REINFORCEMENT LEARNING FOR LARGE-SCALE LIFELONG MULTI-AGENT PATH FINDING WITH SHARED SPATIAL FIELDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale applications of lifelong multi-agent path finding (LMAPF) call for computationally inexpensive movement decisions under changing traffic conditions and across varied layouts. We propose a navigation policy in which a convolutional network computes a shared spatial field over the map once per timestep and a compact goal-conditioned operator derives agents' movement preferences. The complete policy is trained end-to-end using average-reward reinforcement learning with throughput as the reward, without planner demonstrations or a separate online path planner. For structured and unstructured layouts, we train one policy per class on mixtures of small maps and deploy each without retraining on larger, unseen layouts and agent populations. Across six layouts with $10{,}000$ agents, our policies achieve throughput comparable to or higher than the strongest evaluated baselines. The proposed method executes each step in less than 10 ms, more than an order of magnitude faster than several established planners.

1 INTRODUCTION

Lifelong multi-agent path finding (LMAPF) coordinates collision-free movement of agents that receive new goals after completing their current ones. Throughput is the average number of goals completed per timestep (Ma et al., 2017; Li et al., 2021). Applications include large robotic logistics systems that continually transport goods and real-time strategy games in which many units move toward changing objectives. Large populations increase the computation needed to update movement decisions, while real-time applications limit the time available for each update (Silver, 2005). Reuse across facilities or game maps also requires handling varied obstacle and passage geometry (Stern et al., 2019). We therefore seek policies that maintain throughput across layouts and populations with low per-step computation cost.

Existing methods use different forms of coordination within this computation limit. RHCR solves windowed planning problems (Li et al., 2021), while WPPL refines windowed plans within an available time budget (Jiang et al., 2024). PIBT resolves conflicts through local priority rules (Okumura et al., 2022); Traffic Flow Optimisation adds congestion-aware guide paths (Chen et al., 2024). Learned policies can replace part of this online planning. SILLM combines per-agent neural inference, spatial communication and global guidance to coordinate $10{,}000$ agents (Jiang et al., 2025). When agents' local observations overlap, per-agent neural inference can repeatedly process the same map and traffic information.

We use a shared spatial field to reuse this computation across agents. A convolutional network processes the map and current traffic once per timestep, producing scores for movement and waiting at each location. Agents use these shared scores in a goal-conditioned route computation. Neural computation is performed over the map, while each agent's route computation is limited to a fixed number of departures (Fig. 2).

A soft shortest-path distribution derived from the map and each agent's goal supplies a geometric reference that favours shorter routes while retaining alternatives. The route operator combines this reference with the shared field, accumulates scores along short route continuations, and computes the resulting next-move probabilities. Parallel Sinkhorn iterations then adjust these probabilities for

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.

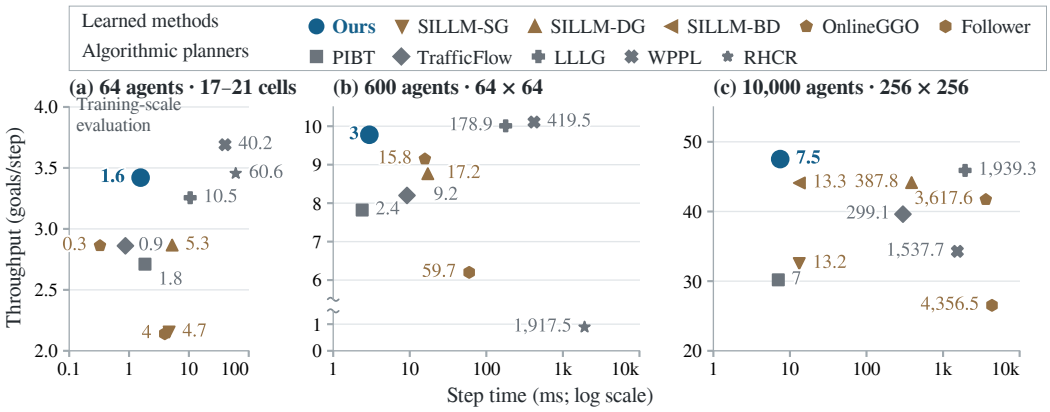


Figure 1: **Throughput and execution cost across random-map scales.** Our unstructured policy, trained on small tiles, is reused without retraining at all three scales. Panel (a) averages 128 connected random maps with 17–21 cells per dimension (Appendix C). WPPL uses 500 LNS iterations in (a) and 40,000 in (b,c). Labels give ms per step; cost axes are logarithmic. Throughput axes differ; (b) omits 1.5–5.5 goals/step and compresses the lower segment.

competing demand for destination cells. The spatial field, route operator and allocation define one navigation policy.

We train this policy end-to-end using average-reward reinforcement learning. The population is treated as one controlled system, with the total number of completed goals at each step as its reward. Gradients through route reweighting and destination allocation train the convolutional network. Training requires no planner demonstrations or additional task reward shaping (Sec. 3.4).

We train one policy for structured layouts and one for unstructured layouts, each on a fixed mixture of small maps with roughly a hundred agents per map. The structured mixture contains rack and sortation layouts; the unstructured mixture contains random maps, dumbbell layouts and street crops. Each policy is applied without retraining to larger layouts excluded from training and to different agent populations (Sec. 4).

Figure 1 summarises throughput and execution cost across random-map scales. Across three training seeds per class at 25k updates, the unstructured policies reach 95–99% of RHCR's throughput on small random maps with 8–64 agents and are reused on 256×256 maps with 2000–20,000 agents. Across the six 10,000-agent layouts, the policies attain throughput comparable to or higher than the highest-throughput evaluated baseline on each layout, with execution below 10 ms per step and 68–320$\times$ faster than that baseline (Sec. 4.1).

2 Problem Formulation

Continuing multi-agent control. We model LMAPF as a continuing Markov decision process on a four-connected grid $G = (V, E)$ with obstacles removed. For a fixed map and population of N agents, the state s_t contains agent positions, goal information, previous moves, elapsed-task counters and the task-stream state. The action is a joint movement proposal $\mathbf{a}_t = (a_{1,t}, \ldots, a_{N,t})$. At cell x, let $\mathcal{M}(x)$ be the moves to adjacent free cells and $\mathcal{A}(x) = \mathcal{M}(x) \cup \{\text{WAIT}\}$ the available actions; x^a denotes action a's destination. Movement follows POGEMA's soft collision rules (Skrynnik et al., 2025), which prevent agents from occupying the same cell or exchanging positions in one timestep. Completing a goal assigns the next task, so operation continues (Ma et al., 2017; Li et al., 2021). These movement rules and task assignment define the transition law $P(s_{t+1} \mid s_t, \mathbf{a}_t)$.

Throughput objective. The reward r_t counts goals completed by all agents at step t. For a policy π_θ, the objective is long-run throughput for a fixed initial-state distribution,

$$\rho(\theta) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{T} r_t \right]. \tag{1}$$

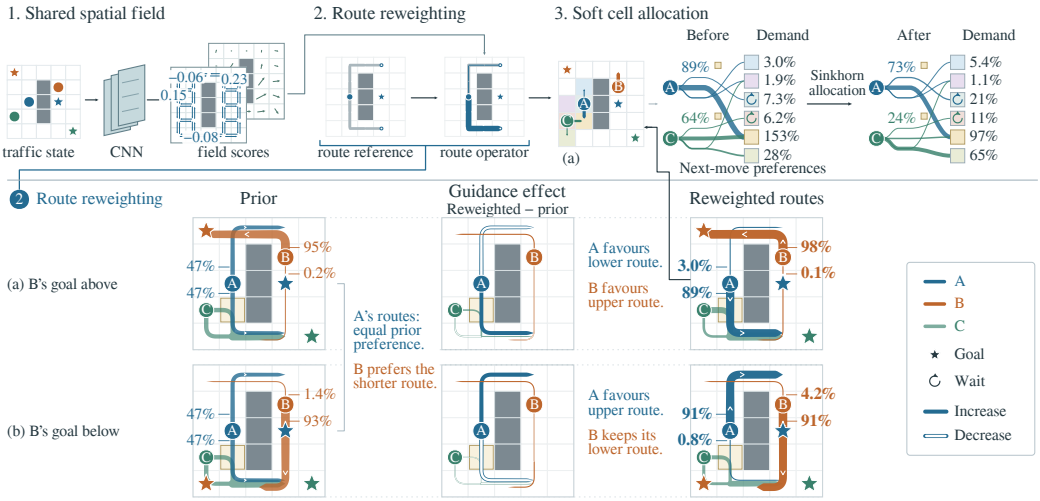


Figure 2: **Policy.** A CNN computes a shared spatial field; route reweighting and Sinkhorn allocation produce next-move probabilities. **Example.** Changing B's goal between (a) and (b) reverses A's preferred route, with A's reference fixed. Route widths give edge-traversal probabilities at each departure under P^0 and Q; the middle column gives their difference, and labels give first-action probabilities. The allocation panel compares q and π, redistributing A's and C's demand for the gold cell in (a).

The limit is assumed to exist. This continuing task has no terminal configuration in which all goals are complete. Average-reward reinforcement learning targets the sustained completion rate; its value estimates describe future reward relative to that rate (Mahadevan, 1996; Sutton & Barto, 2018). Section 3.4 specifies the estimator and treatment of training resets.

Soft shortest-path reference. Before considering traffic, map geometry and a goal g define a reference action distribution. Let $\widehat{D}_g(x)$ be a precomputed soft distance obtained by exponentially weighting the lengths of walks to g. For cells with available movement actions,

$$p^0(\text{WAIT} \mid x, g) = w_0, \qquad w_0 = \frac{e^{-\beta}}{1 + e^{-\beta}},$$
$$p^0(a \mid x, g) = (1 - w_0)\frac{e^{-\beta \widehat{D}_g(x^a)}}{\sum_{b \in \mathcal{M}(x)} e^{-\beta \widehat{D}_g(x^b)}}, \qquad a \in \mathcal{M}(x). \tag{2}$$

We fix $\beta = 3$. The soft distances supply geometric route information beyond a single move, while retaining alternatives to a shortest path. They contain no traffic information or learned parameters. Appendix A.1 gives the distance construction and boundary rules.

3 Approach

The centralised policy computes a shared spatial field, goal-conditioned route probabilities and a destination allocation (Fig. 2). A convolutional network processes the scene once per timestep. The route operator uses its output and each agent's goals to compute action probabilities, which allocation adjusts for competing demand for cells. All learned parameters are shared across agents; their number is independent of population size and map dimensions. Appendix A.6 specifies the example in Figure 2.

3.1 Shared spatial field

Let $O_t = O(s_t)$ be an aggregate spatial representation of the full state s_t, containing map geometry, agent positions and previous moves, current and queued goal information, and aggregates of reference routes. The convolutional network W_{θ_W} computes the shared field $\Phi_t = W_{\theta_W}(O_t)$ once per timestep. Its outputs supply scores for movement and waiting. For a continuation of H departures,

$h \in \{0, \ldots, H-1\}$ indexes the departure depth. A base action-score vector is shared across depths, with additive corrections for later departures. Scores enter the route probabilities exponentially, so higher accumulated scores favour the corresponding continuations. All depths use the current scene; the field is refreshed at the next timestep. The network has 39,615 parameters, with six further scalars in the route operator and allocation. Appendix A.2 specifies the observations, architecture and field construction.

3.2 Goal-conditioned route operator

Goal-conditioned guidance. Agents with different goals can interpret the same field differently. A route state $z = (x, \varphi)$ records a candidate cell and the active goal phase. Write $p_i^0(a \mid z)$ for Eq. (2) evaluated at agent i's active goal. Each move or wait consumes one departure. Appendix A.1 specifies the goal-phase transitions and continuation boundary. For action a, let $\psi_i(z, a) \in \mathbb{R}^2$ contain two contrasts formed after centring the reference probabilities over valid movement actions. They compare opposite directions and perpendicular axes, and are zero for waiting. The local guidance score is

$$U_{i,h}(z, a) = b_h(x, a) + \gamma_h(x)^\top \psi_i(z, a). \tag{3}$$

Here b_h is a shared action score derived from the field. Four learned coefficients, shared across agents, cells and depths, map the field's waiting score and summed movement scores to the two gains $\gamma_h(x)$. The reference determines the direction of the adjustment; the field determines its strength and sign. At an unobstructed junction, the eastward contrast is $\frac{1}{2}[p_i^0(E \mid z) - p_i^0(W \mid z)]$, so a positive gain reinforces the direction favoured by the reference in the local score. Appendix A.3 gives the contrasts and field-to-gain map, including obstacle masks.

