i(S))_{S \in \mathcal{S}}, \tag{8}$$

where $n_i(S)$ counts agents in sector S, including i in its occupied sector. Snapshots refresh only at initialization or when the ego agent enters a new sector. Cached occupancy determines the congestion penalty

$$p_i(S) = C \cdot \frac{n_i(S)}{\max\big(1,\ |S| - n_i(S)\big)}, \tag{9}$$

where $|S|$ is sector capacity and $C = 1$ sets the default guidance strength. The penalty increases as remaining capacity decreases but stays finite at saturation. Entering cell y incurs

$$c_i(x, y) = 1 + p_i(S(y)). \tag{10}$$

Here, $S(y)$ denotes the sector containing y. Backward Dijkstra computes the congestion-aware cost-to-go field for non-goal cells as

$$D_i(x) = \min_{y:(x,y)\in E} \left[c_i(x, y) + D_i(y)\right], \tag{11}$$

with $D_i(g_i) = 0$. This favors less congested routes without prohibiting entry into occupied sectors.

An encoded 11×11 crop of D_i replaces the existing 121 cost-to-go tokens for both initial and recovery decoding. The field is recomputed after snapshot refreshes or goal changes, with goal changes reusing cached occupancy. Agents can hold asynchronous, stale traffic estimates, but these affect only soft route guidance, not hard action restrictions.

4.4 POLICY ADAPTATION AND PREDICTOR DISTILLATION

Training comprises coordination adaptation, recovery adaptation, and NAP distillation (Hinton et al., 2015). Starting from MAPF-GPT (Andreychuk et al., 2025), the adaptation stages sequentially learn coordination-aware initial decisions and conflict-triggered recovery under CS-PIBT (Veerapaneni et al., 2024; 2025) supervision. The pretrained backbone remains frozen; only LoRA adapters and new token embeddings are optimized. Stage-specific ranks appear in Tab. 4.

Coordination and recovery adaptation. Coordination adaptation encodes priority-dependent, CS-PIBT-assigned neighbor actions into the coordination fields and supervises the policy with the teacher's final action. Recovery adaptation starts from this policy, collecting rollout states from the latest adapted policy and relabeling them once with CS-PIBT, without iterative dataset aggregation (Ross et al., 2011). Recovery inputs encode stall level and action-wise availability. Policy inputs and teacher preferences share the same congestion-aware cost-to-go field.

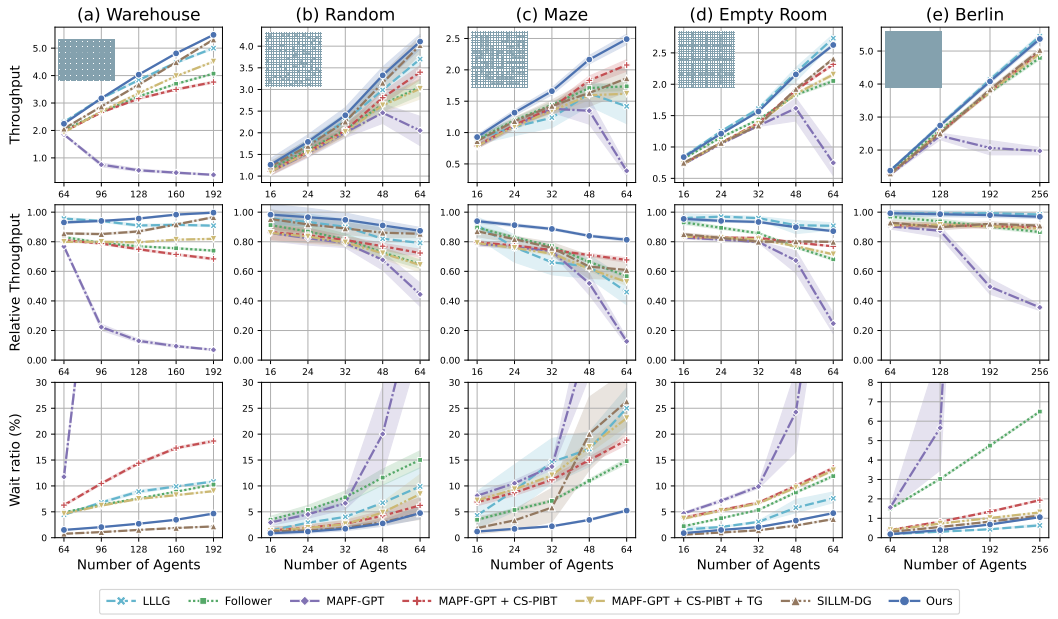


Figure 5: Performance across five standard-scale environments. Columns correspond to Warehouse, Random, Maze, Empty Room, and Berlin; rows report throughput, relative throughput normalized by the representative centralized planner RHCR-PBS (Li et al., 2021), and wait ratio. Curves average 20 runs per setting, with shaded regions indicating 95% confidence intervals.

Predictor distillation. NAP is initialized from the final adapted policy π, which is frozen as its distribution teacher. The dataset $\mathcal{D}_{\mathrm{NAP}}$ pairs complete target inputs O_j with reconstructed inputs $\widetilde{O}_{ij}$, yielding teacher and predictor distributions Q_j and P_{ij}. Both use temperature 0.2, and a separate rank-16 NAP adapter minimizes

$$\mathcal{L}_{\mathrm{NAP}} = \mathbb{E}_{(O_j,\widetilde{O}_{ij})\sim\mathcal{D}_{\mathrm{NAP}}}\left[D_{\mathrm{KL}}(P_{ij}\|Q_j)\right]. \tag{12}$$

We distill distributions because confidence-based screening depends on action probabilities rather than only modal actions.

Training and deployment interfaces. Coordination and recovery training use CS-PIBT-assigned actions and availability; deployment instead uses NAP predictions and constructs $\mathcal{U}_i$ through terrain feasibility, priority, and confidence rules. Recovery therefore uses locally inferred masks rather than the teacher-derived availability seen during adaptation. NAP training accesses complete target inputs, whereas deployment uses observer-visible reconstructions. Only π and f are learned deployment components; priorities, confidence gating, conflict restrictions, and traffic-field construction remain rule-based, with no runtime CS-PIBT teacher (Appendix A.7).

5 Experiments

5.1 Experimental Setup

Baselines and maps. We compare PACT with its pretrained backbone, **MAPF-GPT** (Andreychuk et al., 2025); **MAPF-GPT + CS-PIBT** (Veerapaneni et al., 2024), with and without **Traffic Guidance (TG)**; decentralized **Follower** (Skrynnik et al., 2024); communication-based **SILLM-DG** (Jiang et al., 2025); and centralized **LLLG** (Arita & Okumura, 2026). **MAPF-GPT + CS-PIBT + TG** serves as the adaptation teacher, while **RHCR-PBS** (Li et al., 2021; Ma et al., 2019) provides the centralized reference. We evaluate five standard-scale environments, Warehouse, Random, Maze, Empty Room, and Berlin (Andreychuk et al., 2025; Skrynnik et al., 2025), and two large-scale maps, Warehouse 20-40-10-2-1 and den520d (Stern et al., 2019). These settings evaluate generalization to held-out layouts both within and beyond the training map families.

Evaluation metrics and protocol. The main benchmark uses $T = 512$ timesteps with POGEMA's `soft` collision handling (Skrynnik et al., 2025), under which rejected moves leave agents stationary. Throughput measures goals completed by all agents per environment timestep, while relative throughput normalizes this value by RHCR-PBS under the same setting; 1.0 denotes reference parity rather than optimality. Wait ratio is the fraction of stationary agents averaged over timesteps, including explicit `WAIT` actions and collision-rejected movements. Standard- and large-scale settings use 20 and 10 runs, respectively. Goals are reassigned immediately upon completion, and density is computed over reachable traversable cells. Appendix C provides map statistics, task generation, training–evaluation separation, baseline configurations, and hardware details.

5.2 Benchmark Performance

Fig. 5 compares lifelong MAPF performance across five standard-scale environments. As density increases, MAPF-GPT suffers substantial throughput degradation and rising waiting, particularly on Warehouse, Maze, Empty Room, and Berlin. Runtime CS-PIBT alleviates these failures, while Traffic Guidance further improves several dense settings. PACT maintains high throughput as density grows, showing that prediction-guided coordination and congestion-aware routing remain effective under frequent interactions.

At the highest densities, PACT achieves the highest throughput among decentralized and communication-based baselines across all five environments. In highly congested settings, PACT achieves **over 13× MAPF-GPT throughput** and **over 90% of centralized RHCR-PBS throughput**, on average across maps. The gain is especially pronounced on Warehouse, where MAPF-GPT collapses under dense traffic while PACT continues to scale. Similar trends appear on the other maps, and PACT also outperforms the MAPF-GPT + CS-PIBT + TG adaptation teacher, indicating gains beyond Traffic Guidance and runtime shielding alone.

Wait-ratio results show a consistent pattern. MAPF-GPT accumulates substantial stationary behavior at higher densities, whereas PACT maintains low waiting across environments. Although CS-PIBT and TG also reduce waiting, PACT achieves a stronger throughput–waiting trade-off by combining traffic-aware decisions with selective recovery. While the gap to RHCR-PBS varies by map and density, PACT substantially narrows it without peer-to-peer action exchange or a full runtime CS-PIBT round. Appendix B provides further analyses of scalability, long-horizon behavior, conflict reduction, and interaction with CS-PIBT.

5.3 Ablation and Analysis

Fig. 6 evaluates PACT's components at the highest-density settings, macro-averaged across the five standard-scale maps. Starting from MAPF-GPT, **+CT (Coordination Tokens) only** adds NAP-based coordination tokens, **+TG only** introduces congestion-aware Traffic Guidance, and **+CT & TG** combines both. **+Selective Recovery** further enables confidence- and priority-guided masked re-decoding, while **+Recovery Tokens** adds action-viability and stall-level context to form full PACT.

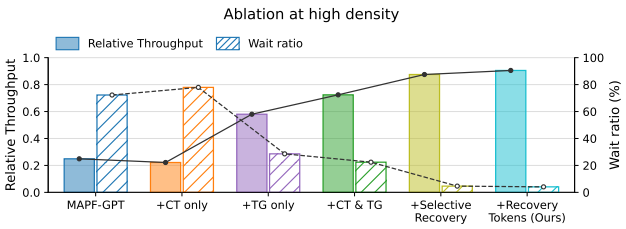


Figure 6: Ablation at high density, showing macro-averaged relative throughput and wait ratio as PACT components are progressively added.