Route reweighting. For a continuation $\xi = (a_0, \ldots, a_{H-1})$ with successive route states z_h, the reference law and cumulative guidance are

$$P_i^0(\xi \mid z_0) = \prod_{h=0}^{H-1} p_i^0(a_h \mid z_h), \qquad S_i(\xi) = \sum_{h=0}^{H-1} U_{i,h}(z_h, a_h). \tag{4}$$

The operator reweights these routes at learned temperature $\tau_f > 0$ and returns their first-action marginal,

$$Q_i(\xi) = \frac{P_i^0(\xi) \exp(S_i(\xi)/\tau_f)}{\sum_{\xi'} P_i^0(\xi') \exp(S_i(\xi')/\tau_f)}, \tag{5}$$

$$q_i(a) = \sum_{\xi : a_0 = a} Q_i(\xi). \tag{6}$$

This is KL-regularized route choice (Todorov, 2006; Kivimäki et al., 2014),

$$Q_i = \arg\max_Q \left\{ \mathbb{E}_Q[S_i(\xi)] - \tau_f \, \mathrm{KL}(Q \| P_i^0) \right\}. \tag{7}$$

The optimisation ranges over distributions on the reference support. It balances cumulative guidance against a KL penalty for departing from the geometric reference. Throughput remains the objective used to train the policy.

A backward log-sum-exp recursion computes q_i over H departures. Fixed H bounds the per-agent continuation computation independently of map size (Appendix A.4). Gradients of the next-move probabilities can therefore update field scores along these continuations, beyond the agent's current cell. Section 5 compares policies trained with different continuation depths.

3.3 Destination allocation

The route operator supplies next-move probabilities q_i. To account for competing demand for cells, we use affinities $A_i(a) = \tau_f \log q_i(a)$ in an entropy-regularized assignment between agents and their one-step destinations (Cuturi, 2013). Each agent receives unit mass and each cell has unit capacity. Additional rows represent unoccupied capacity; entropy applies to both these rows and the agent assignments, at learned temperature $\tau_p > 0$. Each agent connects only to its valid one-step

destinations, and the identical extra rows are represented collectively without constructing a dense assignment matrix.

We apply Sinkhorn iterations twice, using a fixed iteration budget in each pass. The first uses $A_i(a)$ to estimate a penalty $c_i(a)$ for a likely move by the destination's occupant into agent i's current cell. The second uses $A_i(a) - c_i(a)$ and computes destination potentials λ_x jointly across agents. Row normalisation gives the proposal probabilities

$$\pi_{\theta,i}(a \mid s_t) \propto q_i(a)^{\tau_f/\tau_p} \exp\left[-\frac{c_i(a) + \lambda_{x_i^a}}{\tau_p}\right]. \tag{8}$$

Allocation also raises q_i to the power τ_f/τ_p before normalisation, independently of the reverse-move penalties and destination potentials. The potentials adjust demand for each cell. This identity holds at finite iteration count without requiring converged capacities. We abbreviate $\pi_{\theta,i}(a \mid s_t)$ as $\pi_i(a)$ below.

Training samples movement proposals from π_i; deployment chooses each agent's most probable action. Gradients pass through the route operator and final allocation pass, with the first pass's reverse-move probabilities held fixed. Appendix A.5 gives the allocation formulation and numerical procedure.

3.4 Learning from throughput

We train the policy with AR-EAPO (Choe et al., 2025), an average-reward actor–critic method using a PPO surrogate (Schulman et al., 2017) and a separate entropy advantage. We train with the total goal completion count defined in Section 2 as the task reward, without planner demonstrations or additional task reward shaping.

Average-reward estimation. For training-map entry m, $\widehat{\rho}^r_m$ estimates its reward rate and $V^r_t = V^r_{\chi,m}(O^V_t)$ is the critic's differential value estimate from its scene representation O^V_t. Let $V^{r,+}_{t+1}$ denote its value at the successor before any reset. The reward TD residual is

$$\delta^r_t = r_t - \widehat{\rho}^r_m + V^{r,+}_{t+1} - V^r_t. \tag{9}$$

Generalised advantage estimation (Schulman et al., 2016) accumulates these residuals without temporal discounting. A separate average-reward estimator uses the negative log probability of sampled proposals, averaged over active agents, to estimate an entropy advantage. The reward and entropy advantages are scaled separately and combined with entropy temperature τ. Per-map rate estimates and value-target normalisation accommodate different throughput levels across the training mixture. The critic is discarded at deployment.

Joint policy update. Conditional on the scene, agents sample their proposals independently from the probabilities computed by the shared network, route operator and joint allocation. The log probability of the joint proposal is

$$\log \pi_\theta(\mathbf{a}_t \mid s_t) = \sum_{i=1}^{N} \log \pi_{\theta,i}(a_{i,t} \mid s_t). \tag{10}$$

The PPO update uses the corresponding joint likelihood ratio and one advantage for the full population, with an adaptive KL penalty. Gradients through these proposal probabilities train the differentiable policy.

Training mixtures and continuing rollouts. We train three seeds per fixed mixture, using $H = 4$ for the full policies. The structured mixture contains rack and sortation layouts, with uniform goals and alternating workstation–storage tasks. The unstructured mixture contains random maps, dumbbell layouts and street crops from Denver, Moscow, Boston and Milan; Berlin and Paris are excluded. States persist across updates. Independent random resets introduce fresh placements on the same map and change the sampled state distribution. At a reset, value targets bootstrap from the ordinary successor state before replacement and the advantage trace stops. Appendix B gives the full optimisation settings, reset rates, pool composition and final weight-averaging procedure.

		Training and reuse		Planner involvement	
Method	**Learning**	**Trained on**	**Reused on**	**Training**	**Execution**
Ours	RL	Mixed small tiles	All layouts within a class	None	None
SILLM	IL	Target map^m (downscaled)	Larger target; new populations	Planner demonstrations	PIBT; guide paths for DG
OnlineGGO	ES	Small source map^m	Larger layouts; new populations	Planner rollouts	PIBT; distance updates
Follower	RL	40 generated mazes	All layouts	Planned routes and rewards	A* routes

Table 1: **Training and reuse in the headline learned-policy comparison.** RL, IL and ES denote reinforcement learning, imitation learning and evolution strategies. m denotes map-specific training; SILLM also uses separate policies for each guidance variant. The headline SILLM comparison uses IL checkpoints; its authors report that IL outperforms their MAPPO (Yu et al., 2022) implementation (Jiang et al., 2025). Planner involvement excludes geometric preprocessing.

4 EXPERIMENTS

We evaluate throughput and per-step execution cost on unseen layouts, then examine transfer as agent populations change. We use the structured and unstructured policies from Section 3.4, with the terminal averaged checkpoint from each of three training seeds. Unless stated otherwise, results average cases within each seed and then weight the seed means equally. The six-layout benchmark uses $10{,}000$ agents, occupying 17–26% of free cells. Throughput counts the entire episode; map descriptions, case counts and rollout lengths appear in Appendix C. Instances, task streams and rollout lengths follow SILLM's released evaluation protocol for every method; Appendix E.4 extends the rollouts to $10{,}000$ steps.

Baselines. SILLM (Jiang et al., 2025) combines learned action preferences and PIBT with shortest-distance heuristics (BD), fixed directional edge costs (SG), or TrafficFlow's congestion-aware guide paths (DG). OnlineGGO (Zang et al., 2025) instead learns traffic-dependent edge costs for PIBT. We use released SILLM and Follower (Skrynnik et al., 2024a) weights and train OnlineGGO with its released recipe. Each reproduced learned baseline uses one checkpoint per configuration, so its case averages do not measure training-seed variation. Table 1 summarises training and reuse. Non-learned baselines comprise PIBT, RHCR/WPPL, LLLG (Arita & Okumura, 2026) and TrafficFlow (Chen et al., 2024).

All entries in Table 2 are reproduced measurements. Baseline configurations are specified in Appendix C, and timing protocols in Appendix D.

4.1 THROUGHPUT AND COST ACROSS LAYOUTS

We transfer the small-tile policies to larger layouts outside their training pools, keeping learned parameters fixed and constructing goal-distance tables for each deployment map.

On warehouse and sortation (Table 2), the policy trained on the structured mixture matches SILLM-SG's throughput and stays within 1.3% of the reproduced WPPL scores, with a 1.5–$1.6\times$ execution speedup over SILLM-SG.

On unstructured layouts, the policy takes 6.0–8.9 ms per step. On the 20% obstacle-density random map, throughput exceeds SILLM-DG by 7.6% and TrafficFlow by 20.0%, with approximately $52\times$ and $40\times$ faster execution, respectively. On Berlin and Paris, the same policy also exceeds SILLM-DG's throughput despite receiving no training crops from either city.

Figure 1 also reports throughput and cost at smaller map sizes and populations. Additional large-layout and small-map comparisons, per-seed scores and $10{,}000$-step evaluations appear in Appendix E. Step times exclude map preprocessing; Appendix D gives the measurement protocols and setup costs.

Method	Structured		Unstructured			
	Warehouse	Sortation	Berlin	Paris	Random 10%	Random 20%
Ours	42.31 9.0	44.20 8.3	32.99 8.9	33.17 8.9	54.19 6.0	47.52 7.5
SILLM-SG	42.25 13.4	43.98 13.2	21.62 13.3	22.53 13.2	52.49 13.1	32.53 13.2
SILLM-DG	35.40 292.7	39.39 407.1	30.02 617.3	30.89 602.9	50.50 455.0	44.15 387.8
SILLM-BD	27.27 13.5	42.91 13.2	19.16 13.3	18.68 13.4	53.73 15.3	44.09 13.3
OnlineGGO	21.22 3,427.4	38.13 3,928.2	26.15 3,762.6	27.93 3,586.9	50.15 4,084.5	41.72 3,617.6
Follower	8.52 4,623.1	27.14 4,266.9	8.78 6,204.1	10.08 5,510.8	40.34 4,207.2	26.53 4,356.5
WPPL	42.79 1,432.0	44.25 1,759.4	19.86 1,353.5	22.23 1,715.7	53.45 1,201.5	34.28 1,537.7
TrafficFlow	27.45 208.6	36.98 337.0	27.12 244.1	27.10 269.6	46.94 389.2	39.62 299.1
PIBT	19.07 7.1	32.48 6.9	14.92 7.0	15.39 7.0	46.52 6.9	30.19 7.0
LLLG	17.57 1,361.1	42.33 1,567.5	19.53 2,017.9	20.28 1,941.4	54.80 1,909.3	45.90 1,939.3

Table 2: **Throughput and execution cost at 10,000 agents.** Each cell gives throughput (goals/step) above step time (ms). Random percentages denote obstacle density. WPPL uses 40,000 LNS iterations. Ours reuses one policy per class without retraining; throughput averages three training seeds and eight cases. Ours runs 144 plain allocation iterations on the structured and random layouts and 288 on the city layouts. Timing protocols: Appendix D.

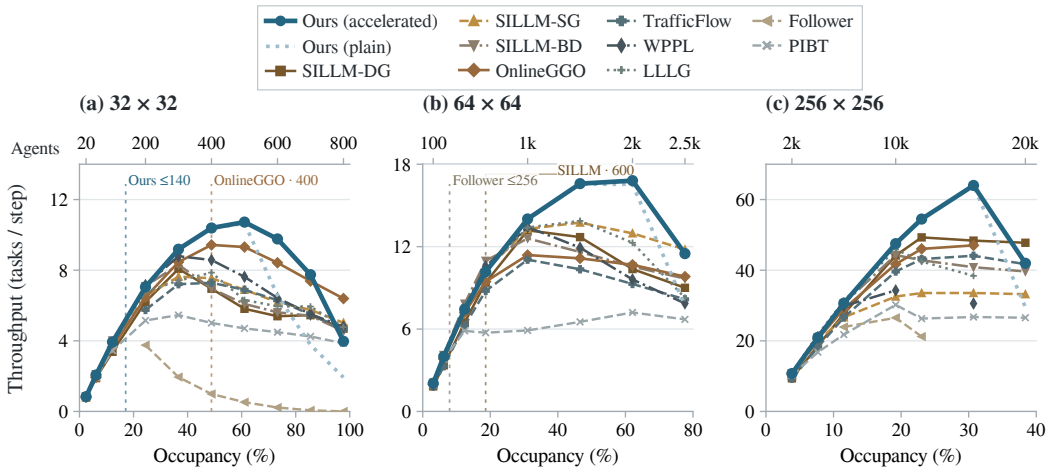


Figure 3: Population transfer with accelerated (solid) and plain (light dotted) Sinkhorn. Both use 288 iterations per pass, except accelerated at 20k agents in (c), which uses 4,608. Ours averages three training seeds and eight cases per seed, with 1,000 steps in (a,b) and 2,500 in (c). Upper axes give agent counts; vertical guides mark training populations (maxima where varied).

4.2 Transfer across populations

We vary population on three fixed random maps of sizes 32^2, 64^2 and 256^2, reusing the unstructured policy with unchanged learned parameters (Figure 3). Occupancy denotes agents per free cell; Appendix C.2 specifies the evaluation. The 16,000–20,000-agent settings extend beyond the 12,000-agent evaluations reported in the original SILLM and TrafficFlow studies (Jiang et al., 2025; Chen et al., 2024).

Although training uses at most 140 agents, mean throughput exceeds the displayed baselines at 600 and 700 agents on 32^2, 1000–2000 on 64^2, and 16,000 on 256^2. On 256^2, throughput declines

at 20,000 agents even with the larger allocation budget. At 20,000 agents SILLM-DG reaches 47.8 tasks per step against our 42.0 with 4608 iterations, at about 2.8× our per-step computation (Appendix D). In extremely dense settings, competition for destination cells can extend across large parts of the map. Appendix E.3 analyses this degradation by comparing the extent of connected competition in training and evaluation.

4.3 Evaluation-time Sinkhorn

When competition for destination cells extends across the map, Sinkhorn may require more iterations to converge. We vary the number and form of these updates at evaluation while keeping the trained parameters fixed. Temperature scaling (Schmitzer, 2019) starts above the learned temperature τ_p and decreases to it; over-relaxation (Thibault et al., 2021) accelerates the potential updates. Both retain the same fixed-point equations at the final temperature.

At 20,000 agents in Figure 3c, 4608 accelerated iterations per pass attain throughput comparable to TrafficFlow. Recorded costs are 323 ms per step for our policy and 616 ms for TrafficFlow on their respective evaluation hosts (Appendix D). On the headline city layouts, acceleration also improves throughput with half as many iterations, reducing per-step time by about 35%. Appendix E.3 reports convergence, throughput and cost measurements. Section 5 complements these evaluations with a retrained control that removes allocation.

5 Mechanism and Interpretation

We examine how route continuation and destination allocation affect learning and throughput, using retrained controls matched to each policy's training pool.

Continuation depth. The next-move probabilities depend on field scores along the route continuations. Policy gradients therefore update scores at successive cells along these routes. We examine this structure by training with $H = 1, 2, 4$, retaining the structured training pool, training budget and destination allocation. Figure 4a shows that longer continuations improve validation throughput during training. The terminal averaged policies retain this ordering on the 10,000-agent warehouse.

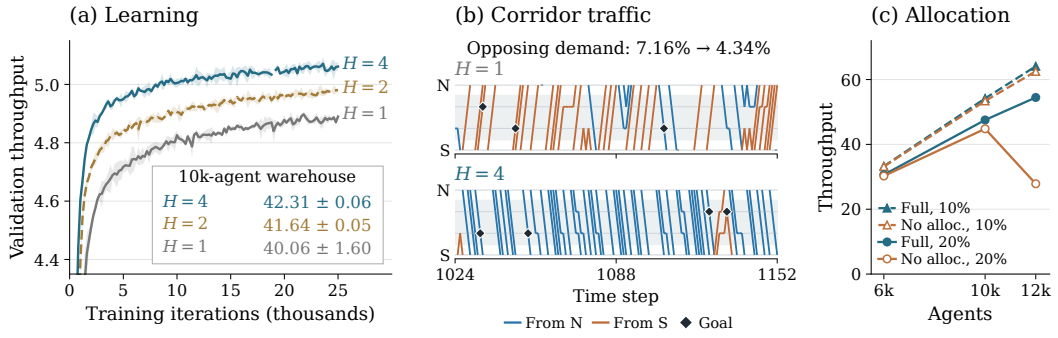


Figure 4: **Continuation and destination allocation.** (a) Validation throughput with current weights; annotations report terminal averaged-policy throughput on the 10,000-agent warehouse. Both show three-run means and standard deviations. The deployment annotations average eight warehouse instances per run. (b) Actual agent trajectories through one passage; colours indicate entry side and diamonds mark goal completions. Opposing-demand percentages aggregate all passages, eight instances and three training runs at 400 agents. (c) Throughput with and without allocation on 10% and 20% obstacle-density random maps, averaged over three training runs and eight instances per run.

To examine the resulting traffic, we record terminal-policy rollouts on a 33×57 warehouse with 400 agents. Across its narrow passages, the mean probability that neighbouring agents request moves towards each other falls from 7.16% with $H = 1$ to 4.34% with $H = 4$, while throughput rises from 8.38 to 9.17. Figure 4b illustrates the traffic through one fixed passage. Preferred travel directions through a passage can change with the traffic state. These measurements describe traffic generated over complete rollouts. Appendix F.3 gives the protocol, a comparison on identical traffic states and additional depth comparisons under load.

Destination allocation. To test the contribution of allocation, we replace both allocation passes and the reverse-move penalty with independent normalisation, then retrain with the same convolutional network, route operator and simulator (Figure 4c). At $12{,}000$ agents, removing allocation reduces throughput from 54.47 to 27.85 on the 20%-obstacle random map, but by only 2.4% on the 10%-obstacle map. Allocation therefore contributes most on the more obstructed layout. Appendix F.4 reports the full population comparison.

6 Related work

Large-scale lifelong MAPF. Planning methods balance coordination over future steps against the computation available for each decision. RHCR solves windowed MAPF problems and reports lifelong experiments with up to 1000 agents (Li et al., 2021). PIBT uses priority inheritance and backtracking for single-step coordination, with runtime experiments reaching $10{,}000$ agents (Okumura et al., 2022). WPPL refines PIBT-initialized plans using large-neighborhood search and evaluates $10{,}000$ agents in lifelong MAPF with rotations (Jiang et al., 2024). Traffic Flow Optimisation supplies congestion-aware guide paths to PIBT, reporting lifelong experiments with $12{,}000$ agents (Chen et al., 2024). LLLG combines local guidance with LaCAM in a receding-horizon planner, evaluated with up to $10{,}000$ agents (Arita & Okumura, 2026).

PRIMAL and PRIMAL2 combine reinforcement and imitation learning, reporting evaluations with up to 1024 agents in one-shot MAPF and 2048 in lifelong MAPF, respectively (Sartoretti et al., 2019; Damani et al., 2021). PRIMAL2 introduces corridor observations and movement conventions. Follower learns to follow planned paths and reports lifelong evaluations with up to 256 agents (Skrynnik et al., 2024a). SILLM combines imitation from windowed plan refinement, spatial communication, global guidance and PIBT-style collision resolution, with population sweeps reaching $12{,}000$ agents (Jiang et al., 2025).

Spatial guidance and shared representations. Spatial guidance and our shared field both associate movement preferences with map locations or directed moves. Direction maps (Jansen & Sturtevant, 2008) and highways (Cohen et al., 2015) encode preferred traffic directions. GGO optimizes guidance graphs, including through convolutional update models, with lifelong evaluations reaching 6000 agents (Zhang et al., 2024). Online GGO adapts their costs to recent traffic and reports $12{,}000$-agent evaluations with guided PIBT (Zang et al., 2025). These costs are used by planners or guide-path calculations. SJRL learns environment guidance and communicating subgoal policies, with Causal PIBT resolving subgoal conflicts; its lifelong experiments with rotations reach 320 agents (Jiang et al., 2026). RAILGUN predicts per-cell action distributions with a map-based convolutional policy trained on planner-generated trajectories, with reported city-tile experiments using up to 256 agents (Tang et al., 2025).

7 Conclusion

We presented an LMAPF policy that reuses neural computation across agents through a shared spatial field. Trained with average-reward reinforcement learning without planner demonstrations, policies learned on small-map mixtures transfer to larger, unseen layouts and populations. Across six layouts with $10{,}000$ agents, they attain throughput comparable to or higher than the highest-throughput evaluated baseline, with execution below 10 ms per step.

Potential extensions include coupling the policy with task assignment and studying shared spatial fields in MAPF with rotations.

AI use statement

We used AI coding assistants to write code to launch experiments and collect data, to search for related work, and to draft and polish text.

REFERENCES

Tomoki Arita and Keisuke Okumura. Lifelong LaCAM with local guidance for lifelong MAPF. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, volume 19, pp. 2–10, 2026. doi: 10.1609/socs.v19i1.43067.

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*, volume 38, pp. 20674–20682, 2024. doi: 10.1609/aaai.v38i18.30054.

Jean Seong Bjorn Choe, Bumkyu Choi, and Jong-kook Kim. Simplifying reward design in complex robotics: Average-reward maximum entropy reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, 2025. doi: 10.1109/ICRA55743.2025.11128043.

Liron Cohen, Tansel Uras, and Sven Koenig. Feasibility study: Using highways for bounded-suboptimal multi-agent path finding. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, volume 6, pp. 2–8, 2015. doi: 10.1609/socs.v6i1.18363.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2292–2300. Curran Associates, Inc., 2013.

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. doi: 10.1109/LRA.2021.3062803.

M. Renee Jansen and Nathan R. Sturtevant. Direction maps for cooperative pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, volume 4, pp. 185–190, 2008. doi: 10.1609/aiide.v4i1.18695.

He Jiang, Yulun Zhang, Rishi Veerapaneni, and Jiaoyang Li. Scaling lifelong multi-agent path finding to more realistic settings: Research challenges and opportunities. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, volume 17, pp. 234–242, 2024. doi: 10.1609/socs.v17i1.31565.

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. doi: 10.1109/ICRA55743.2025.11127445.

He Jiang, Jingtian Yan, Yulun Zhang, Yimin Tang, Tanishq Duhan, Rishi Veerapaneni, Guillaume Sartoretti, and Jiaoyang Li. Search-aided joint agent-environment reinforcement learning for robust lifelong multi-agent path finding with rotations. arXiv preprint arXiv:2608.05588, 2026.

Ilkka Kivimäki, Masashi Shimbo, and Marco Saerens. Developments in the theory of randomized shortest paths with a comparison of graph node distances. *Physica A: Statistical Mechanics and its Applications*, 393:600–616, 2014. doi: 10.1016/j.physa.2013.09.016.

Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11272–11281, 2021. doi: 10.1609/aaai.v35i13.17344.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Hang Ma, Jiaoyang Li, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 837–845, 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*, volume 33, pp. 7643–7650, 2019. doi: 10.1609/aaai.v33i01.33017643.

Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1–3):159–195, 1996. doi: 10.1007/BF00114727.

Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310:103752, 2022. doi: 10.1016/j.artint.2022.103752.

Mike Phillips and Maxim Likhachev. SIPP: Safe interval path planning for dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635, 2011. doi: 10.1109/ICRA.2011.5980306.

Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, T. K. 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. doi: 10.1109/LRA.2019.2903261.

Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 2019. doi: 10.1137/16M1106018. URL https://epubs.siam.org/doi/10.1137/16M1106018.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

David Silver. Cooperative pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, volume 1, pp. 117–122, 2005. doi: 10.1609/aiide.v1i1.18726.

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*, volume 38, pp. 17541–17549, 2024a. doi: 10.1609/aaai.v38i16.29704.

Alexey Skrynnik, Anton Andreychuk, Konstantin Yakovlev, and Aleksandr Panov. Decentralized Monte Carlo tree search for partially observable multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17531–17540, 2024b. doi: 10.1609/aaai.v38i16.29703.

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

Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Barták. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, volume 10, pp. 151–158, 2019. doi: 10.1609/socs.v10i1.18510.

Nathan R. Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012. doi: 10.1109/TCIAIG.2012.2197681. URL https://www.cs.du.edu/~sturtevant/papers/benchmarks.pdf.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018. ISBN 9780262039246.

Yimin Tang, Xiao Xiong, Jingyi Xi, Jiaoyang Li, Erdem Bıyık, and Sven Koenig. RAILGUN: A unified convolutional policy for multi-agent path finding across different environments and tasks. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8645–8651, 2025. doi: 10.1109/IROS60139.2025.11247527.

Alexis Thibault, Lénaïc Chizat, Charles Dossal, and Nicolas Papadakis. Overrelaxed Sinkhorn–Knopp algorithm for regularized optimal transport. *Algorithms*, 14(5):143, 2021. doi: 10.3390/a14050143. URL https://www.mdpi.com/1999-4893/14/5/143.

Emanuel Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24611–24624, 2022. doi: 10.52202/068431-1787.

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*, volume 39, pp. 14726–14735, 2025. doi: 10.1609/aaai.v39i14.33614.

Yulun Zhang, He Jiang, Varun Bhatt, Stefanos Nikolaidis, and Jiaoyang Li. Guidance graph optimization for lifelong multi-agent path finding. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 311–320, 2024. doi: 10.24963/ijcai.2024/35.

A ACTOR SPECIFICATION

A.1 REFERENCE MOVEMENT PROBABILITIES

Soft-distance construction. For a reachable goal g, let $\mathcal{W}_g(x)$ contain finite walks from x whose first visit to g is their final cell. These walks use valid movement actions, may revisit other cells, and contain no WAIT actions; $\mathcal{W}_g(g)$ contains only the empty walk. The ideal soft distance is

$$D_g(x) = -\frac{1}{\beta}\log \sum_{w\in\mathcal{W}_g(x)} e^{-\beta|w|}, \qquad D_g(g) = 0. \tag{11}$$

We use $\beta = 3$. Since the grid degree is at most four and $4e^{-\beta} < 1$, this sum converges. For $x \neq g$, it satisfies $e^{-\beta D_g(x)} = \sum_{a\in\mathcal{M}(x)} e^{-\beta(1+D_g(x_a))}$.

Numerical tables. Write $\widehat{D}_g$ for the numerical table used by the actor. The standard builder performs $2\max(L_y, L_x)+64$ soft-Bellman sweeps in float32 on a map of height L_y and width L_x. The deployment configuration uses a layered fixed-point iteration in $m_g(x) = \exp\{\beta[d(x,g) - D_g(x)]\}$, where d is shortest-path distance. It stops when the largest change relative to the updated $m_g(x)$ is below 10^{-3} over reachable cells, with a budget of 100 passes, and stores distances in uint16 increments of $1/64$. These finite computations approximate Eq. (11). Tables depend only on the static map and goal, are constructed before online action selection, and have no learned parameters.