CT alone provides little benefit, indicating that predicted neighbor actions are insufficient without broader traffic awareness. TG substantially improves throughput and reduces waiting by steering agents away from congested regions, while combining CT with TG yields a further gain. **Selective recovery provides the largest additional improvement**, sharply reducing waiting by revising blocked initial actions. Recovery tokens give a smaller final gain through explicit viability and stall context. Overall, the results show that PACT benefits from the complementary combination of traffic guidance, prediction-aware coordination, and selective recovery.

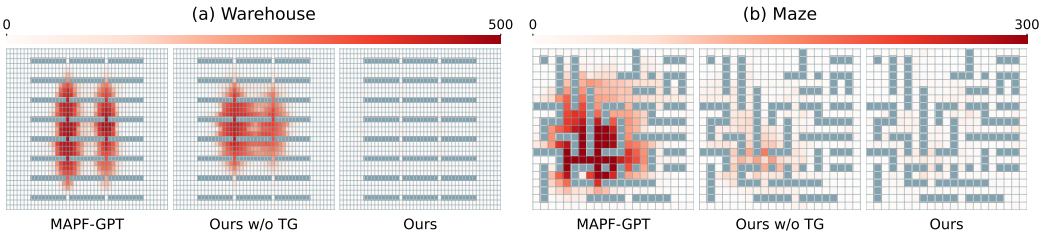


Figure 7: Spatial congestion analysis on representative high-density Warehouse ($N = 192$) and Maze ($N = 64$) settings over 20 runs. We compare MAPF-GPT, PACT without Traffic Guidance (Ours w/o TG), and full PACT. Heatmaps show per-cell accumulated stops.

Table 1: Effect of coordination information at the highest-density settings. GT uses actual initial actions as an oracle; results are averaged over 20 runs.

Method	Warehouse	Random	Maze	Empty Room	Berlin
Throughput ↑					
Greedy	4.912	3.794	2.042	2.405	5.154
NAP	5.487	4.164	2.506	2.633	5.359
GT (oracle)	5.622	4.174	2.513	2.674	5.402
Wait ratio (%) ↓					
Greedy	9.15	8.06	11.44	6.60	2.40
NAP	4.80	4.90	5.43	4.84	1.13
GT (oracle)	2.76	2.74	3.14	3.17	0.44

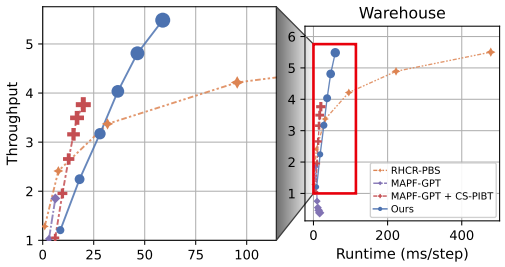


Figure 8: Runtime–throughput trade-off on Warehouse with 64–192 agents. Runtime denotes the total computation time per timestep.

Effect of coordination information. Table 1 compares Greedy, NAP, and ground-truth (GT) initial actions within the same PACT framework. Greedy uses each neighbor's shortest-path action as the coordination estimate. Replacing it with NAP predictions improves throughput by 4.0–22.7% and reduces waiting by 1.27–6.01 percentage points across all five maps. NAP also reaches **97.6–99.8% of GT-oracle throughput**, showing that predicted initial actions provide substantially more useful coordination information than greedy-action assumptions.

Spatial congestion analysis Fig. 7 visualizes accumulated stops in high-density Warehouse and Maze settings. MAPF-GPT shows concentrated stopping around aisles and bottlenecks, while PACT without Traffic Guidance (Ours w/o TG) substantially disperses these hotspots. Full PACT further suppresses congestion, particularly along central Warehouse aisles and major Maze bottlenecks. These results show that local prediction and recovery reduce immediate conflicts, while Traffic Guidance additionally steers agents away from persistently congested regions.

Runtime analysis Fig. 8 compares the runtime–throughput trade-off on Warehouse with 64–192 agents. MAPF-GPT is fastest but degrades at high density, while CS-PIBT improves coordination with added overhead. PACT reaches about 5.5 throughput at **roughly 60 ms per step with 192 agents**, whereas RHCR-PBS requires several hundred milliseconds for comparable performance. Thus, PACT approaches centralized throughput at substantially lower computational cost.

6 Conclusion

We presented PACT, a scalable, decentralized lifelong MAPF framework that bridges predictive local coordination with aggregate traffic awareness. PACT operates through two key mechanisms: NAP, which anticipates higher-priority neighbor actions to resolve local conflicts, and sector-level Traffic Guidance, which steers agents away from distant congestion without relying on peer-to-peer communication. Across diverse benchmarks, these complementary approaches substantially improve overall throughput and minimize wait times. Our results demonstrate that combining local action prediction with coarse global traffic information not only narrows the performance gap with centralized planners, but also provides a highly practical direction for scaling lifelong MAPF.

AI USE STATEMENT

We used large language models (LLMs) solely to correct grammar and improve the wording, clarity, and readability of text during the preparation of this manuscript. These tools were not used to generate research ideas, develop methods, design or conduct experiments, write code, or analyze or interpret results. No ideas, experiments, analyses, or research contributions were generated by LLMs in the preparation of this manuscript. All scientific contributions and substantive writing were carried out directly by the authors. All results produced with the assistance of AI were reviewed and revised by the authors, and the authors take full responsibility for the final content of this paper.

REFERENCES

Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, and Alexey Skrynnik. Mapf-gpt: Imitation learning for multi-agent pathfinding at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 23126–23134, 2025.

Tomoki Arita and Keisuke Okumura. Lifelong lacam with local guidance for lifelong mapf. *arXiv preprint arXiv:2605.16855*, 2026.

Zhe Chen, Daniel Harabor, Jiaoyang Li, and Peter J Stuckey. Traffic flow optimisation for lifelong multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 20674–20682, 2024.

Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

He Jiang, Yutong Wang, Rishi Veerapaneni, Tanishq Duhan, Guillaume Sartoretti, and Jiaoyang Li. Deploying ten thousand robots: Scalable imitation learning for lifelong multi-agent path finding. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, 2025.

Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11272–11281, 2021.

Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. Graph neural networks for decentralized multi-robot path planning. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 11785–11792, 2020.

Zhongbo Liu, Guillermo Palacios-Navarro, and Raquel Lacuesta. Pico-mapf: Predictive influence-based coordinated multi-agent path finding for large-scale high-density environments. *IEEE Software*, 2026.

Hang Ma, Jiaoyang Li, TK Kumar, and Sven Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. *arXiv preprint arXiv:1705.10868*, 2017.

Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7643–7650, 2019.

Ziyuan Ma, Yudong Luo, and Hang Ma. Distributed heuristic multi-agent path finding with communication. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8699–8705, 2021a.

Ziyuan Ma, Yudong Luo, and Jia Pan. Learning selective communication for multi-agent path finding. *IEEE Robotics and Automation Letters*, 7(2):1455–1462, 2021b.

Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310:103752, 2022.

Georgios Papoudakis, Filippos Christianos, and Stefano Albrecht. Agent modelling under partial observability for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:19210–19222, 2021.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pp. 4257–4266, 2018.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008.

Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

Alexey Skrynnik, Anton Andreychuk, Maria Nesterova, Konstantin Yakovlev, and Aleksandr Panov. Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 17541–17549, 2024.

Alexey Skrynnik, Anton Andreychuk, Anatolii Borzilov, Alexander Chernyavskiy, Konstantin Yakovlev, and Aleksandr Panov. Pogema: A benchmark platform for cooperative multi-agent pathfinding. In *International Conference on Learning Representations*, pp. 5576–5606, 2025.

Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, Thirunarayanapuram Krishnamachari Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the international symposium on combinatorial search*, pp. 151–158, 2019.

Rishi Veerapaneni, Qian Wang, Kevin Ren, Arthur Jakobsson, Jiaoyang Li, and Maxim Likhachev. Improving learnt local mapf policies with heuristic search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 597–606, 2024.

Rishi Veerapaneni, Arthur Jakobsson, Kevin Ren, Samuel Kim, Jiaoyang Li, and Maxim Likhachev. Work smarter not harder: Simple imitation learning with cs-pibt outperforms large-scale imitation learning for mapf. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10229–10236, 2025.

Yutong Wang, Bairan Xiang, Shinan Huang, and Guillaume Sartoretti. Scrimp: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9301–9308, 2023.

Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.

Hongzhi Zang, Yulun Zhang, He Jiang, Zhe Chen, Daniel Harabor, Peter J Stuckey, and Jiaoyang Li. Online guidance graph optimization for lifelong multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 14726–14735, 2025.

Yulun Zhang, He Jiang, Varun Bhatt, Stefanos Nikolaidis, and Jiaoyang Li. Guidance graph optimization for lifelong multi-agent path finding. *arXiv preprint arXiv:2402.01446*, 2024.

Table 2: Runtime information-access model of PACT. Public denotes system-wide information, Private denotes agent-local or locally observed information that is not globally shared, and Not shared denotes information that PACT does not receive from other agents. "Enters NAP input for j" refers specifically to the reconstructed neural input $\widetilde{O}_{ij}$, excluding auxiliary variables used for priority or keyed decoding. A visible agent block contains relative position, clipped relative goal, the five most recent actions, and one static greedy-direction field.

Information	**Class**	**Ego i access**	**Enters NAP input for target j**	**Role**
Static map G	Public	Yes	Indirectly	Used to construct terrain feasibility, static shortest-path guidance, and static greedy directions.
Public coordination keys (ID, seed, timestep)	Public	Yes	No	Used for deterministic priority tie-breaking and target-keyed decoding.
Per-sector occupancy counts	Public	Via snapshot	No	Used for the ego agent's Traffic Guidance, but not for target reconstruction.
Ego-local policy state	Private	Yes	Only if visible	Used for ego action selection; when ego appears in the target-centered context, only its visible-agent descriptors are exposed.
Visible neighbor block	Private	Within FoV	Yes	Provides relative position, clipped relative goal, five recent actions, and static greedy direction for target reconstruction.
Target exact goal g_j	Private	No	No	Unavailable to NAP; reconstruction uses only the observable goal descriptor.
Target history beyond five actions	Private	No	No	Not represented in the reconstructed target input.
Target traffic cache K_j	Private	No	No	Omitted from reconstruction and replaced by static shortest-path guidance.
Target initial/final actions	Not shared	No	No	The initial action is predicted by NAP; recovery and final actions are not directly observed.
Per-agent routes or action proposals	Not shared	No	No	PACT neither requires nor exchanges explicit routes or peer-to-peer action proposals.