Normalised action rows and goal phase. At a live route state $z = (x, \varphi)$, the active goal $g_i(z)$ and Eq. (2) define the reference row. We use the shorthand

$$p_i^0(a \mid z) = p^0(a \mid x, g_i(z)). \tag{12}$$

Equation (2) normalises rows even for approximate distance tables and adds WAIT separately from the walks in Eq. (11). Before each row is read, arrival at the current goal promotes the queued goal without consuming an extra departure. Completing both goals makes the finite continuation absorbing, with a probability-one WAIT row and zero local scores thereafter. Cells with no valid moves and inactive padding also use WAIT-only rows. Every WAIT consumes one of the $H = 4$ departures. This boundary applies only to the finite continuation; the environment continues assigning tasks.

A.2 OBSERVATION AND CONVOLUTIONAL NETWORK

The network's raw output is

$$\Phi_t = \big(\widehat{F}_{t,0:3}, G_t\big), \qquad \widehat{F}_{t,h}(x) \in \mathbb{R}^5, \quad G_t(x) \in \mathbb{R}^2. \tag{13}$$

The five-entry rows use $[\mathrm{WAIT}, N, S, W, E]$ coordinates, and G_t is the vector branch output. The shared score b_h and gain inputs σ_h in Eq. (16) are derived from these outputs, not separate learned heads.

Scene image. The convolutional network's 21 input planes consist of occupancy (one plane), the previous realised move of each occupant (two), the navigable mask (one), descent directions for both goals (eight), a relative goal-distance feature (one), and rendered reference routes (eight). Previous moves are zero after a stay. Descent directions and the distance feature are deposited at the agent's current cell. In particular, the distance feature for agent i is

$$\frac{d(x_i, g_i^{\mathrm{next}})}{d(x_i, g_i^{\mathrm{next}}) + d(x_i, g_i) + 1},$$

where d is graph distance. Reference-route rendering draws one 40-action route per agent from the reference probabilities, using a fixed hash to make the sample deterministic. Four planes accumulate directional move counts at departure cells across agents; four more use linearly decreasing weights to emphasise earlier route steps. All eight are divided by one plus the 9×9 local mean of the total unweighted counts, then passed through $\tanh$. This trace length is separate from the route operator's four departures.

Replacing the frozen actor's next-goal input with random goals changes mean throughput by less than 0.3% on all six main layouts, over three training seeds and eight cases per seed. On the Random 30% and Random 40% layouts (Appendix E.5), the changes are $+1.91\%$ and -1.38%, respectively.

Architecture and parameters. The branches have independent encoders and hidden activations. Each base output sums a local 3×3 head and half the summed responses of an affine multiscale head. Two shared kernel bases B_0, B_1 form kernels $B_0 - B_1$, B_0, and $B_0 + B_1$ at dilations 2, 4, and 8; the local head supplies dilation 1. The action-row bases have shape $5 \times 48 \times 3 \times 3$. The vector bases retain only the eight non-central offsets of each 2×8 kernel; the local head supplies the centre.

Component	Architecture	Parameters
Action-row branch	Two width-48 3×3 ReLU convolutions; local and affine heads	36,389
Vector branch	Two width-8 3×3 ReLU convolutions; zero-initialised local and affine heads	2,506
Continuation	Bias-free 15×48 pointwise map	720
Guidance network		**39,615**
Route operator	Four coefficients and τ_f	5
Allocation	τ_p	1
Total actor		**39,621**

Writing $\mathcal{H}_t$ for the action branch's hidden image, the continuation rows are

$$\widehat{F}_{t,0} = \widehat{F}_t, \qquad \widehat{F}_{t,h} = \widehat{F}_t + W_h \mathcal{H}_t, \quad h = 1, 2, 3. \tag{14}$$

The W_h are slices of the pointwise map. The base row and vector output are shared across depths; all outputs are constructed before route branching and refreshed at the next real timestep. The reference tables, recurrence and bound have no learned parameters.

Continuation bound. Restrict $\widehat{F}_{t,h}(x)$ to actions valid at x, giving $F_{t,h}(x)$, and write $\Delta_{t,h} = F_{t,h} - F_{t,0}$ for the corresponding depth correction. Let $\bar{F}$ and $d_F = F - \bar{F}$ denote a row's valid-entry mean and mean-free part, including WAIT. The fixed transformation is applied to the sum of the base row and each continuation correction:

$$\begin{aligned} \mathcal{B}(F) &= \bar{F} + \min\left(1, \frac{\beta_b \tau_f}{\|d_F\|_2}\right) d_F, \qquad \beta_b = 3, \\ F^{\mathrm{cont}}_{t,0} &= F_{t,0}, \qquad F^{\mathrm{cont}}_{t,h} = \mathcal{B}(F_{t,0} + \Delta_{t,h}), \quad h = 1, 2, 3. \end{aligned} \tag{15}$$

The scale is one at zero norm, with numerical norm stabilisation. The immediate row and all row means remain unrestricted. Only the corrected continuation row's mean-free norm is bounded in temperature units by β_b. Subsequent goal-conditioned adjustment and vector addition are unbounded, so this bound does not apply to the full score $U_{i,h}$. The transformation adds no learned parameters.

A.3 Goal-conditioned guidance coefficients

Goal-conditioned directional and axis preferences. At route state $z = (x, \varphi)$, let $p^0_i(\cdot \mid z)$ be the reference row of Eq. (12). Use canonical coordinates $[\mathrm{WAIT}, N, S, W, E]$, with unavailable move entries set to zero after mapping from valid actions. Write $\mathcal{M}(x)$ for the valid movement actions. First remove WAIT and the uniform mass over valid moves from the reference row:

$$r_{\mathrm{WAIT}} = 0, \qquad r_a = \mathbf{1}[a \in \mathcal{M}(x)] \left(p^0_i(a \mid z) - \frac{\sum_{m \in \mathcal{M}(x)} p^0_i(m \mid z)}{|\mathcal{M}(x)|} \right).$$

For a cell without valid moves, set $r = 0$. The opposite-direction preference D_i and the vertical–horizontal preference B_i forming ψ_i in Eq. (3) are the rows

$$\begin{aligned} D_i(z, \cdot) &= \tfrac{1}{2}(0, r_N - r_S, r_S - r_N, r_W - r_E, r_E - r_W), \\ B_i(z, \cdot) &= \tfrac{1}{4}(r_N + r_S - r_W - r_E)(0, 1, 1, -1, -1), \\ \psi_i(z, a) &= \big(D_i(z, a), B_i(z, a)\big)^\top. \end{aligned}$$

These components satisfy $r = D_i + B_i$ and use the same construction for every agent.

Shared scores and field-controlled gains. Suppress timestep t and write $F = F_h^{\mathrm{cont}}(x)$ for the departure row. The shared action score and the two gains used in Eq. (3) are

$$b_h(x,a) = F_a + d_a^\top G(x), \qquad \sigma_h(x) = \begin{pmatrix} F_{\mathrm{WAIT}} \\ \sum_{m\in\mathcal{M}(x)} F_m \end{pmatrix}, \\ \gamma_h(x) = K\sigma_h(x), \qquad K = \begin{pmatrix} \eta_1 - \eta_3 & \eta_2 - \eta_4 \\ \eta_1 + \eta_3 & \eta_2 + \eta_4 \end{pmatrix}. \tag{16}$$

Here d_a is the action displacement, $d_{\mathrm{WAIT}} = 0$, and $\eta_1, \dots, \eta_4$ are the learned coefficients, shared across agents, cells and depths with unrestricted signs. The gains multiply D_i and B_i, respectively. The gain inputs contain F only. The term $d_a^\top G(x)$ contributes to b_h after the goal-conditioned adjustment; including it in the gain inputs would change the route operator at obstacle boundaries.

Obstacle masks. Reference centring uses the mean over valid moves. Form D_i and B_i in canonical, zero-padded coordinates and discard invalid output actions after combining them: either component can be nonzero in blocked directions. The combined term need not have zero mean over valid moves; its WAIT entry is zero.

A.4 Computing the guided route distribution

Soft dynamic programming. The distribution in Eq. (5) is evaluated by a backward recursion. Write $\mathcal{A}(z)$ for the available actions at route state z, and z_a for its successor after action a, including any goal-phase change. Define a soft route value $\mathcal{V}_{i,h}$ in the same score units as $U_{i,h}$:

$$\mathcal{V}_{i,H}(z) = 0, \\ \mathcal{V}_{i,h}(z) = \tau_f \log \sum_{a\in\mathcal{A}(z)} p_i^0(a \mid z) \exp\left(\frac{U_{i,h}(z,a) + \mathcal{V}_{i,h+1}(z_a)}{\tau_f}\right), \qquad h = H-1, \dots, 0. \tag{17}$$

Here $H = 4$, and goal phases follow Appendix A.1. Absorbing and inactive states have zero local scores and soft value. The first-action marginal is

$$q_i(a) = p_i^0(a \mid z_0) \exp\left(\frac{U_{i,0}(z_0,a) + \mathcal{V}_{i,1}(z_a) - \mathcal{V}_{i,0}(z_0)}{\tau_f}\right). \tag{18}$$

The implementation uses $\log\max(p_i^0, 10^{-30})$ on positive numerical reference probabilities, with zero-probability and invalid actions masked separately. The finite route law uses the supplied numerical reference weights, subject to this floor and floating-point evaluation. The root affinity equals $\tau_f \log q_i(a)$ up to a per-agent constant removed by allocation row centring.

Score derivative. Differentiating the finite route marginal with respect to one local score gives

$$\frac{\partial \log q_i(a_0)}{\partial U_{i,h}(z,b)} = \frac{1}{\tau_f}\left[\Pr_{Q_i}(z_h = z, a_h = b \mid a_0) - \Pr_{Q_i}(z_h = z, a_h = b)\right]. \tag{19}$$

Field gradients chain through the route operator and continuation transform and accumulate across agents, phases and depths. Equation (19) applies before allocation; proposal gradients also pass through its final allocation pass.

A.5 Destination allocation and action selection

For allocation, express each route-operator output in score units as

$$A_i(a) = \tau_f \log q_i(a). \tag{20}$$

Allocation constraints and slack. Let $M = |V|$ be the number of cells, N the number of agents, and $D = M - N$ the number of dummy rows representing unused cell capacity, termed slack rows. Write $\mathcal{C}_i = \{x_i^a : a \in \mathcal{A}(z_i)\}$, with root state $z_i = (x_i, \varphi_i)$, for agent i's candidate destinations, including its current cell. The augmented allocation in Eq. (22) has one row per agent or slack row and one column per cell, with feasible set

$$\mathcal{U} = \left\{T \in \mathbb{R}_+^{M\times M} : T\mathbf{1} = \mathbf{1}, \quad T^\top\mathbf{1} = \mathbf{1}, \quad T_{ix} = 0 \text{ if } i \le N,\ x \notin \mathcal{C}_i\right\}. \tag{21}$$

Thus every real agent receives one unit of mass and every cell has unit capacity; slack fills the unused capacity. With fixed affinity matrix B, including a common score for slack rows, the ideal assignment is

$$T^*(B) = \arg\max_{T \in \mathcal{U}} \big\{ \langle T, B \rangle + \tau_p \mathcal{H}(T) \big\}, \qquad \mathcal{H}(T) = -\sum_{j,x} T_{jx} \log T_{jx}. \tag{22}$$

We apply a fixed number of Sinkhorn iterations to this balanced entropic transport formulation on the candidate support (Cuturi, 2013).

Entropy applies to both agent and slack rows. When $D = 0$, there are no slack rows and every cell receives one unit of real-agent mass.

Finite scaling and reverse-edge repricing. The allocation procedure subtracts each valid affinity row's arithmetic mean, $\bar{A}_i(a) = A_i(a) - |\mathcal{A}(z_i)|^{-1} \sum_b A_i(b)$. Row constants leave the finite allocation procedure unchanged in exact arithmetic. Each pass starts from zero destination potentials. For scores B_{ix}, one sweep computes

$$f_i = -\tau_p \log \sum_{x \in \mathcal{C}_i} e^{(B_{ix} + g_x)/\tau_p}, \qquad f_d = -\tau_p \log \sum_{x \in V} e^{(b_d + g_x)/\tau_p},$$
$$g_x = -\tau_p \log \left[\sum_{i : x \in \mathcal{C}_i} e^{(B_{ix} + f_i)/\tau_p} + D e^{(b_d + f_d)/\tau_p} \right]. \tag{23}$$

The sums use log space and sparse candidate gathers and scatters. The common slack affinity b_d cancels in $b_d + f_d$; slack rows are represented by the scalar f_d and multiplicity D, without materialising the augmented matrix.

The first pass uses $B_{i,x_i^a} = \bar{A}_i(a)$. Its candidate logits are normalised within each valid row to give $p_i^{(1)}$. The second pass uses $B_{i,x_i^a} = \bar{A}_i(a) - c_i(a)$, where

$$c_i(a) = \begin{cases} 2p_j^{(1)}(b), & a \neq \text{WAIT},\ x_i^a = x_j,\ x_j^b = x_i, \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

Here j is the current occupant of the candidate destination and b is its reverse move. WAIT and vacant destinations incur no surcharge. Each pass uses 48 sweeps during training; headline evaluations use 144 on structured and random layouts and 288 on Berlin and Paris (Appendix C). The objective in Eq. (22) describes each pass with its scores held fixed.

Returned probabilities and gradients. From the final pass, the allocation procedure returns valid candidate logits

$$\ell_i(a) = \frac{\bar{A}_i(a) - c_i(a) + f_i + g_{x_i^a}}{\tau_p}, \qquad \pi_i(a) = \mathrm{softmax}_a\, \ell_i(a). \tag{25}$$

Row normalisation removes f_i and the subtracted affinity mean. Using Eq. (20) and setting $\lambda_x = -g_x$ therefore gives Eq. (8) at any finite sweep count, up to numerical error.

Each sweep ends with column normalisation of the augmented transport, while finite row-sum errors can remain. The final row normalisation can therefore violate the cell capacities.

The temperatures are $\tau_f = \exp(\omega_f)$ and $\tau_p = \exp(\omega_p)$, both initialised at 0.15 and exempt from weight decay. The backward pass differentiates the final allocation pass, including its dependence on the affinity rows and τ_p. It holds the first pass's probabilities fixed when forming the reverse-edge surcharge. Training uses the log probability of the sampled proposals.

A.6 Construction of the mechanism illustration

Figure 2 uses the unstructured seed-2 terminal checkpoint with averaged weights at iteration 24999 (nominal 25k). Selected route prefixes span four departures, using the current guidance throughout. Ribbon widths indicate the probability of traversing each edge at its departure depth; a minimum

width preserves rare alternatives. The guidance effect is the reweighted probability minus the prior probability at the same edge and departure. Numeric route labels give first-move probabilities. The allocation views compare the operator output q_i with the complete output π_i of Eq. (8), including temperature scaling within allocation. B participates in the calculation; its candidate destinations are disjoint from A's and C's and are omitted from the connection diagram.

B TRAINING DETAILS

Shared settings. Table 3 gives the shared optimisation settings. Each seed uses $25{,}000$ updates of 128×64 joint environment steps, totalling $204{,}800{,}000$ steps. Training takes approximately 12–13 hours per seed on one RTX 5090, including periodic evaluation. Advantages use generalised advantage estimation (Schulman et al., 2016); the actor uses the clipped PPO surrogate (Schulman et al., 2017) with an adaptive KL penalty and Adam with decoupled weight decay (Loshchilov & Hutter, 2019). The scene likelihood ratio is the product of agents' proposal likelihood ratios. The actor and critic have separate parameters; the critic is discarded at deployment.

Component	Setting
learning setup	average reward, $\gamma = 1$, per-map rate; entropy temperature $\tau = 0.05$
advantage estimation	GAE: reward $\lambda = 0.9$, entropy $\lambda = 0.5$
actor update	PPO clip 0.296; 2 passes; 8 minibatches
KL penalty	adaptive coefficient, initial value 1; target 0.015
critic update	4 passes, decoupled; per-map target normalisation; no weight decay
optimiser	Adam, constant learning rate 2×10^{-4}; actor weight decay 0.03, learned temperatures exempt
batch	$B = 128$ environments $\times$ $T = 64$ steps per iteration
reported checkpoint	terminal EMA weights; decay 0.9998, initialised from the initial actor and updated once per iteration
allocation	48 Sinkhorn iterations per pass during training

Table 3: Shared training settings for the two selected policies.

Reward and entropy advantages. AR-EAPO (Choe et al., 2025) treats reward and entropy as separate learning signals, each centred by its estimated long-term average. For map entry m, the two signals are $x_t^r = r_t$ and $x_t^e = -N^{-1} \sum_i \log \pi_{\mathrm{old},i}(a_{i,t} \mid s_t)$, where N counts active agents. Write $V_t^j = V_{\chi,m}^j(O_t^V)$ for critic output $j \in \{r, e\}$ and $V_{t+1}^{j,+}$ for its value at the successor before any random reset. The differential TD residual and GAE recursion are

$$\begin{aligned} \delta_t^j &= x_t^j - \widehat{\rho}_m^j + V_{t+1}^{j,+} - V_t^j, \\ \mathcal{A}_t^j &= \delta_t^j + \lambda_j c_t \mathcal{A}_{t+1}^j, \end{aligned} \tag{26}$$

where $c_t = 0$ at a reset and 1 otherwise. At rollout boundaries, the advantage trace ends with zero and the critic supplies the successor value. After each rollout, the zero-initialised rates are updated by $\widehat{\rho}_m^j \leftarrow \widehat{\rho}_m^j + \eta_j(1 - \lambda_j)\mathbb{E}_t[\mathcal{A}_t^j]$, with $(\eta_r, \eta_e) = (0.23, 0.116)$ and $\mathbb{E}_t$ the empirical average of unnormalised rollout advantages. The actor divides each advantage by its map's temporal standard deviation plus 10^{-6}, combines them as $\widetilde{\mathcal{A}}_t^r + \tau \widetilde{\mathcal{A}}_t^e$ with entropy temperature τ, and subtracts the minibatch mean.

Critic and value-target normalisation. The critic augments the actor's 21 channels with four planes at agents' current cells. These encode normalised time since the last goal completion and three indicators of whether the current goal is within 1, 2 or 4 moves. Two 3×3 convolutions with 128 channels and tanh activations, global spatial averaging and a shared two-output linear head produce $g_\chi^j(O_t^V)$. Per-map statistics give $V_{\chi,m}^j = \sigma_m^j g_\chi^j + \mu_m^j$ for target construction. Targets $\mathcal{A}_t^j + V_{\chi,m}^j(O_t^V)$ update running first and second moments with weight 0.2, then are normalised using the updated moments and clipped to $[-10, 10]$. Each standard deviation is floored at the larger of 0.01 and 0.1 times the cross-map RMS standard deviation. This follows the value-target normalisation of PopArt (van Hasselt et al., 2016), but the shared head is not rescaled to preserve outputs when the statistics change.

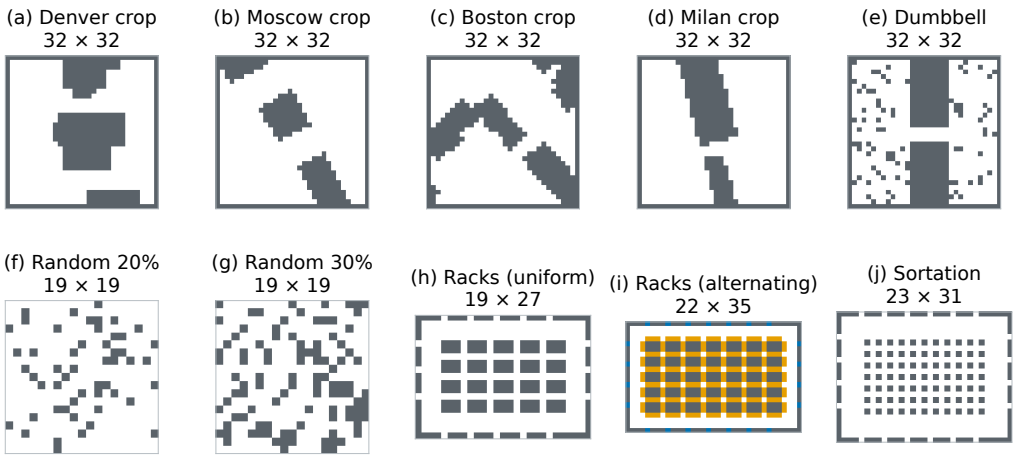


Figure 5: Examples from the fixed training pools. Gray cells are obstacles; the dumbbell layout connects two regions through a narrow passage. In (i), blue and orange mark workstations and storage endpoints. Dimensions are in cells.

KL control. The penalty uses the old-to-new categorical KL averaged over padded agent rows; inactive rows are deterministic waits and contribute zero. Its coefficient is multiplied by 1.5 above 1.5 times the target and divided by 1.5 below the target divided by 1.5, with bounds $[10^{-3}, 10^{3}]$. The PPO surrogate uses the joint scene likelihood ratio, not per-agent ratios.

Fixed training pools. Figure 5 illustrates the training layouts. Each policy uses three independent training seeds and one fixed pool of 128 entries, shared across its seeds; maps are padded and masked to a common tensor size within the pool. The structured policy (kiva160) uses 48 rack layouts with uniform goals and 128 agents, 48 rack layouts with alternating workstation–storage goals and 160 agents, and 32 sortation layouts with uniform goals and 128 agents. Rack layouts use a three-cell outer ring. The unstructured policy (Denser) uses 48 32×32 street crops and 48 32×32 dumbbell layouts, each with 140 agents, plus 16 19×19 random maps at each of 20% and 30% obstacle probability, with 64 agents. Dumbbell layouts are rotated and reflected across the pool. Pool generation uses seeds 4300 and $10{,}000$ for the structured and unstructured policies, respectively; training and initialisation use seeds 0, 1 and 2.

Street-crop windows are sampled from Denver, Moscow, Boston and Milan in the Moving AI benchmarks (Sturtevant, 2012). An automatic geometric criterion retains a dominant connected free region and favours passages linking substantial free regions, detected by erosion of free space. The sampler accepts a qualifying window or uses the best-scoring valid candidate.

Goal sampling and resets. Uniform training goals are drawn from reachable cells at graph distance at least 15 on structured maps and 10 on unstructured maps. On the alternating-goal rack entries, a workstation source draws uniformly from storage endpoints; any other source draws uniformly from workstations. Workstations occupy perimeter gaps, and endpoints are free cells adjacent to interior racks. The distance floor of 15 applies within the required category; if no candidate satisfies it, the floor relaxes to 1 within that category, preserving alternation. Deployment consumes the released benchmark task streams. Each environment resets independently at every step with probability 5×10^{-4} on structured maps or 2.5×10^{-4} on unstructured maps, giving expected intervals of 2000 and 4000 steps, respectively. Resets place fresh agents on the same map; states otherwise persist across updates.

C Evaluation protocol

C.1 Layout and small-map comparisons

Policies and reporting. The structured and unstructured policies are kiva160 and Denser, respectively, trained as specified in Appendix B. Each seed's terminal averaged parameters are reused

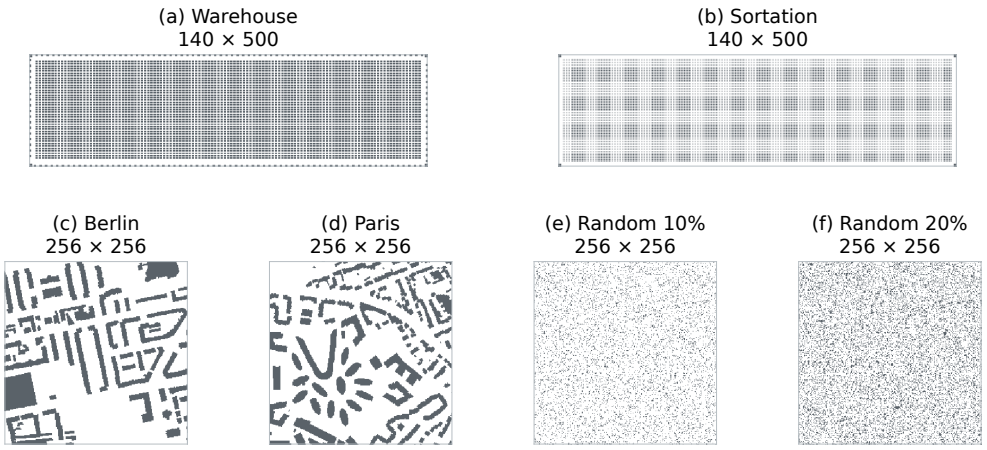


Figure 6: The six deployment layouts evaluated with $10{,}000$ agents. Gray cells are obstacles; dimensions are in cells.

across every layout of its class. Throughput counts all completed goals over the episode; we average cases within each seed, then average seed means equally.

Berlin and Paris are excluded from the training crops.

Baselines. The headline SILLM comparison uses released IL checkpoints trained on downscaled target layouts, with BD, SG and DG guidance as specified in Section 4. Follower uses its released checkpoint across layouts. The Follower scores reported by Jiang et al. (2025, Table III) use target-retrained models and are not paired with our released-checkpoint timings. The headline OnlineGGO on+PIBT comparison transfers policies trained on smaller source maps to the corresponding large layouts. TrafficFlow uses GP-R100-Re10-F2 without action-LNS. The headline WPPL comparison uses PIBT-LNS with $40{,}000$ iterations, eight threads and a 15-step planning window.

Maps and cases. Figure 6 shows the six large layouts from SILLM (Jiang et al., 2025). We use its released start/task cases; Table 4 gives the evaluation settings. Random percentages specify obstacle density; occupancy is agents per free cell.

Class	Layout	Size	Free cells	Occupancy	Steps
Structured	Warehouse	140×500	38,586	25.9%	3200
Structured	Sortation	140×500	54,320	18.4%	3200
Unstructured	Berlin	256×256	46,880	21.3%	2500
Unstructured	Paris	256×256	47,096	21.2%	2500
Unstructured	Random, 10%	256×256	58,859	17.0%	2500
Unstructured	Random, 20%	256×256	52,090	19.2%	2500

Table 4: Deployment maps at $10{,}000$ agents, with eight released cases (0–7) per layout. Size is height $\times$ width; occupancy is agents per free cell. Every call uses one actor and one instance.