A Additional Method Details

A.1 Information Access Model

Tab. 2 summarizes the information available to an agent during PACT execution and distinguishes it from the information entering NAP's reconstructed target input. NAP does not recover a target's complete policy input. Instead, $\widetilde{O}_{ij}$ is constructed from observer-visible descriptors together with static guidance derived from the public map. The target's exact goal, history beyond the visible five-action window, and traffic cache remain unavailable to the predictor. Public identifiers, the episode seed, and the timestep are also excluded from the neural input; they are used only for deterministic priority tie-breaking and target-keyed action sampling.

Traffic Guidance introduces the only shared non-local environment-state information: anonymous per-sector occupancy counts obtained through a one-way shared source. Each agent uses its cached snapshot to construct its own congestion-aware cost-to-go field, but these counts are not used to reconstruct a target-specific traffic field for NAP. PACT therefore does not require peer-to-peer action proposals, explicit per-agent routes, or neighbors' concurrent decisions. Neighbor initial actions are instead inferred locally, while recovery and final actions are not directly predicted or available to NAP at decision time.

A.2 Observation Reconstruction and Conditional Consistency

Coverage of one-step interactions. A vertex conflict requires two agents to reach a common cell in at most one cardinal move each, placing their current positions within Manhattan distance two. A swap requires adjacent positions. Thus, the neighborhood in Eq. 1 contains every direct one-step conflict partner. With at most one agent per cell, the four cells at distance one and eight at distance two give at most twelve neighbors. NAP predicts only the higher-priority subset. This coverage does not extend to longer movement-dependency chains affected by collision handling.

Spatial coverage of reconstructed context. For a prediction target $j \in \mathcal{N}_i$, a context agent r within Manhattan distance three of j satisfies

$$\|x_r - x_i\|_1 \leq \|x_r - x_j\|_1 + \|x_j - x_i\|_1 \leq 3 + 2 = 5. \tag{13}$$

Its position therefore lies inside the 11×11 FoV centered at observer i. Reconstruction uses the raw observation Z_i before ego-centered neighbor truncation, which could discard agents needed for the target's context. The target occupies the ego block, while context descriptors are expressed relative to it. Ordering, tie-breaking, truncation to twelve context agents, and padding follow shared rules. If observer i appears in this context, it contributes only visible-agent descriptors, not its additional private information. The bound establishes position coverage, not equivalence to the target's complete input O_j.

Information restrictions. Spatial coverage does not reveal exact private goals, complete histories, or the target's traffic cache. Accordingly, $\widetilde{O}_{ij}$ replaces private traffic guidance with static shortest-path guidance derived from the public map $\mathcal{G}$ and goal information available in Z_i. An observer-relative projected goal is not treated as the target's unobserved exact goal. Static greedy-direction fields avoid recursive prediction and use the same information restrictions. Reconstruction therefore provides an observer-visible approximation rather than recovering omitted target-private inputs.

Conditional observer consistency. For observers i and k that both predict target j, identical reconstructed inputs yield identical distributions under deterministic NAP inference:

$$\widetilde{O}_{ij} = \widetilde{O}_{kj} \quad \Longrightarrow \quad P_{ij} = P_{kj}, \qquad j \in \mathcal{N}_i \cap \mathcal{N}_k. \tag{14}$$

Under the matching decoding conditions in Appendix A.3, the shared target-keyed draw also gives $\tilde{a}_{ij} = \tilde{a}_{kj}$. This conditional agreement does not guarantee accuracy: observers can omit the same private information and share a prediction error. Their admissible sets may still differ because their positions, other retained hypotheses, and confidence thresholds differ. Matching predictions therefore imply neither identical restrictions nor identical final actions.

A.3 Reproducible Decoding

PACT uses temperature-scaled probabilities, a fixed action ordering, and pseudo-random draws keyed by the public episode seed, agent identifier, timestep, and decision stage. To sample $\tilde{a}_{ij}$, observer i uses target j's initial-stage key rather than its own identifier. Target j uses the same key for its initial request a^0_j. Recovery uses a separate stage key and does not reuse the initial-stage draw.

For matching temperature, action ordering, action mask, sampling rule, and draw,

$$P_{ij} = Q_j \quad \Longrightarrow \quad \tilde{a}_{ij} = a^0_j. \tag{15}$$

This implication concerns the initial request only: recovery can change the target's final request, and collision handling can further change its realized motion. Distribution equality is not assured because P_{ij} uses $\widetilde{O}_{ij}$ rather than the complete input O_j determining Q_j. Throughout inference, policy

and predictor distributions are evaluated at the common decoding temperature. The resulting P_{ij} is used unchanged for keyed sampling and the computation of F_i and $\mathcal{H}_i$. Keyed sampling applies no further temperature scaling; recovery restricts and renormalizes the action-policy distribution over $\mathcal{U}_i$.

A.4 Token Encoding and Recovery Context

Input layout and pairwise roles. The 256-token interface contains 121 guidance tokens, one 10-token ego block, up to twelve 10-token neighbor blocks, and padding. Each agent block contains position and goal coordinates, five history entries, and a greedy-action field (Andreychuk et al., 2025). Initial decoding retains the ego block and repurposes neighbor greedy-action fields for coordination. Following Eq. 3, c_{ij} denotes neighbor j's role in O_i: COMMIT when $j \succ i$ and YIELD when $i \succ j$. Higher-priority fields encode sampled predictions $\tilde{a}_{ij}$ with their COMMIT role; lower-priority fields indicate YIELD without requiring an action prediction. These roles are not movement actions, and YIELD does not prescribe WAIT. Unmodified fields retain their original meanings. Reconstructed NAP inputs instead use the static guidance and greedy-direction fields in Appendix A.2.

Traffic guidance tokens. For a traversable cell x with finite $D_i(x)$ and $D_i(x_i)$, the local guidance value is

$$q_i(x) = \mathrm{clip}_{[-20,20]}(\mathrm{round}\,[D_i(x) - D_i(x_i)])\,. \tag{16}$$

The 11×11 crop centered at x_i supplies the 121 guidance tokens, while obstacles retain their blocked encoding. Reachable-goal sampling ensures a finite reference $D_i(x_i)$ in the evaluated episodes. The numerical rule applies only to finite distances; disconnected free cells remain outside its domain. Initial and recovery decoding share the cached field and guidance crop within each timestep.

Recovery fields. The recovery input O_i^{rec} preserves the neighbor coordination fields and traffic-guidance crop from O_i. Only the ego history and greedy-action fields change. The five history slots encode membership in $\mathcal{U}_i$ using VIABLE/BLOCKED tokens in the fixed action ordering; viability denotes local mask membership, not guaranteed joint feasibility. The ego greedy-action field encodes $\ell_i = b(s_i) \in \{0,1,2,3,4\}$, where s_i counts consecutive stationary transitions and b is the five-level discretization. These transitions include collision-rejected movements, not only requested WAIT actions. Stall level supplies context; recovery requires $a_i^0 \notin \mathcal{U}_i$ and $\mathcal{U}_i \neq \emptyset$, rather than merely a nonempty restriction set.

A.5 Conflict Restrictions and Full Inference

Public priority order. Let m_i be the terrain-feasible movement count in Eq. 2. For episode seed e and timestep t, define

$$\kappa_i = \big(m_i, h(e,t,i), i\big), \qquad j \succ i \iff \kappa_j <_{\mathrm{lex}} \kappa_i, \tag{17}$$

where h is the public tie-breaking hash and the identifier resolves any remaining ties. Smaller m_i gives higher priority. Unlike the movement-flexibility score F_i, m_i treats cells currently occupied by other agents as unavailable but does not depend on predicted neighbor motion. Pairwise comparisons can therefore be reproduced from visible agent positions, the public map, and public tie-breaking variables. Under Eq. 3, $j \succ i$ means $c_{ij} = \mathrm{COMMIT}$ and $c_{ji} = \mathrm{YIELD}$.

Conflict cells. For a hypothesized action a of neighbor j, write $y_j(a) = d_j(a)$ for its destination. The ego destinations inducing a vertex or swap conflict with $x_j \to y_j(a)$ are

$$\mathcal{C}_{ij}(a) = \big\{v \in V : v = y_j(a) \ \vee\ \big(v = x_j \wedge y_j(a) = x_i\big)\big\}\,. \tag{18}$$

Equivalently,

$$\mathcal{C}_{ij}(a) = \big(\{y_j(a)\} \cap V\big) \cup \begin{cases} \{x_j\}, & y_j(a) = x_i, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{19}$$

The first term excludes the hypothesized neighbor destination, preventing a vertex conflict with that hypothesis. Intersecting with V preserves the valid-destination domain. If $y_j(a) = x_i$, the additional exclusion of x_j prevents the swap $x_i \to x_j$, $x_j \to x_i$; the first term also excludes waiting at x_i.

Restriction construction. The set $\mathcal{H}_i$ retains only confidence-qualified predictions of higher-priority neighbors. We define

$$\mathcal{R}_i = \bigcup_{(j,a)\in\mathcal{H}_i} \mathcal{C}_{ij}(a), \tag{20}$$

and construct the terrain-feasible admissible set as

$$\mathcal{U}_i = \{a \in \mathcal{A} : d_i(a) \in V \setminus \mathcal{R}_i\}. \tag{21}$$

Every candidate is screened against all retained hypotheses, and set union makes $\mathcal{R}_i$ independent of their processing order. Lower-priority neighbors impose no restriction on $\mathcal{U}_i$; they independently apply the same priority rule when choosing their actions.

All sampled higher-priority actions remain available as coordination context, regardless of confidence gating. Only the retained subset imposes hard restrictions. Since $F_i \in [0, 4]$, the default parameters give

$$\tau_i \in [\tau_{\min}, \tau_{\min} + 4\lambda] = [0.2, 0.6].$$

Each higher-priority neighbor contributes at most one hypothesis. Its sampled action is retained only if $P_{ij}(\tilde{a}_{ij}) > \tau_i$; otherwise, it contributes no hard hypothesis. Neither a modal action nor another sample is substituted after a failed gate.

Selective one-shot recovery and execution. PACT first tests the initial request against $\mathcal{U}_i$. If $a_i^0 \in \mathcal{U}_i$, it returns $a_i = a_i^0$ without another policy evaluation. Otherwise, a nonempty $\mathcal{U}_i$ triggers one evaluation on O_i^{rec}, with decoding masked to $\mathcal{U}_i$ and a separate recovery-stage draw. The recovered action therefore respects every retained hypothesis, not only the one invalidating the initial request. A nonempty $\mathcal{R}_i$ alone does not trigger recovery when the initial action remains admissible.