The small random benchmark uses 128 POGEMA (Skrynnik et al., 2025) maps, 17–21 cells per axis, populations 8, 16, 24, 32, 48 and 64, and 256-step episodes. Baselines use the corresponding cases (Li et al., 2021; Arita & Okumura, 2026; Skrynnik et al., 2024b); LLLG uses its recorded model-A configuration. The dense 64^2/600-agent comparison uses eight released start/task cases and 256 steps. These populations support the small and intermediate panels of Figure 1; the separate population-transfer study is specified below. Panel (a) of Figure 1 uses a connected twin of the 128 maps: each map is regenerated with the POGEMA generator at the same size and obstacle fraction (within 0.01) until its free cells form one 4-connected component, because SILLM's map loader keeps only the largest component and 43 of the original maps place agents outside it. Every method in panel (a) is measured on that set with the same instances. PIBT denotes SILLM's released implementation throughout, so that PIBT and the SILLM variants share one simulator.

Movement rules. Training and evaluation use POGEMA's soft collision rules (Skrynnik et al., 2025). Attempted swaps are cancelled. At a contested cell, a stationary occupant takes precedence

over incoming agents; otherwise, the lowest-index mover takes precedence. Rejected moves become WAIT, and the resulting cancellations propagate until no conflicts remain. A cancelled move remains WAIT for the rest of the timestep.

Execution. The headline policies use argmax proposals and layered 16-bit reference tables. Table 2 uses 144 plain allocation iterations on structured and random layouts and 288 on Berlin and Paris. Each rollout evaluates one actor on one instance. Every task stream is long enough to supply fresh goals throughout the episode.

Map preprocessing and execution-cost measurements are reported in Appendix D.

C.2 Population-transfer study

Policy reuse. Figure 3 uses the same unstructured policy as the headline comparison, with its training pool specified in Appendix B. Each seed reuses its terminal averaged parameters across every map and population, with argmax proposals and 288 accelerated allocation iterations, increased to 4608 at $20{,}000$ agents on the 256^2 map. A plain 288-iteration reference uses the same checkpoints and cases. The accelerated updates are specified in Appendix E.3. Geometric route tables are constructed for each deployment map.

Evaluation cases. The 32^2, 64^2 and 256^2 maps contain 819, 3221 and $52{,}090$ free cells, respectively. Episodes last 1000 steps on the two smaller maps and 2500 steps on the largest; throughput counts the whole episode. Our policy uses eight start/task cases at every population. The 32^2 cases use uniform goals on OnlineGGO's random map. For the two larger maps, additional populations use SILLM's released agent-placement generator and task streams. Our points average three training seeds and eight cases per seed.

Baseline training and execution. OnlineGGO is trained on the 32^2 evaluation map with 400 agents using the released recipe; its guidance is updated every 20 steps, with the initial 20-step unit-guidance warm-up included in the episode. The SILLM policies are trained on the 64^2 evaluation map with 600 agents. Follower uses the released policy trained on 40 separate 65^2 mazes with 128 or 256 agents. The vertical guides therefore mark training agent counts, not the occupancy of the source maps. WPPL uses 2500, $10{,}000$ and $40{,}000$ LNS iterations on the 32^2, 64^2 and 256^2 maps, respectively (Appendix D); TrafficFlow uses GP-R100-Re10-F2 without additional action-LNS. The PIBT curve uses SILLM's released implementation. Unmeasured populations are left as gaps in the curves.

D Execution-cost protocol and measurements

Measurement definitions. Our execution cost is synchronised wall time for the complete policy and simulator, excluding setup and warm-up. Table 5 summarises the timing intervals; throughput always uses complete episodes.

Measurement	Scope
Full rollouts	Our policy, SILLM-SG, SILLM-BD, city and dense-random SILLM-DG, and dense-random WPPL; full-rollout medians for Follower and warehouse OnlineGGO.
Simulation loops	PIBT and TrafficFlow on large maps, excluding initialisation.
Rollout segments	Steps 20–300 for other SILLM-DG, WPPL and OnlineGGO layouts, and LLLG; recorded driver intervals for TrafficFlow. The 600-agent comparison uses 256-step process timings, including analysis overhead.
Small-map planners	Planner time for LLLG and action-call time for RHCR on the 128-map benchmark.

Table 5: Execution-cost measurements. Our rollouts, warehouse SILLM-SG, dense-random SILLM-DG and 600-agent segments use case 0; other SILLM full rollouts and large-map PIBT timings average eight cases.

Hardware and baseline settings. Dedicated timings use an AMD Ryzen 7 7800X3D and RTX 5090. OnlineGGO timings on sortation, Berlin, Paris and the two random layouts use a Ryzen

9 7950X3D restricted to its V-Cache CCD. Follower and warehouse OnlineGGO use timings from their evaluation hosts, with RTX 5090 GPUs for Follower. Headline timings use 16 OpenMP threads for SILLM-SG/DG, eight for the refreshed SILLM-BD runs, six for PIBT, one for TrafficFlow and eight LNS threads for WPPL. The headline WPPL comparison uses no-rotation PIBT-LNS, $40{,}000$ refinement iterations and a 15-step planning window; its Random 20% 10k cost covers a complete rollout. LLLG's 10k segments use the dedicated CPU, planning window 8 and horizon 5. RHCR at 600 agents uses PBS/SIPP, windows 5/10, a 10-second planning-call limit and its Python executor. The 20k-agent TrafficFlow cost in Section 4.3 averages eight 2500-step runs on an AMD EPYC 7702 with four OpenMP threads. It includes process setup; recorded planning time averages 599 ms per step. Our corresponding 323 ms cost uses seed 0, case 0 on the dedicated hardware and excludes setup. The 20k-agent SILLM-DG cost in Section 4.2 is one case-0 2500-step rollout of 924 ms per step on a second host with an AMD EPYC 7B12 CPU and the same RTX 5090, where our 4608-iteration rollout costs 325 ms per step (323 on the dedicated hardware).

WPPL refinement budget. Table 6 varies the LNS iteration budget on Random 10% with $10{,}000$ agents. Increasing the budget from 2500 to $40{,}000$ improves throughput by 3.0% while increasing execution cost by $4.8\times$. On the $10{,}000$-agent layouts, 2500 iterations give 41.97 on the warehouse, 43.75 on sortation (seven of eight cases) and 28.34 on Random 20%, against 42.79, 44.25 and 34.28 at $40{,}000$ iterations (Table 2), at 255 and 348 ms per step on the warehouse and Random 20% (step-segment costs on the dedicated hardware). The population study (Figure 3) scales the budget with map size: 2500 iterations on the 32^2 map, $10{,}000$ on 64^2 and $40{,}000$ on 256^2, the setting SILLM's harness applies to the $10{,}000$-agent layouts. At $40{,}000$ iterations on Random 20%, complete case-0 rollouts cost 1033, 1538 and 1603 ms per step at 6000, $10{,}000$ and $12{,}000$ agents. At the smaller budgets, case-0 rollouts cost 61–100 ms per step on the 32^2 map (200–800 agents, 2500 iterations) and 146–377 ms per step on the 64^2 map (100–2500 agents, $10{,}000$ iterations), on the same host.

LNS iterations	Throughput (tasks/step)	Execution cost (ms/step)
2500	51.867 ± 0.089	252.0
10,000	52.725 ± 0.110	474.9
40,000	53.448 ± 0.107	1201.5

Table 6: WPPL refinement-budget comparison. Throughput is mean $\pm$ standard deviation over eight 2500-step cases. Costs use case 0, steps 20–300, on the dedicated hardware above, following Table 5.

Map preparation. Reference tables are constructed once per map and reused across tasks (Table 7).

Map	Construction (s)	Arrays (GiB)
Warehouse	16.5	5.53
Sortation	65.8	10.86
Berlin	24.7	8.13
Paris	24.5	8.21
Random, 10%	91.1	12.70
Random, 20%	63.1	10.00

Table 7: Reference-table construction times and sizes on RTX 5090. Times cover the host construction call; compilation and asynchronous completion are included in warm-up.

Deployment memory. Table 8 reports peak memory at $10{,}000$ agents on the Random 20% and warehouse layouts (case 0, the Table 2 horizons). For our policy it lists the sampled device peak and the host resident-set peak of one fresh process at 144 iterations, the latter including map preparation and compilation; for the baselines, the peak resident set of the process tree and, where a GPU is used, the process's device memory. Entries marked † cover a 1000-step prefix.

	Random 20%		Warehouse	
Method	Host (GiB)	GPU (GiB)	Host (GiB)	GPU (GiB)
Ours	12.2	16.6	7.5	9.0
SILLM-SG	19.3	11.5	11.6	6.9
SILLM-DG	23.4	1.3	22.2	1.3
SILLM-BD	20.2	11.5	9.8	6.9
PIBT	10.6	—	6.1	—
WPPL†	11.1	—	6.5	—
TrafficFlow	23.3	—	22.5	—
LLLG†	8.2	—	6.2	—
Online GGO†	40.3	—	44.9	—
Follower†	31.1	0.9	33.4	0.9

Table 8: Peak deployment memory at 10,000 agents. Host values are resident-set peaks; GPU values are the process's device memory (our sampled device peak includes the allocator's reservation). †: 1000-step prefix.

E Additional evaluation results

E.1 Deployment results by training seed

Table 9 gives the seed-level plain-allocation results behind Table 2, under the protocol in Appendix C.

Layout	Seed 0	Seed 1	Seed 2
Warehouse	42.272	42.387	42.279
Sortation	44.191	44.228	44.180
Berlin	33.518	32.033	33.412
Paris	33.618	31.097	34.809
Random 10%	54.161	54.217	54.180
Random 20%	47.483	47.564	47.514

Table 9: Terminal-checkpoint throughput by training seed; each entry averages eight cases.

E.2 Small and intermediate operating scales

Table 10 reports six populations on the 128-map random benchmark, using the unstructured policy with 144 plain allocation iterations and 256-step episodes, with RHCR as the reference across populations.

Agents	Full s0	Full s1	Full s2	RHCR
8	0.607	0.610	0.607	0.625
16	1.208	1.211	1.207	1.251
24	1.725	1.728	1.722	1.798
32	2.175	2.174	2.164	2.286
48	3.041	3.042	3.036	3.203
64	3.795	3.795	3.785	3.819

Table 10: Throughput on 128 small random maps at every tested population. Policy columns distinguish training seeds; each covers the same benchmark cases as RHCR.

Table 11 compares the same eight dense 64^2/600-agent cases over 256 steps. Ours is 3.3% below WPPL, 2.3% below LLLG and 11.5% above SILLM-DG. RHCR uses PBS (Ma et al., 2019) with SIPP (Phillips & Likhachev, 2011), simulation/planning windows 5/10 and a 10-second planning-call limit, reached by 46–49 calls per case.

Method	Throughput
Ours (three seeds)	9.778
WPPL	10.108
LLLG (model A)	10.007
SILLM-DG	8.766
PIBT	7.821
RHCR (PBS/SIPP)	0.892

Table 11: Reproduced dense 64^2/600-agent comparison, eight cases and 256 steps. Ours uses plain allocation and averages three training seeds.

E.3 Evaluation-time Sinkhorn

Accelerated updates. We evaluate the same terminal checkpoints with temperature scaling (Schmitzer, 2019) and over-relaxation (Thibault et al., 2021). For an allocation pass with K iterations, iteration $k \in \{0, \dots, K-1\}$ uses

$$\tau_{p,k} = \tau_p \, 8^{\max\{1-k/\lfloor K/2 \rfloor, 0\}}, \qquad \xi \leftarrow (1-\omega)\xi + \omega\widetilde{\xi}, \quad \omega = 1.6. \tag{27}$$

Here $\widetilde{\xi}$ is the corresponding update in Eq. (23), evaluated at $\tau_{p,k}$, for $\xi = f_i, f_d, g_x$. Row and slack potentials are updated before computing the destination update. Both allocation passes start from zero potentials and use this schedule; the reverse-edge repricing rule is unchanged. At the final temperature τ_p, the relaxed updates have the same fixed-point equations as plain scaling. Learned parameters remain fixed, and all checkpoints were trained with 48 plain Sinkhorn iterations per pass.

Deployment layouts. Table 12 compares plain allocation with 144 accelerated iterations on the six headline layouts. Throughput averages eight cases within each of three training seeds. Execution cost uses seed 0, case 0 and the synchronised rollout protocol in Appendix D. The accelerated timing implementation precomputes the temperature schedule and omits per-iteration diagnostics.

	Plain			Accelerated, 144 iterations	
Layout	Iterations	Throughput	ms/step	Throughput	ms/step
Warehouse	144	42.31	9.02	42.30	9.24
Sortation	144	44.20	8.31	44.19	8.52
Berlin	288	32.99	8.89	33.47	5.77
Paris	288	33.17	8.91	33.79	5.87
Random 10%	144	54.19	5.97	54.19	6.19
Random 20%	144	47.52	7.46	47.53	7.91

Table 12: Evaluation-time allocation at $10{,}000$ agents. Throughput is in tasks per step and execution time in ms per step. Plain values are those of Table 2; accelerated values use 144 iterations on every layout.

At 96 accelerated iterations, warehouse and sortation attain 42.31 and 44.20 tasks per step in 6.81 and 6.37 ms. Their throughput, and that on the two random layouts, differs from the headline result by less than 0.004 tasks per step. At 288 accelerated iterations, Berlin and Paris reach 33.86 and 34.24 tasks per step.

Population and iteration count. Figure 7 extends the population study to the warehouse and varies the accelerated iteration count. The $20{,}000$-agent point on 256^2 uses 4608 iterations. Our points average three training seeds and eight cases per seed. We use 288 iterations on random maps and 144 on the warehouse as baseline counts. For the conditions in Table 14, additional iterations are chosen using the allocation-error criterion below. Table 13 reports accelerated throughput and measured costs, including intermediate 2304-iteration and additional seed-0 9216-iteration results at $20{,}000$ agents.

On the 33×57 warehouse with 700 agents, 144 accelerated iterations attain 9.42 tasks per step, compared with 2.63 at 144 plain iterations and 9.51 at 4032 plain iterations. At 600 agents on 32^2 and 2500 on 64^2, acceleration at 288 iterations raises throughput from 6.58 to 9.77 and from 7.71 to 11.49. These comparisons show that changing the allocation updates can improve throughput under congestion without retraining.

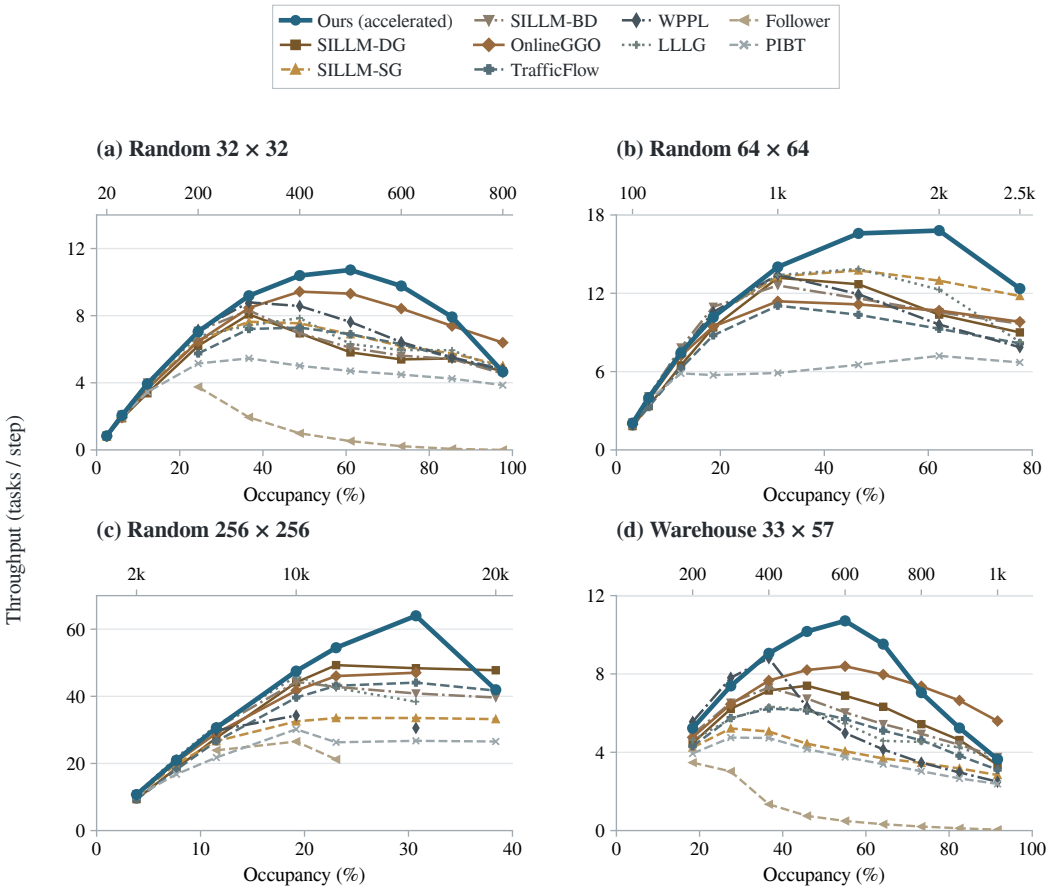


Figure 7: Population transfer with accelerated Sinkhorn. Iteration counts use the probability-stability calibration described in the text, with fixed baseline counts at lower populations; the 20k-agent point in (c) uses 4,608 iterations. Ours averages three training seeds and eight cases per seed, with 1,000 steps in (a,b,d) and 2,500 in (c). Upper axes give agent counts.

Map	Agents	Iterations	Ours	OnlineGGO	ms/step
Random 32×32	700	288	7.75	7.39	6.12
		576	7.92		—
Random 32×32	800	288	3.97	6.39	6.79
		2,304	4.66		—
Random 64×64	2,500	288	11.49	9.82	8.38
		2,304	12.36		—
Random 256×256	20,000	288	32.80	—	24.66
		2,304	40.05		163.81
		4,608	41.95		323.29
		9,216	43.80†		—
Warehouse	700	144	9.42	7.97	4.48
		288	9.52		8.01
Warehouse	800	144	4.62	7.37	5.27
		4,608	7.04		—
Warehouse	900	144	2.21	6.64	5.34
		4,608	5.24		—
Warehouse	1,000	144	1.01	5.59	5.68
		2,304	3.64		—

Table 13: Accelerated allocation at the baseline and calibrated iteration counts, with intermediate and extended-budget results at 20k agents. Our throughput averages three training seeds and eight cases per seed; † marks seed-0 results at 9,216 iterations. Costs use seed 0, case 0 at the listed map, population and budget; dashes indicate unavailable measurements.

Competition across scales. We connect agents through destination cells to which they assign probability greater than 0.01 and whose total assigned probability exceeds 0.95. Table 14 reports the largest connected group's size and approximate graph diameter, measured in agent–cell–agent links. Training measurements use the saved states and live actors at 3000, 10,000 and 24,999 updates, with 128 streams per checkpoint and policy. Evaluation measurements use 25 evenly spaced states from each of ten 1000-step rollouts, with training seed 0 and case 0. The largest group averages about four agents during training, compared with 3641 at 20,000 agents on 256^2. On the 900-agent warehouse, its mean spatial span is 49 by 26 cells on the 33×57 map. Increasing population can therefore connect local competition over distances substantially larger than those observed during training.

Setting	Group size	Diameter	$K^\star$
Unstructured training	4.34	2.25	—
Structured training	4.03	2.51	—
Warehouse, 600	13.84	6.36	144
Warehouse, 700	49.68	17.56	288
Warehouse, 800	314.68	53.52	4608
Warehouse, 900	456.36	60.68	4608
Warehouse, 1000	675.80	68.68	2304
Random 32^2, 700	96.52	24.00	576
Random 32^2, 800	521.28	56.96	2304
Random 64^2, 2500	816.60	60.36	2304
Random 256^2, 16,000	32.76	12.72	288
Random 256^2, 20,000	3641.16	91.00	4608

Table 14: Connected competition in historical training and evaluation states. Group sizes and graph diameters are means over sampled states, using an 18,432-iteration allocation reference. $K^\star$ is the smallest eligible count selected by the empirical allocation-error criterion; training rows report no evaluation count. Diameter is estimated by two successive breadth-first searches.

Iteration criterion. We choose iteration counts with an empirical numerical check, comparing allocation probabilities with those from a much longer run on the same saved states. Holding each state's inputs fixed, let $p_{i,K}(\cdot \mid s)$ be agent i's allocation distribution after K iterations per pass. We

measure

$$e_K = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{1}{2n(s)} \sum_{i=1}^{n(s)} \left\| p_{i,K}(\cdot \mid s) - p_{i,K_{\text{ref}}}(\cdot \mid s) \right\|_1 , \tag{28}$$

where $\mathcal{S}$ is the set of sampled states, $n(s)$ is the number of active agents, and $K_{\text{ref}} = 18{,}432$. Thus e_K measures the average difference in allocation probabilities over agents and sampled states. We use 0.005 as an empirical tolerance for this comparison; the longer run is a finite numerical reference. Among counts with complete three-seed, eight-case evaluations, we choose the smallest at or above the family baseline for which $e_K \leq 0.005$ and remains within this tolerance at larger tested counts. For example, on 256^2 we select 288 iterations at 16,000 agents and 4608 at 20,000 agents. The 9216- and 18,432-iteration references differ by less than 0.0005 in mean total variation in every evaluated condition.

Training occupancy spans 16.8–34.2% for the unstructured pool and 21.7–68.4% for the structured pool; the highest evaluated occupancies also extend beyond these ranges. We hypothesise that the abrupt throughput drops and increased iteration requirements reflect widespread competition for destination cells that is underrepresented during training. In this study, accelerated Sinkhorn is used only at evaluation. Training with accelerated updates or greater exposure to such congestion may reduce this sensitivity; we leave these possibilities to future work.

E.4 Long-horizon evaluation

Figure 8 reports throughput over 10,000 uninterrupted steps with 10,000 agents on warehouse, sortation, Berlin and the two random layouts. Throughput remains steady on these five layouts. Policy parameters remain frozen throughout evaluation.

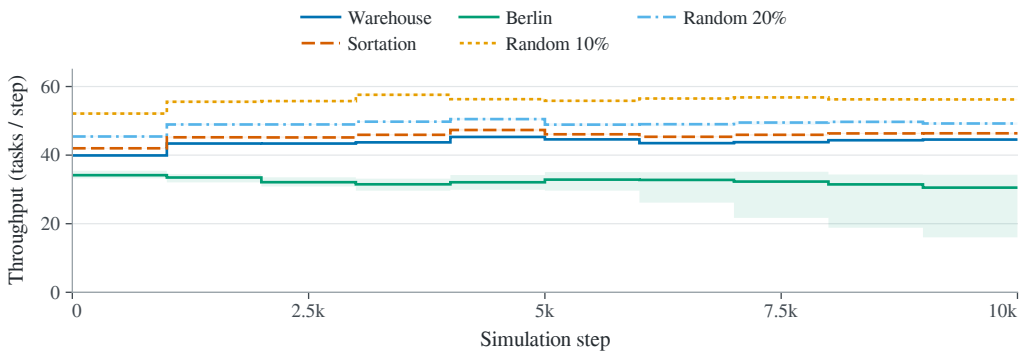


Figure 8: **Throughput over** 10,000 **uninterrupted steps with** 10,000 **agents.** Curves show means over released cases 0–7, with minimum–maximum case bands in non-overlapping 1000-step windows. Curves pool the three training seeds (eight cases each) and a frozen checkpoint, with 144 accelerated Sinkhorn iterations per pass.

Figure 9 compares the Paris continuations with the evaluated baselines. Over 10,000 steps our policy averages 23.4 ± 7.5 tasks per step at 144 accelerated iterations (mean $\pm$ SD over three training seeds) and 26.28 at 288 iterations, compared with 24.69 for SILLM-DG.

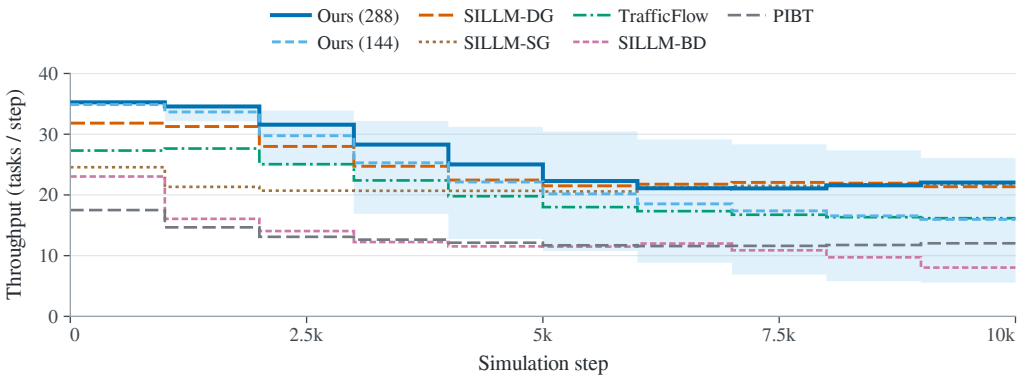


Figure 9: **Paris continuation with** 10,000 **agents.** Curves show mean throughput over released cases 0–7 in non-overlapping 1000-step windows. Ours uses 144 (three training seeds; the shaded band spans them) or 288 accelerated Sinkhorn iterations per pass. All learned policies remain fixed.

E.5 Additional large layouts

Table 15 evaluates the same unstructured policy on two 256^2 random layouts and the 194^2 Dragon Age: Origins map `ost003d` from the Moving AI benchmark (Sturtevant, 2012). Random 30% retains the largest connected component of a 30%-obstacle draw; Random 40% uses clustered obstacles. The three maps contain 45,049, 39,322 and 13,214 free cells, respectively.

Each method uses the same eight cases per layout and 2500 scored steps. Our policy uses terminal averaged weights from three training seeds and 288 plain allocation iterations. On the additional random layouts, SILLM-BD and SILLM-DG transfer their released RL checkpoints trained on 20%-obstacle random maps, TrafficFlow uses GP-R100-Re10-F2, and LLLG uses its 10,000-agent setting (planning window 8, horizon 5, two refinements, 10-second limit). WPPL uses 2500 LNS iterations on these layouts. PIBT uses SILLM's implementation on every layout, and hm+GPIBT uses OnlineGGO's handcrafted guidance on `ost003d`. All learned policies retain their trained parameters.