If $\mathcal{U}_i = \emptyset$, PACT returns WAIT without recovery inference. This fallback may itself violate a predicted restriction. More generally, PACT does not guarantee a collision-free joint request: prediction errors, discarded hypotheses, lower-priority interactions, or a neighbor's own recovery can leave residual conflicts. Final requests remain subject to the environment's collision handler, which can reject movements and leave agents stationary.

Algorithm 2 expands Algorithm 1 with traffic updates and target-centered prediction. Required traffic updates precede input construction. NAP predicts higher-priority neighbors before the initial action is decoded, and sampled predictions populate coordination fields independently of confidence qualification. The procedure uses one initial action-policy evaluation and at most one recovery evaluation, in addition to NAP inference. Recovery reuses the existing neighbor predictions and traffic field rather than repeating either computation.

A.6 Traffic-Field Updates and Computational Cost

Directed costs and refresh events. Agent i caches $K_i = (n_i(S))_{S\in\mathcal{S}}$ and constructs D_i using Eqs. Eq. 9–Eq. 11. Although grid adjacency is bidirectional, destination-dependent transition costs can be asymmetric. Backward Dijkstra must therefore relax a predecessor x through y using $c_i(x, y) + D_i(y)$, not $c_i(y, x) + D_i(y)$. Occupancy snapshots refresh at initialization or when agent i enters a different sector. The field is rebuilt after a refresh or goal change; a goal change alone reuses K_i. Updates finish before initial input construction, and the field remains fixed between initial and recovery decoding. Changes in other agents' sectors do not independently trigger an update of i's cache.

PACT does not exchange action proposals or learned messages between agents. Traffic Guidance nevertheless requires anonymous sector-occupancy counts from a shared one-way information source. Thus, action decoding is decentralized, while traffic awareness depends on shared aggregate information. Table 3 summarizes this interface.

Computation and information trade-offs. A map-wide field update has binary-heap complexity $O((|V| + |E|) \log |V|)$. Caching avoids repeated searches between refresh or goal-change events without reducing the per-update cost. NAP requires reconstruction and prediction for up to twelve higher-priority targets, in addition to one or two action-policy evaluations. Fixed-length inputs and one-shot recovery therefore do not imply map-size-independent total computation or negligible prediction overhead. Agents can hold snapshots of different ages and consequently different route

Algorithm 2 PACT inference for agent i at one timestep

Require: Raw observation Z_i, goal g_i, stall level ℓ_i
Require: Public map $\mathcal{G}$, sectors $\mathcal{S}$, seed e, timestep t
Require: Cached sector counts K_i and cost-to-go field D_i
Ensure: Final action request a_i

if K_i is uninitialized or i has entered a different sector **then**
    $K_i \leftarrow \textsc{RefreshOccupancy}(\mathcal{S})$
end if
if D_i is uninitialized, K_i was refreshed, or g_i changed **then**
    $D_i \leftarrow \textsc{BuildTrafficField}(\mathcal{G}, g_i, \mathcal{S}, K_i)$
end if
$\mathcal{N}_i \leftarrow \{j \neq i : \|x_j - x_i\|_1 \leq 2\}$
Compute c_{ij} for $j \in \mathcal{N}_i$ using Eqs. Eq. 17 and Eq. 3
for $j \in \mathcal{N}_i$ with $c_{ij} = \text{COMMIT}$ **do**
    $\widetilde{O}_{ij} \leftarrow \textsc{ReconstructTarget}(Z_i, \mathcal{G}, j)$
    $P_{ij} \leftarrow f(\cdot \mid \widetilde{O}_{ij})$ at the decoding temperature
    $\tilde{a}_{ij} \leftarrow \textsc{DecodeInitial}(P_{ij}, j, e, t)$
end for
Compute F_i, τ_i, and $\mathcal{H}_i$ using Eqs. Eq. 6–Eq. 7
Build O_i from Z_i, g_i, D_i, roles c_{ij}, and higher-priority samples $\tilde{a}_{ij}$
$a_i^0 \leftarrow \textsc{DecodeInitial}(\pi(\cdot \mid O_i), i, e, t)$
Construct $\mathcal{R}_i$ from $\mathcal{H}_i$ using Eq. 20
$\mathcal{U}_i \leftarrow \{a \in \mathcal{A} : d_i(a) \in V \setminus \mathcal{R}_i\}$
if $a_i^0 \in \mathcal{U}_i$ **then**
    return a_i^0
end if
if $\mathcal{U}_i = \emptyset$ **then**
    return WAIT
end if
$O_i^{\text{rec}} \leftarrow \textsc{BuildRecoveryInput}(O_i, \mathcal{U}_i, \ell_i)$
$a_i \leftarrow \textsc{DecodeRecovery}(\pi(\cdot \mid O_i^{\text{rec}}), \mathcal{U}_i, i, e, t)$
return a_i

Table 3: Traffic-Guidance information and update interface. The mean update interval is measured over the highest-density standard-scale evaluations.

Property	**Definition**
Information	Anonymous per-sector occupancy counts
Source	Shared occupancy service via one-way broadcast
Peer-to-peer action exchange	None
Snapshot refresh	Initialization or ego sector transition
Mean update interval	3.4 steps
Infomap sector target	n sectors; Warehouse uses manual partitions

preferences. These differences affect soft guidance, not the hard restriction rule. Because $\widetilde{O}_{ij}$ omits the target's private cache, a cache update can change Q_j without being directly represented in the predictor input.

A.7 Training Data and Deployment Inputs

Distinct supervision sources. CS-PIBT supplies action labels for coordination and recovery adaptation using preferences from the frozen pretrained policy π_0 with Traffic Guidance. NAP instead learns the initial-action distribution $Q_j = \pi(\cdot \mid O_j)$ of the final adapted policy π, which remains

Table 4: Training configurations for PACT. Effective batch size is the microbatch size multiplied by the gradient-accumulation count. Learning rates use cosine decay to 10^{-5} after warmup. Both NAP distributions use temperature 0.2.

Setting	**Coordination**	**Recovery**	**NAP**
Initialization	MAPF-GPT	Coordination-adapted policy	Recovery-adapted policy
LoRA (r, α)	$(16, 32)$	$(32, 64)$	$(16, 32)$
Peak learning rate	3×10^{-4}	1.4×10^{-3}	1×10^{-4}
Warmup length	200	200	100
Effective batch size	$128 \times 32 = 4096$	$256 \times 32 = 8192$	$256 \times 8 = 2048$
Training examples	2,211,840	256,000	1,277,952 pairs
Training budget	5,000 microbatch iterations	500 microbatch iterations	6,000 training iterations
Objective	Cross-entropy	Cross-entropy	Reverse KL
Checkpoint selection	Final checkpoint	Validation throughput	Validation throughput

frozen during predictor training. Each pair $(O_j, \widetilde{O}_{ij})$ provides the complete input for computing Q_j and the reconstructed input for predicting $P_{ij} = f(\cdot \mid \widetilde{O}_{ij})$. Distillation targets the distribution before selective recovery or collision handling, not a CS-PIBT-assigned or executed action. Distribution supervision preserves the action probabilities needed for confidence gating rather than only a modal label.

Adaptation sequence. Coordination and recovery adaptation are performed sequentially. The pretrained backbone remains frozen while LoRA adapters and newly introduced token embeddings are optimized. Coordination adaptation encodes priority roles and CS-PIBT-assigned neighbor actions, with the teacher's final action supervising the ego decision. Recovery adaptation starts from the coordination-adapted policy, collects rollout states from the latest adapted policy, and relabels them once with CS-PIBT, without iterative dataset aggregation. Recovery inputs encode action-wise availability and stall level. Policy inputs and teacher preferences use the same congestion-aware cost-to-go field. NAP is then initialized from the final adapted policy and trained with a separate LoRA adapter. Stage-specific ranks and optimization settings appear in Table 4.

Training–execution differences. Deployed coordination replaces teacher-assigned neighbor actions with sampled higher-priority predictions $\tilde{a}_{ij}$; lower-priority neighbors retain their YIELD role. Deployed recovery replaces teacher-derived availability with $\mathcal{U}_i$, constructed from terrain feasibility and confidence- and priority-based restrictions. These differences concern the action-policy inputs. NAP itself receives $\widetilde{O}_{ij}$ during both training and execution; complete inputs O_j are used only by its frozen distribution teacher. At deployment, only π and f are learned components. Priorities, confidence gating, conflict restrictions, and traffic-field construction remain rule-based, and no CS-PIBT teacher is invoked.

A.8 Training Hyperparameters

Table 4 summarizes the three training stages. Each stage freezes the pretrained MAPF-GPT backbone and optimizes stage-specific LoRA adapters; policy adaptation additionally updates the new token embeddings. LoRA modifies four projection modules in each of eight Transformer blocks: two in self-attention and two in the feed-forward MLP, giving 32 adapted modules. We use AdamW with $\beta = (0.9, 0.95)$, zero weight decay, gradient clipping at 1.0, and no LoRA dropout.

Recovery training uses Warehouse, Maze, and Random sampling weights 0.5:2:0.2, corresponding to 18.5%, 74.1%, and 7.4%, respectively, with WAIT balancing across map families. Dataset sizes in Table 4 refer to collected examples rather than cumulative training draws. Conflict examples constitute 5.38% of the collected NAP pairs and are oversampled to account for 50% of sampled

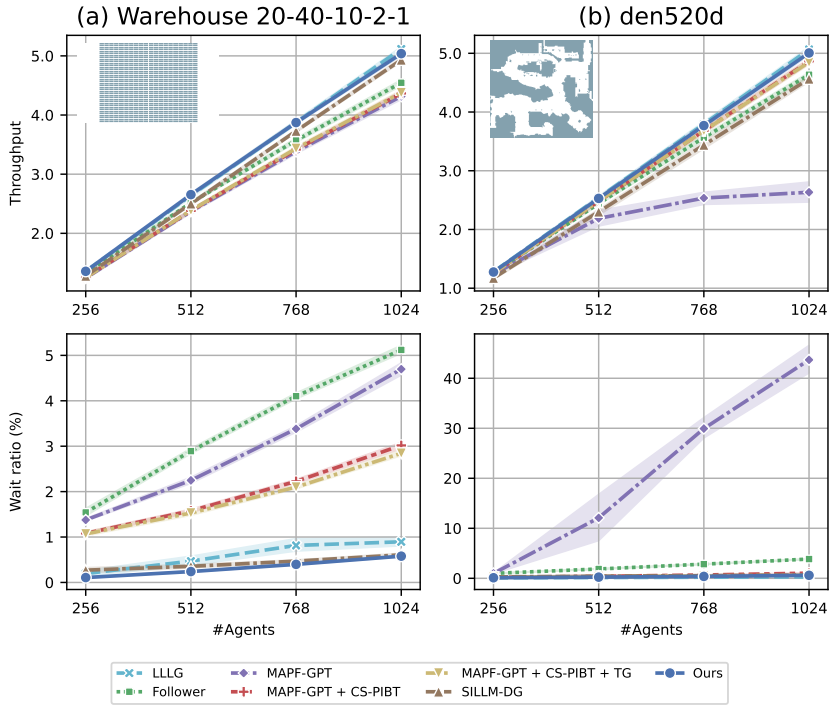


Figure 9: Large-scale evaluation on Warehouse 20-40-10-2-1 and den520d with 256–1024 agents. Rows show throughput and wait ratio, respectively. Curves average 10 runs per setting over $T = 512$ timesteps; shading denotes 95% confidence intervals. MAPF-GPT + CS-PIBT + TG denotes the traffic-guided CS-PIBT configuration used as PACT's adaptation teacher.

training pairs. This reweighting changes their training frequency, not their prevalence in the collected dataset. The reverse-KL loss in Eq. 12, $D_{\mathrm{KL}}(P_{ij} \| Q_j)$, is evaluated on these reweighted minibatches.

A.9 One-Step Coordination Pipeline

Fig. 3 illustrates PACT's one-step coordination. Before action restriction, mobility counts $m_J = 2$, $m_K = 3$, and $m_I = 4$ establish $J \succ K \succ I$, giving $c_{IJ} = c_{IK} = \mathrm{COMMIT}$ in agent I's input. NAP then predicts P_{IJ} and P_{IK} from reconstructed observations and samples $\tilde{a}_{IJ}$ and $\tilde{a}_{IK}$ using the target-keyed draws in Appendix A.3. Both samples enter O_I as coordination context; a sample enters $\mathcal{H}_I$ only if $P_{Ij}(\tilde{a}_{Ij}) > \tau_I$ for $j \in \{J, K\}$. Both illustrated predictions pass this gate. The policy subsequently decodes $a_I^0 = \mathrm{LEFT}$ from the prediction-conditioned input O_I. The first three panels illustrate these components rather than their strict execution order.

In **(IV)**, the retained rightward predictions of J and K exclude LEFT and DOWN, respectively, from agent I's admissible set, leaving $\mathcal{U}_I = \{\mathrm{UP}, \mathrm{RIGHT}, \mathrm{WAIT}\}$. Since $a_I^0 \notin \mathcal{U}_I$ and $\mathcal{U}_I \neq \emptyset$, **(V)** updates the ego fields of O_I with action-wise viability and stall level ℓ_I to form O_I^{rec}, then performs one policy evaluation masked to $\mathcal{U}_I$. Neighbor predictions and traffic guidance are reused without another update. Panel **(VI)** shows I moving right while J and K reach their predicted destinations. Here, the final joint request is conflict-free and requires no collision-handler rejection. This example does not imply a general guarantee: predicted initial actions can differ from neighbors' final requests or realized motions, and residual conflicts remain subject to collision handling.

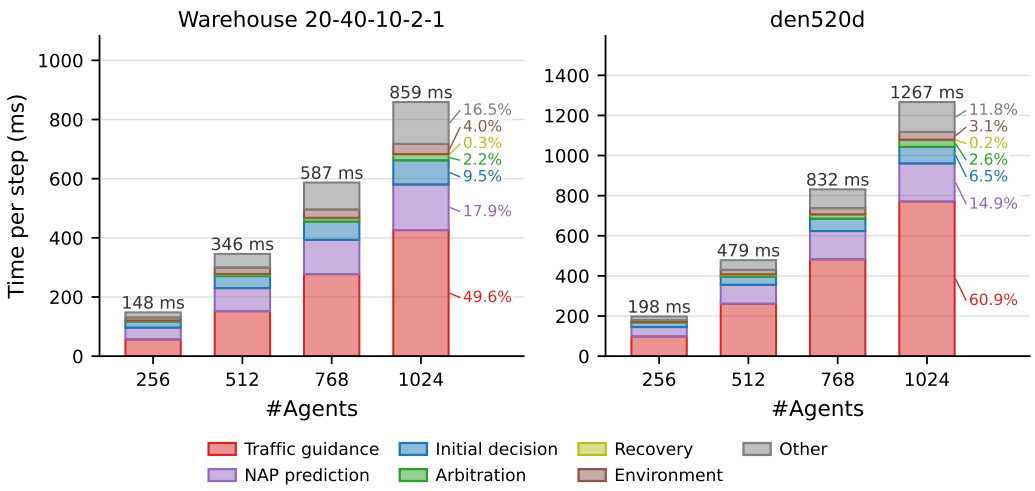


Figure 10: Per-environment-step runtime breakdown on the large-scale maps with 256–1024 agents, averaged over 10 runs per setting. Bars separate Traffic Guidance, NAP prediction, initial decision, arbitration, recovery, environment execution, and other overhead. Percentages indicate component shares at 1024 agents. Traffic Guidance includes field construction and local encoding.

B Additional Evaluation Results

B.1 Large-Scale Lifelong Evaluation

Large-scale performance. Fig. 9 evaluates transfer to two unseen layouts with up to 1024 agents. Warehouse 20-40-10-2-1 is a larger layout from a training map family, whereas den520d belongs to an unseen family. PACT's throughput increases throughout the evaluated agent-count range while its wait ratio remains low. Maximum densities are 4.5% and 3.6%, respectively, so these experiments test larger populations and workspaces rather than higher global density than the standard-scale benchmark. Wait ratio includes both requested WAIT actions and movements rejected by collision handling.

On Warehouse 20-40-10-2-1, all methods increase throughput with agent count, but their waiting behavior differs. Follower and MAPF-GPT accumulate more waiting as density increases. CS-PIBT reduces MAPF-GPT's wait ratio, while MAPF-GPT + CS-PIBT + TG maintains similarly low waiting. PACT achieves higher throughput than both CS-PIBT variants with comparably low waiting, and remains competitive with centralized LLLG at 1024 agents. SILLM-DG also maintains low waiting but achieves lower throughput than PACT.

On den520d, unshielded MAPF-GPT exhibits sharply increasing waiting and saturating throughput. CS-PIBT substantially mitigates this degradation, while MAPF-GPT + CS-PIBT + TG maintains similarly low waiting. Nevertheless, PACT sustains increasing throughput across the evaluated agent counts and remains above the traffic-guided teacher. PACT also remains competitive with LLLG and outperforms Follower and SILLM-DG in throughput on this map. Together, these results support transfer of the complete framework across the evaluated unseen layouts. PACT uses shared sector-occupancy summaries but neither peer-to-peer action exchange nor a full runtime CS-PIBT round; residual conflicts remain subject to environment collision handling.

Runtime breakdown. Fig. 10 reports total wall-clock time per environment step for all agents, rather than per-agent latency. Traffic Guidance includes event-triggered traffic-field updates, goal-change rebuilds, and 11×11 guidance encoding. NAP includes target-observation reconstruction, predictor inference, and keyed sampling; Initial Decision includes the policy forward pass and decoding. Arbitration denotes local priority comparison, confidence gating, conflict screening, and mask construction, not joint CS-PIBT shielding. Recovery includes input construction and the selective additional policy evaluation. Environment measures `env.step()`, including POGEMA

Table 5: NAP prediction quality at the highest-density settings of the five standard-scale maps. Top-1 compares modal actions of P_{ij} and Q_j over FoV neighbor pairs and their contended subset. Precision and recall evaluate conflicts between initial requests using sampled neighbor predictions, with and without confidence gating. All entries are percentages.

Metric	Condition	Warehouse	Random	Maze	Empty Room	Berlin
Top-1 (%)	All	89.0	88.6	86.1	86.5	89.6
	Contended	70.3	69.9	65.0	69.0	65.9
Precision (%)	No Gate	66.2	61.5	64.5	63.6	52.6
	Gate	70.3	65.5	67.5	67.0	56.2
Recall (%)	No Gate	69.3	68.0	64.1	68.0	63.4
	Gate	64.3	64.8	60.5	63.8	59.1

collision handling. Other covers observation encoding, coordination-token construction, tensor preparation and transfer, and Python bookkeeping.

At 1024 agents, runtime reaches 859 ms on Warehouse 20-40-10-2-1 and 1267 ms on den520d. Traffic Guidance accounts for 49.6% and 60.9%, respectively, making aggregate per-agent traffic-field updates the dominant measured cost. Caching avoids unnecessary searches but does not remove their map-wide computation. NAP contributes 17.9% and 14.9%, while initial decoding contributes 9.5% and 6.5%, despite batched neural inference. Local arbitration occupies 2.2–2.6%, and recovery 0.2–0.3%, of total runtime at this scale. These small shares concern screening and amortized recovery, not the complete prediction-based coordination pipeline or the cost of an individual recovery call. The breakdown identifies Traffic Guidance as the main bottleneck in the measured implementation, while showing that throughput scalability does not imply constant decision latency.

B.2 Evaluation of Neighbor Action Prediction

Prediction quality and confidence gating. Table 5 compares $\arg\max_a P_{ij}(a)$ with $\arg\max_a Q_j(a)$. Top-1 agreement is 86.1–89.6% over FoV neighbor pairs, falling to 65.0–70.3% for contended pairs, whose initial requests a_i^0 and a_j^0 induce a vertex or swap conflict. Thus, interaction-critical cases are harder than the aggregate population. This metric compares distribution modes, not the sampled actions $\tilde{a}_{ij}$ and a_j^0. Deployment instead restricts predictions to $j \in \mathcal{N}_i$ with $j \succ i$.

For conflict detection, ground truth uses (a_i^0, a_j^0), while the predicted label substitutes only the neighbor request, using $(a_i^0, \tilde{a}_{ij})$. No Gate considers every sampled prediction; Gate retains a sample only when $P_{ij}(\tilde{a}_{ij}) > \tau_i$. Gating raises precision by 3.0–4.1 percentage points and lowers recall by 3.2–5.0 points. Gated precision remains 56.2–70.3% and recall 59.1–64.8%, so retained hypotheses still contain false positives and omit genuine conflicts. Gating therefore trades coverage for reliability rather than uniformly improving prediction accuracy. It supports selective use of predictions for restrictions: deployed $\mathcal{H}_i$ also requires higher priority, while sampled higher-priority actions remain coordination context regardless of gating. These diagnostics concern initial requests before recovery and collision handling, not realized movements.

B.3 Analysis of CS-PIBT Coordination

Beyond Traffic Guidance and CS-PIBT. Fig. 11 tests whether adding TG and runtime shielding to the frozen pretrained policy reproduces PACT's performance. Starting from MAPF-GPT, +TG only replaces the cost-to-go input with congestion-aware guidance, while +CS-PIBT only shields the policy's joint action preferences. +TG & CS-PIBT combines both mechanisms and corresponds to the teacher used for policy adaptation. None of these pretrained-policy variants includes PACT's policy adaptation, NAP-conditioned decisions, or selective recovery. Ours denotes the complete adapted framework without runtime CS-PIBT.

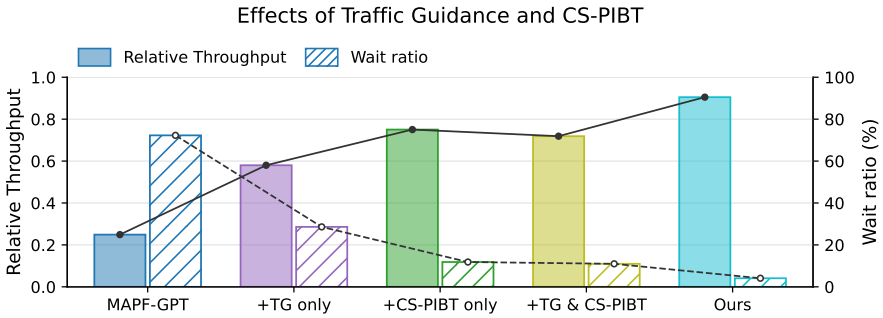


Figure 11: Traffic Guidance (TG), CS-PIBT, and their combination at the highest-density settings of the five standard-scale maps. Bars macro-average per-map relative throughput, normalized by RHCR-PBS, and wait ratio (%). Maps are weighted equally. TG changes the pretrained MAPF-GPT cost-to-go input; Ours denotes full PACT without runtime CS-PIBT.

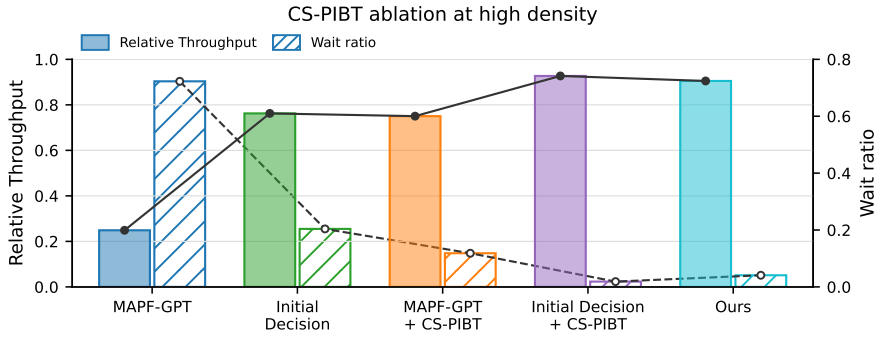


Figure 12: Macro-averaged relative throughput and wait ratio at the five standard-scale maps' highest densities, with 20 runs per setting. Initial Decision uses PACT's prediction-conditioned initial preferences without selective recovery; +CS-PIBT shields the corresponding preferences. Ours is full PACT. Relative throughput uses RHCR-PBS as the per-map reference; wait ratio is displayed as a fraction.

Both TG and CS-PIBT individually improve relative throughput and reduce waiting over MAPF-GPT. Combining them slightly reduces waiting further but does not improve throughput over CS-PIBT alone; its displayed macro-average relative throughput is slightly lower. Full PACT achieves the highest relative throughput and lowest wait ratio among these variants. Thus, simply combining congestion-aware inputs with runtime shielding does not reproduce PACT's observed performance. This comparison evaluates the complete framework and does not separately attribute the additional gain to adaptation, prediction, or recovery. Relative throughput is normalized within each map before averaging, rather than computed from a ratio of pooled throughputs.

Interaction with CS-PIBT coordination. Fig. 12 compares unshielded and shielded initial preferences from MAPF-GPT and PACT. Initial Decision uses predicted neighbor-action context and Traffic Guidance to produce a_i^0 but omits selective recovery. +CS-PIBT applies joint shielding to the corresponding policy preferences, rather than to PACT's recovered requests. Full PACT instead screens candidates using confidence-qualified higher-priority hypotheses and performs at most one recovery evaluation when $a_i^0 \notin \mathcal{U}_i$ and $\mathcal{U}_i \neq \emptyset$. All variants remain subject to environment collision handling.

Initial Decision increases throughput and reduces waiting relative to MAPF-GPT, attaining relative throughput comparable to MAPF-GPT + CS-PIBT, although its wait ratio remains higher. Adding CS-PIBT to Initial Decision yields the highest relative throughput and lowest wait ratio in this comparison, indicating that improved initial preferences remain complementary to joint shielding. Full PACT approaches this shielded variant's throughput but has a higher wait ratio. Selective recovery therefore provides a high-throughput alternative without a full runtime CS-PIBT round, not an

equivalent collision-free substitute. Its additional policy evaluations are restricted to blocked initial requests with admissible alternatives.

B.4 Analysis of Conflict Resolution

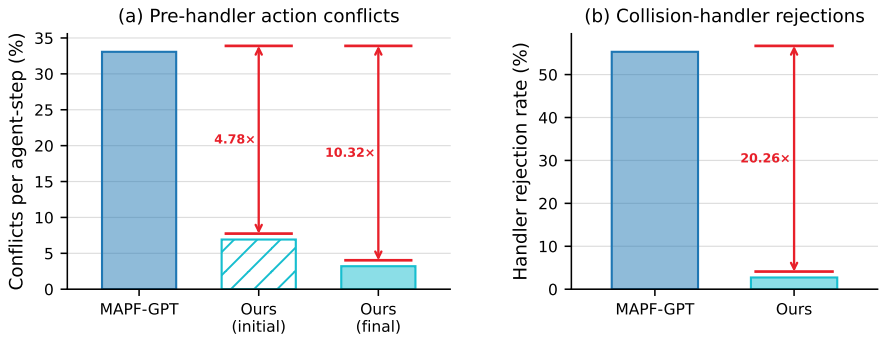


Figure 13: Action conflicts and collision-handler intervention at high density. (a) Pairwise vertex and swap conflicts per agent-step, shown as percentages, for MAPF-GPT and PACT's initial and final requests before collision handling. (b) Fraction of all agent requests rejected by POGEMA. Results macro-average the five standard-scale maps at their highest densities, with 20 runs per setting and equal map weights.

PACT does not guarantee collision-free joint requests, unlike a full CS-PIBT shielding round. Fig. 13(a) measures conflicts in the initial requests a_i^0 and final requests a_i before environment intervention. Relative to MAPF-GPT, the reported conflict rate falls to $1/4.78$ after initial decoding and $1/10.32$ after selective recovery. Thus, PACT produces more compatible requests before collision handling, with recovery further reducing the aggregate rate. The initial-stage comparison includes policy adaptation, prediction-conditioned context, and Traffic Guidance; it does not isolate NAP's contribution. Final requests include preserved initial actions, recovered actions, and WAIT fallbacks, rather than only agents receiving a second policy evaluation.

Fig. 13(b) reports rejected requests rather than conflicting pairs. Full PACT reduces this rejection rate to $1/20.26$ of MAPF-GPT's value. This metric also differs from wait ratio, which includes requested WAIT actions. Together, the measurements support reduced reliance on collision-handler intervention, rather than increased rejection compensating for incompatible requests. They do not remove that reliance: residual conflicts remain, and the handler is still needed to resolve them at execution.

B.5 Analysis of Traffic Guidance

Sector partition examples. Fig. 14 illustrates the partitions underlying the occupancy summaries. Warehouse follows its aisle structure, while Random, Maze, Empty Room, and Berlin use Infomap with preferred module count n, the number of agents. Partitions are precomputed for each map–agent-count configuration and remain fixed throughout an episode. Each traversable cell belongs to exactly one nonempty sector $S \in \mathcal{S}$; occupancy changes do not change sector boundaries.

Table 6: Occupancy-summary quantization at the highest densities of the five standard-scale maps. Capacity-normalized counts are uniformly quantized to 2, 4, or 8 bits; Ours retains unquantized counts. Relative throughput is normalized by RHCR-PBS within each map before macro-averaging. Wait ratio is macro-averaged without reference normalization. Both metrics use equal map weights and 20 runs per setting.

Metric	No TG	2-bit	4-bit	8-bit	Ours
Relative throughput	0.646	0.907	0.908	0.908	0.905
Wait ratio (%)	31.3	4.01	4.23	4.23	4.22

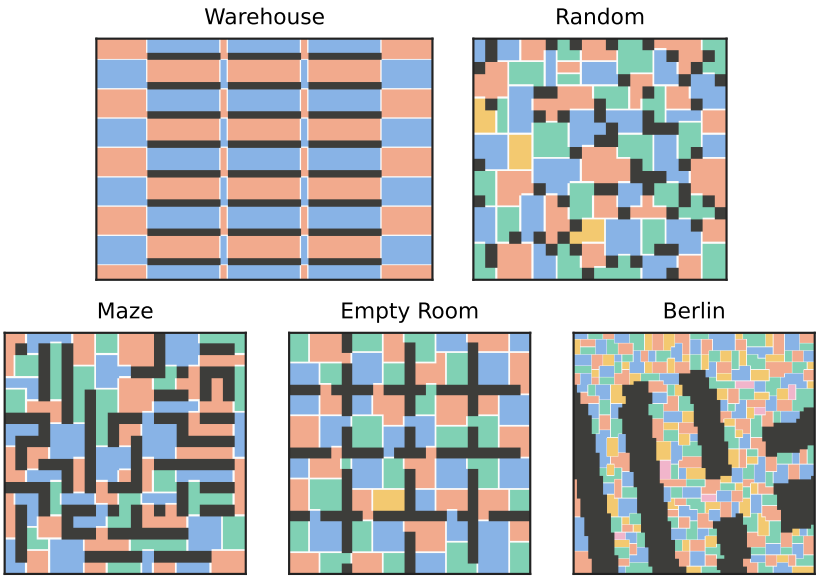


Figure 14: Sector partitions on the five standard-scale maps. Warehouse uses manually defined topological sectors; the other maps use Infomap with an agent-count-dependent preferred module count. Black cells are obstacles. Colors distinguish sectors, not occupancy or congestion, and are reused across the map.

Quantized traffic summaries. Table 6 tests the precision of the cached occupancy summaries K_i. Each sector count is normalized as $n_i(S)/|S|$ before uniform quantization. Only summary precision changes; the other PACT components remain fixed. This differs from quantizing the local cost-to-go tokens after field construction. No TG removes traffic-aware guidance while retaining local coordination and recovery; Ours uses the original counts.

Removing TG lowers relative throughput from 0.905 to 0.646 and raises waiting from 4.22% to 31.3%. The 2-bit variant reaches 0.907 relative throughput and 4.01% waiting, close to the unquantized result. Four and eight bits yield 0.908 and 4.23%, providing no consistent advantage across both metrics. These point estimates support coarse occupancy summaries in the tested settings, not a performance gain from quantization. Thus, retaining distant congestion information matters more here than preserving fine occupancy precision. These summaries still require a shared occupancy source; quantization changes the supplied values rather than making traffic sensing purely local. The experiment supports a compact representation but does not measure the communication cost of collecting, distributing, or refreshing it.

Sensitivity to Traffic Guidance design. Fig. 15(a) evaluates the effect of the congestion-penalty strength C while keeping all other PACT components fixed. We vary $C \in \{0.25, 0.5, 1, 1.5, 2, 3\}$, with $C = 1$ used throughout the main experiments. Relative throughput peaks at $C = 1$ among the evaluated values. Smaller values weaken the influence of occupancy relative to unit movement cost, with a pronounced degradation at $C = 0.25$. Increasing C beyond 1 also progressively reduces throughput, consistent with stronger congestion penalties favoring longer detours through less occupied sectors. These results support $C = 1$ as the best tested aggregate setting, without implying the same optimum for every map.

Fig. 15(b) compares PACT's event-triggered occupancy refresh with synchronous periodic updates every $k \in \{1, 2, 4, 8, 16, 32\}$ steps. The event-triggered rule refreshes an agent's cached occupancy snapshot when it enters a new sector and has a measured mean update interval of 3.4 steps. Its relative throughput remains close to that of frequent periodic updates, whereas performance declines as the periodic update interval increases, with the degradation becoming pronounced at 16 and 32 steps. Here, the update period refers only to occupancy-snapshot refreshes: goal changes may independently rebuild the congestion-aware cost-to-go field using the currently cached snapshot.

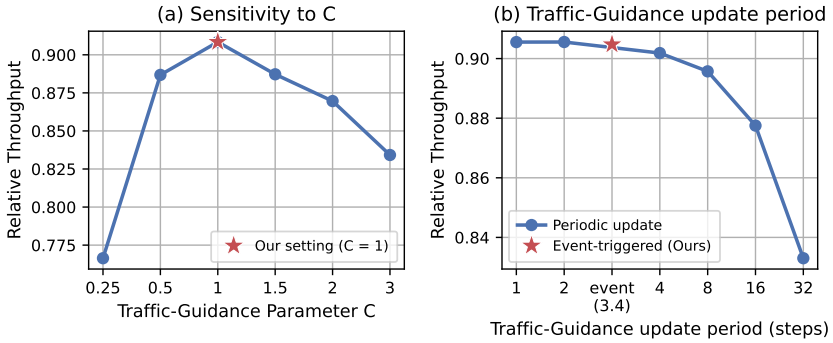


Figure 15: Sensitivity analysis of Traffic Guidance at the highest-density settings of the five standard-scale maps. (a) Relative throughput as the congestion-penalty strength C is varied, with $C = 1$ denoting the default setting. (b) Comparison of synchronous periodic occupancy-snapshot updates every k steps with PACT's event-triggered refresh on ego sector transitions; the event marker corresponds to a measured mean update interval of 3.4 steps rather than a fixed period. Relative throughput is normalized by RHCR-PBS within each map and macro-averaged with equal map weights over 20 runs per setting. All other PACT components are fixed.

These results support sector transitions as an effective asynchronous refresh trigger without requiring a fixed globally synchronized update period.

Traffic Guidance update analysis. Fig. 15 (b) compares synchronous snapshot refreshes every $k \in \{1, 2, 4, 8, 16, 32\}$ steps with PACT's agent-specific event trigger. After initialization, PACT refreshes K_i when agent i enters a different sector. Its observed mean interval is 3.4 steps. Snapshot refreshes must be distinguished from field rebuilds: a goal change also recomputes D_i, but reuses cached occupancy. Thus, 3.4 steps is neither a bound on snapshot age nor the interval between all field computations.

Frequent periodic updates yield similar relative throughput, whereas longer periods reduce performance, especially at 16 and 32 steps. Event-triggered refreshes remain close to frequent periodic updates despite their asynchronous timing. This supports sector transitions as a useful refresh trigger in the evaluated settings. It does not guarantee fresh information for an agent remaining in one sector, nor does mean refresh frequency alone establish end-to-end communication or runtime savings.

Table 7: Sensitivity to the preferred sector-to-agent ratio M/n on Random, Maze, Empty Room, and Berlin at their highest densities. Infomap targets $M = rn$ modules for $r \in \{0.25, 0.5, 1, 2\}$. Relative throughput uses RHCR-PBS as the per-map reference; both metrics are macro-averaged over 20 runs per setting. Warehouse is excluded because its sectors are manually defined.

Metric	**0.25**	**0.5**	**1.0 (Ours)**	**2.0**
Relative throughput	0.890	0.894	0.882	0.883
Wait ratio (%)	5.56	4.44	4.08	4.67

Sensitivity to sector granularity. Table 7 varies the preferred Infomap module count $M = rn$, where n is the agent count. This is a target count, not a requirement that the realized $|\mathcal{S}|$ equal M. Other components remain fixed, and each resulting partition is fixed throughout its evaluation episodes. Warehouse is excluded because its manual partition does not use this construction.

Relative throughput varies from 0.882 to 0.894, while waiting ranges from 4.08% to 5.56%. The coarsest target, $r = 0.25$, yields the most waiting. The default $r = 1$ minimizes waiting, whereas $r = 0.5$ achieves the highest relative throughput. Increasing r from 1 to 2 slightly raises relative throughput from 0.882 to 0.883 but increases waiting from 4.08% to 4.67%. The throughput differences are small, but the waiting differences show that partition choice still affects stationary

behavior. These aggregate results support $r = 1$ as a low-waiting default while showing that finer partitions do not monotonically improve performance.

B.6 Effect of Prediction-Draw Coupling

Table 8: Prediction-draw coupling at the highest densities of the five standard-scale maps. Ours shares the target's initial-action draw; Independent separates prediction and target draws; Obs-Dependent also uses observer-specific prediction draws; GT-Independent substitutes Q_j with an independent draw. Relative throughput is normalized by RHCR-PBS per map; both metrics are macro-averaged with equal map weights.

Metric	Ours	Independent	Obs-Dependent	GT-Independent
Relative throughput	0.905	0.906	0.906	0.917
Wait ratio (%)	4.22	4.42	4.27	2.71

Table 8 separates two sampling choices. Ours uses the same target-keyed draw for $\tilde{a}_{ij} \sim P_{ij}$ and $a_j^0 \sim Q_j$. Independent separates these draws but retains a common target-keyed prediction draw across observers. Obs-Dependent additionally includes observer identity in the prediction key. Sharing a draw aligns observer predictions only when their distributions and decoding conventions match; it does not make differing reconstructed inputs equivalent. GT-Independent replaces P_{ij} with the oracle initial-action distribution Q_j while retaining independent sampling. It supplies exact probabilities, not the target's sampled initial request or realized movement. Other PACT components remain fixed.

Independent and Obs-Dependent both achieve 0.906 relative throughput, compared with 0.905 for Ours; waiting increases by 0.20 and 0.05 percentage points, respectively. At the level of the reported means, these differences suggest limited aggregate sensitivity to draw coupling in the tested settings. GT-Independent reaches 0.917 relative throughput and 2.71% waiting, compared with Independent's 0.906 and 4.42%. Thus, better distribution information remains useful even without target–prediction coupling. This comparison supports improving distribution estimates without attributing all remaining errors to NAP. Shared draws retain their conditional alignment role described in Appendix A.3, without guaranteeing accurate predictions.

B.7 Sensitivity to Episode Length

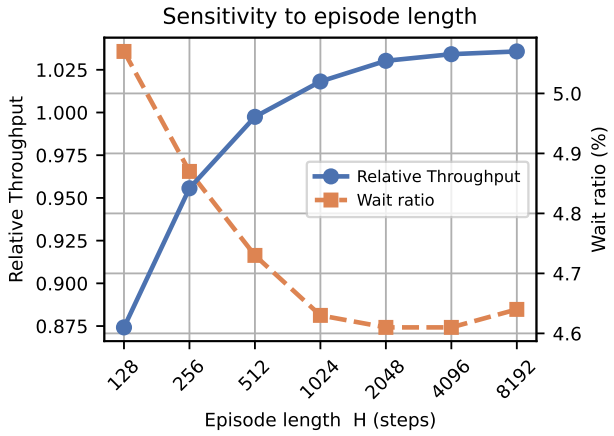


Figure 16: Episode-length sensitivity on Warehouse with 192 agents. Full PACT is evaluated for $T \in \{128, 256, 512, 1024, 2048, 4096, 8192\}$ steps. Relative throughput and wait ratio are averaged over 10 runs per horizon.

Fig. 16 extends evaluation beyond the main horizon $T = 512$. Reported relative throughput increases with episode length and gradually plateaus. Wait ratio decreases toward approximately

4.6%, remains close to this level from 1024 steps onward, and rises slightly at 8192 steps. Both metrics remain favorable beyond the main evaluation horizon, although the wait-ratio trend is not strictly monotonic.

These results support sustained aggregate performance on the evaluated Warehouse setting up to 8192 steps. The measurements are whole-episode averages and do not reveal within-episode fluctuations. The main benchmark retains $T = 512$ for consistency with the lifelong evaluation protocol of Follower (Skrynnik et al., 2024) and to keep the broader comparison computationally manageable.

B.8 Analysis of Priority

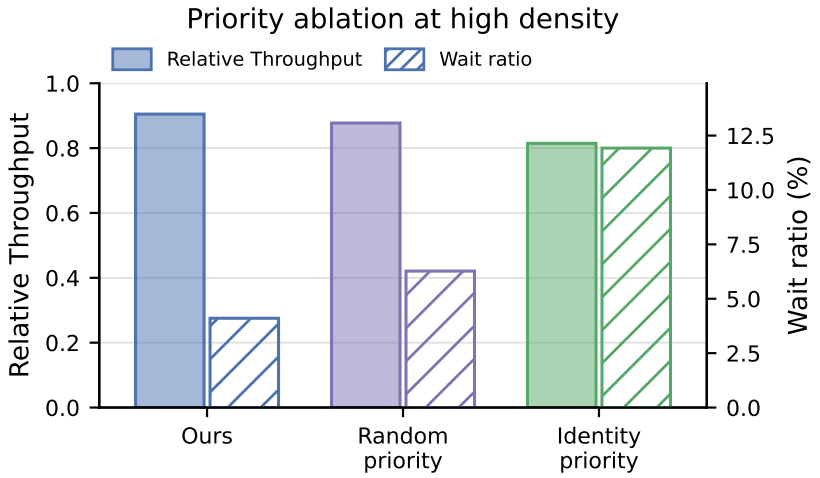


Figure 17: Priority-rule ablation at the highest densities of the five standard-scale maps. Mobility-based, per-timestep random, and fixed identity-based priorities each use a coordination policy retrained under the same configuration; other components are unchanged. Bars macro-average RHCR-PBS-normalized relative throughput and wait ratio across maps, with 20 runs per setting and equal map weights.

Fig. 17 compares three reproducible priority rules. Ours gives precedence to smaller terrain-feasible movement counts m_i, with ties resolved by the public hash and agent ID. This count excludes WAIT and treats cells currently occupied by other agents as unavailable; unlike the prediction-dependent flexibility score F_i, it does not use predicted neighbor motion. Random priority uses a publicly reproducible ordering resampled each timestep, whereas Identity priority uses a fixed ID ordering. Each coordination policy is retrained under the same training configuration, with other components unchanged, so the comparison is not a test-time-only replacement of the ordering.

Mobility-based priority yields the highest relative throughput and lowest wait ratio. Random priority performs worse on both metrics, and Identity priority degrades further. Unlike these geometry-independent rules, mobility-based ordering gives precedence to agents with fewer immediately available alternatives due to local geometry and current occupancy. The results are consistent with this information providing a more useful yielding order under the evaluated retraining protocol. Fixed identity ordering can repeatedly favor the same agents, whereas terrain-based ordering responds to the constraints at their current positions. This supports using local geometry to guide yielding under the stated training protocol. The aggregate metrics do not, however, establish starvation or quantify agent-level fairness.

C Evaluation Details and Reproducibility

This appendix specifies the benchmark maps, task-generation rules, baseline configurations, and training–evaluation separation underlying the reported experiments.

Table 9: Benchmark map statistics. Grid dimensions include POGEMA observation padding. Start and goal columns count admissible sampling cells. Maximum density is $n_{\max}/|V_{\text{reach}}|$ using the listed reachable-cell count.

Map	Grid	Reachable cells	Start cells	Goal cells	Agent counts n	Max. density
Warehouse	43×56	1278	192	480	64, 96, 128, 160, 192	15.0%
Random	30×31	356	356	356	16, 24, 32, 48, 64	18.0%
Maze	31×31	305	305	305	16, 24, 32, 48, 64	21.0%
Empty Room	33×33	424	424	424	16, 24, 32, 48, 64	15.1%
Berlin	74×74	2837	2837	2837	64, 128, 192, 256	9.0%
Warehouse 20-40-10-2-1	133×331	22599	22599	22599	256, 512, 768, 1024	4.5%
den520d	267×266	28178	28178	28178	256, 512, 768, 1024	3.6%

C.1 Benchmark Maps and Task Generation

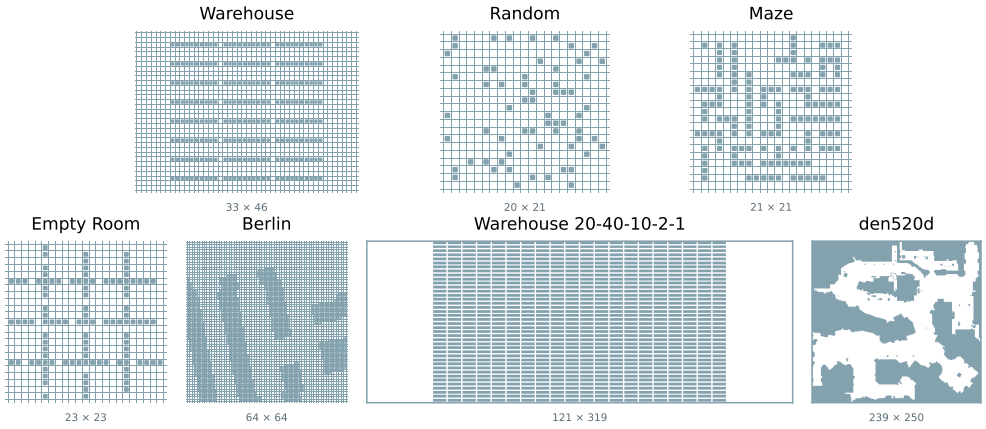


Figure 18: Evaluation layouts. Warehouse, Random, Maze, Empty Room, and Berlin form the standard-scale benchmark; Warehouse 20-40-10-2-1 and den520d support large-scale evaluation. Gray cells denote obstacles. Dimension annotations refer to the visualized layouts, whereas Table 9 lists loaded-grid dimensions. Random, Maze, and Empty Room show representative instances.

Table 9 reports the POGEMA-loaded grid dimensions, including observation padding with $R_{\text{obs}} = 5$. Density is defined using the reachable task workspace rather than every free entry in the padded array:

$$\rho = \frac{n}{|V_{\text{reach}}|},$$

where n is the agent count and $V_{\text{reach}} \subseteq V$ contains the reachable traversable cells retained for task generation. Disconnected padding cells are excluded even if encoded as free. Density therefore measures workspace occupancy, not the fraction of admissible start or goal cells occupied by agents.

The standard Warehouse is the only environment with distinct start and goal roles: starts are selected from 192 workstation cells and goals from 480 goal-designated shelf cells, limiting simultaneous initialization to 192 agents. Other maps admit starts and goals throughout their reachable task regions.

Lifelong runs immediately reassign a goal upon completion using `on_target="restart"`, drawing from the same admissible goal distribution. On disconnected maps, assigned goals re-

Table 10: Baseline evaluation configurations. Time limits refer to wall-clock cutoffs, not the episode horizon T; None indicates no imposed cutoff.

Method	Key configuration	Time limit
MAPF-GPT	Released 6M-parameter policy π_0, also used as PACT's pretrained backbone.	None
MAPF-GPT + CS-PIBT	Pretrained policy preferences with CS-PIBT shielding at every timestep.	None
MAPF-GPT + CS-PIBT + TG	Same pretrained policy with Traffic Guidance and per-step CS-PIBT; PACT's adaptation teacher.	None
Follower	Released checkpoint and recommended evaluation configuration.	None
SILLM-DG	Released dynamic-guidance checkpoint with inter-agent communication and CS-PIBT.	None
LLLG	`lacam_horizon`=10 and `lg_window`=20.	Deadline-based
RHCR-PBS	PBS with planning window $w = 20$ and replanning interval $h = 5$ steps.	10 s cutoff

main within the agent's reachable component. The POGEMA configuration uses `soft` collision handling, `observation_type=MAPF`, and `obs_radius=5`, giving an 11×11 FoV. Rejected moves leave agents stationary and contribute to wait ratio along with requested WAIT actions.

Unless otherwise stated, episodes last $T = 512$ steps. Each standard-scale setting uses 20 runs; Random, Maze, and Empty Room use five map seeds with four runs each. Large-scale settings use 10 runs.

C.2 Baseline Implementations and Evaluation Settings

Table 10 summarizes the implementations and evaluation configurations of all baselines. We use released checkpoints and recommended settings whenever available. PACT, MAPF-GPT, MAPF-GPT + CS-PIBT, and Follower are evaluated with POGEMA 1.3.2a4, while SILLM-DG retains its released POGEMA 1.2.2 environment. LLLG and RHCR-PBS use their standalone C++ implementations. All experiments are run on an AMD Ryzen 9 7900X CPU and an NVIDIA GeForce RTX 5070 Ti GPU. Wall-clock-dependent search methods are evaluated on an otherwise idle machine. Runtime reports total wall-clock time per environment step for all agents, including method-specific inference, coordination, or planning, rather than per-agent latency.

Baseline configurations. MAPF-GPT (Andreychuk et al., 2025) provides the pretrained policy backbone. MAPF-GPT + CS-PIBT (Veerapaneni et al., 2024) applies runtime collision shielding to its action preferences, while MAPF-GPT + CS-PIBT + TG additionally incorporates our Traffic Guidance and serves as the teacher for PACT adaptation. These variants isolate the effects of runtime shielding and traffic-aware guidance under the same pretrained policy. Follower (Skrynnik et al., 2024) represents decentralized lifelong planning, whereas SILLM-DG (Jiang et al., 2025) combines dynamic guidance, inter-agent communication, and CS-PIBT. LLLG (Arita & Okumura, 2026) provides a centralized search-based lifelong solver, and RHCR-PBS (Li et al., 2021; Ma et al., 2019) serves as the centralized planning reference. Together, these baselines span learned local policies, runtime shielding, communication-based coordination, and centralized search.

Planner settings and hardware. We retain recommended baseline configurations unless stated otherwise. RHCR-PBS uses a planning window of 20 timesteps, a replanning interval of 5 timesteps, and a 10 s cutoff (Li et al., 2021). Evaluations run on an AMD Ryzen 9 7900X CPU and an NVIDIA GeForce RTX 5070 Ti GPU.

C.3 Training–Evaluation Separation

PACT-stage training and evaluation are separated at both the instance and map-family levels. For map families used during coordination adaptation, recovery adaptation, or NAP distillation, generated training and evaluation instances use disjoint map-seed namespaces. This separation also applies to instances used for NAP data collection. These guarantees concern only PACT's adaptation and distillation stages and do not establish disjointness from the pretraining data of externally released checkpoints.

For evaluation within familiar map families, the standard `wfi_warehouse` layout is excluded from the Warehouse training pool, and Random and Maze use evaluation seeds disjoint from their training seeds. Warehouse 20-40-10-2-1 is also fully held out, providing a larger unseen layout from the familiar Warehouse family. In contrast, Empty Room, Berlin, and den520d belong to map families absent from PACT-stage training and therefore evaluate transfer to unseen environment families.

Empty Room instances are generated using the Room-map generator from the HMAGAT benchmark implementation; no HMAGAT model or checkpoint is used. Map-generation seeds and episode seeds are treated as separate sources of randomness. The former determine generated layouts, whereas episode seeds control episode-level stochasticity and are also used by PACT's public priority tie-breaking and reproducible decoding keys.