Layout	Agents	Ours	SILLM-BD	SILLM-DG	LLLG	TrafficFlow	PIBT	WPPL	hm+GPIBT
Random 30% (LCC)	10,000	18.86±2.26	11.76	16.24	16.68	17.50	13.91	14.26	—
Random 40% (clustered)	10,000	22.10±1.02	8.01	11.34	12.05	13.50	11.88	14.33	—
`ost003d`	6000	6.09±0.53	—	—	—	—	3.78	—	3.90

Table 15: Throughput on additional large layouts, in tasks per step. Our entries show mean ± sample standard deviation across three training seeds; each seed and each baseline average eight matching cases. Dashes indicate unavailable eight-case results.

F Mechanism controls and supporting evidence

The following controls use separate runs from the structured and unstructured training pools in Appendix B. Deployment evaluations use terminal averaged weights; learning curves use current parameters.

F.1 Actor structure at deployment scale

Table 16 compares the full actor with six retrained variants on the six deployment layouts at 10,000 agents. Each variant retains its policy family's training pool and 25,000-update budget. We evaluate terminal averaged weights from three training seeds on eight cases per layout, using 3200 steps and 144 plain allocation iterations on structured maps, and 2500 steps and 288 iterations on unstructured maps.

Controls. The depth variants use $H = 1$ or $H = 2$ during both training and evaluation. Removing allocation replaces both Sinkhorn passes and the reverse-move penalty with independent normalisation of route affinities, leaving the simulator unchanged. The combined variant also sets

$H = 1$. Removing the vector branch sets $G = 0$ in Eq. (16), retaining the action-score branch, goal-conditioned gains and depth corrections. The no-gain control fixes $\gamma_h(x) = 0$ in Eq. (3), retaining the shared scores, vector branch, depth corrections, geometric reference and allocation.

	Structured		Unstructured			
Actor	Warehouse	Sortation	Berlin	Paris	Random 10%	Random 20%
Full ($H = 4$)	42.31±0.06	44.20±0.02	32.99±0.83	33.17±1.90	54.19±0.04	47.53±0.04
$H = 1$	40.06±1.60	43.90±0.04	30.23±1.37	33.33±1.89	53.80±0.01	45.99±0.04
$H = 2$	41.64±0.05	44.06±0.03	26.73±3.00	32.13±2.59	54.14±0.05	46.95±0.06
Without allocation	40.77±2.26	44.10±0.07	18.76±3.44	22.66±3.41	53.40±0.04	44.81±0.34
$H = 1$, no allocation	22.24±16.61	43.82±0.05	20.65±0.61	20.79±1.69	53.04±0.05	42.91±0.33
Without vector field G	42.18±0.09	44.20±0.01	29.33±3.34	31.24±2.82	54.27±0.03	47.66±0.05
Without gains γ	42.00±0.17	44.23±0.03	29.17±1.91	31.29±1.81	54.34±0.05	47.91±0.04

Table 16: Actor-structure ablations at 10,000 agents. Entries are throughput in tasks per step, reported as mean ± sample standard deviation over three training seeds; each seed averages eight cases.

Four departures improve mean throughput over one on warehouse, sortation and both random maps; the city results do not follow a monotonic depth ordering. Removing allocation reduces Berlin throughput from 32.99 to 18.76 and Paris from 33.17 to 22.66. On warehouse, combining $H = 1$ with allocation removal gives a larger mean throughput reduction than either change alone. Removing the vector branch leaves structured and random-map throughput close to the full actor, while reducing throughput on both cities. Appendices F.3 and F.4 examine continuation depth and allocation across populations.

Goal-conditioned gains. Removing the gains reduces mean throughput from 42.31 to 42.00 on warehouse, from 32.99 to 29.17 on Berlin, and from 33.17 to 31.29 on Paris. Sortation throughput is nearly unchanged, while both random-map means increase slightly. The benefit of these gains therefore depends on the layout.

F.2 Learning progress

Figure 10 tracks validation throughput for selected variants in Table 16. Current parameters are evaluated every 200 updates on 32 procedural validation maps generated separately from the training pool, using 64 reset cases, deterministic proposals and 128 steps per case, with the first 15 excluded from scoring. Each training update collects 128×64 joint environment transitions.

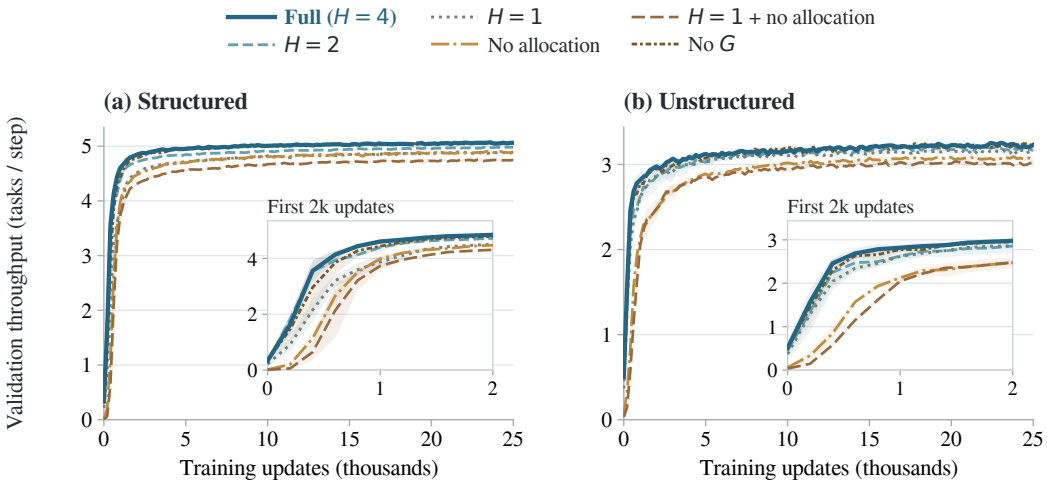


Figure 10: Current-parameter validation throughput during training for the structured (left) and unstructured (right) actor families. Lines show three-seed means and shaded bands one standard deviation. Insets show the first 2,000 updates; gaps indicate incomplete seed coverage.

On structured validation, $H = 4$ outperforms $H = 1$ from early training onward. Allocation improves early validation in both families, with a larger separation on unstructured maps that narrows as training progresses. Removing G has little effect on late validation in either family, despite the lower city throughput in Table 16.

F.3 CONTINUATION DEPTH

Training controls. The full policies use $H = 4$ departures. Depth controls set $H = 1$ or $H = 2$ during both training and evaluation, retaining their respective fixed training pools and the other training settings in Appendix B. Each run completes $25{,}000$ training iterations. These comparisons measure the effect of the trained continuation depth, including its forward calculation and training gradients. The convolutional network still observes the same traffic inputs at every depth.

Learning and deployment. Figure 4a uses three independent structured-policy training runs per depth. Current weights are evaluated every 200 iterations on 32 procedural rack/sortation validation maps generated separately from the training pool, using 64 reset cases, 128 steps with the first 15 excluded, deterministic proposals and 48 allocation iterations. Curves show equal-run means and sample standard deviations without smoothing. A missing $H = 4$ evaluation at iteration $19{,}000$ is left as a gap. Each training iteration collects 128×64 joint environment steps before two actor-update passes; the horizontal axis therefore compares sample budgets, rather than training time.

The annotations use terminal exponentially averaged weights (decay 0.9998), evaluated on eight large-warehouse instances per training run with $10{,}000$ agents, 3200 steps, 144 allocation iterations and layered reference tables. All steps contribute to throughput. Cases are averaged within each run before computing the mean and sample standard deviation across the three runs. The $H = 1, 2$ evaluations batch eight instances; $H = 4$ uses single-instance calls.

Corridor traffic. Figure 4b uses terminal averaged policies from three independent training runs per depth, evaluated separately on eight 33×57 warehouse instances with 400 agents and 3200 scored steps. Execution uses deterministic proposals, 144 allocation iterations and layered reference tables. We identify all 172 narrow passages from the walls alone: 84 horizontal passages with three interior cells and 88 vertical passages with two.

For neighbouring occupied cells, let $q_i(j)$ be agent i's probability of moving into agent j's cell after route reweighting and before allocation. Opposing demand is the mean of $q_i(j)q_j(i)$ over occupied internal passage edges and time steps. We first compute this rate within each case, then average equally over the eight cases and three training runs. It measures incompatible route preferences, not executed collisions; outward movement by both agents is valid. The traffic illustration shows actual moves and waits in passage 122, training run 0, case 0, at steps 1024–1151. The central passage and time window were fixed before inspecting the results. Colours track the entry side, not the current direction. Diamonds mark goal completions because passage cells can be goals, and returning towards the entry can therefore be productive.

At 400 agents, opposing demand falls by 39.4% with $H = 4$ and throughput rises by 9.4%. Actual axial moves per occupied passage-cell step remain similar (87.97% and 87.59% for $H = 1, 4$). A lower-load control at 200 agents gives smaller improvements of 5.9% in opposing demand and 2.9% in throughput, with overlapping training-run ranges for demand.

To distinguish the visited traffic states from the immediate effect of the route operator, we also apply all six intact policies to the same 1152 states, sampled every 128 steps from 128 to 3072 across all 48 policy–case rollouts. On these common states, mean opposing demand is 5.69% for $H = 1$ and 5.36% for $H = 4$, with overlapping training-run ranges. Both depth variants produce lower opposing demand on states generated by $H = 4$. The larger rollout difference concerns the traffic produced over time; the immediate difference on identical states is smaller.

Evaluation under load. The following population comparisons use terminal averaged weights. The structured policy is evaluated on the 33×57 warehouse with alternating workstation–storage tasks; the unstructured policy uses the 32×32 random map with uniform tasks. The maps contain 1091 and 819 free cells, respectively. Occupancy is population divided by this count. Each point uses eight cases, 1000-step episodes and deterministic proposals; all steps count. Plain allocation

uses 144 iterations on warehouse and 288 on random, fixed across populations and continuation depths. Reference tables use the layered and sweep builders, respectively. Within each map, full and depth-control evaluations use the same initial-state and task files, with eight instances batched in one call.

Table 17 reports throughput by training seed. The comparisons below average training seeds 0–2.

At approximately 37% occupancy, $H = 4$ raises mean throughput over $H = 1$ by 8.4% on warehouse and 7.9% on random. The depths give similar throughput at light load on random; depth ordering becomes nonmonotonic at the highest loads, where warehouse throughput also deteriorates.

			$H = 1$			$H = 2$			$H = 4$		
Map	Agents	Occ. (%)	s0	s1	s2	s0	s1	s2	s0	s1	s2
Warehouse	200	18.3	5.092	5.100	5.086	5.141	5.158	5.129	5.223	5.246	5.225
	300	27.5	6.943	6.991	6.973	7.070	7.145	7.055	7.382	7.440	7.340
	400	36.7	8.163	8.364	8.329	8.440	8.654	8.426	9.052	9.103	8.984
	500	45.8	8.667	9.192	7.203	6.533	9.676	9.165	9.925	9.470	10.069
	600	55.0	3.509	3.574	2.089	3.086	6.167	5.055	5.030	3.266	9.561
Random	20	2.4	0.855	0.853	0.847	0.849	0.853	0.849	0.844	0.835	0.836
	50	6.1	2.090	2.088	2.088	2.094	2.092	2.084	2.063	2.056	2.061
	100	12.2	3.931	3.944	3.940	3.967	3.975	3.953	3.943	3.949	3.933
	200	24.4	6.755	6.830	6.805	6.947	6.997	6.959	7.042	7.051	7.072
	300	36.6	8.311	8.569	8.526	8.865	8.902	8.919	9.225	9.050	9.302
	500	61.1	8.761	9.338	8.965	7.694	7.177	9.852	10.622	10.474	10.861

Table 17: Throughput by continuation depth and training seed. Each entry averages the same eight cases for its layout and population.

F.4 Destination allocation under load

The allocation control replaces both allocation passes and the reverse-move penalty with independent normalisation of next-move affinities at the learned allocation temperature. The convolutional network, goal-conditioned route calculation and simulator are unchanged. Controls use the unstructured training pool, with the full policy's batch, rollout length, update count and optimiser settings.

Table 18 compares terminal averaged weights at populations 6000, 10,000 and 12,000, on the two 256^2 random maps. Each cell averages eight released start/task cases over 2500 steps, with argmax proposals, layered route tables and 288 plain allocation iterations in the full policy. Results average three training runs per condition (seeds 0–2).

		Full			Without allocation		
Layout	Agents	Seed 0	Seed 1	Seed 2	Seed 0	Seed 1	Seed 2
Random 20%	6,000	30.63	30.67	30.62	30.20	30.28	30.25
Random 20%	10,000	47.50	47.58	47.51	44.42	45.06	44.95
Random 20%	12,000	54.40	54.52	54.49	22.55	32.28	28.72
Random 10%	6,000	33.25	33.26	33.26	33.14	33.17	33.16
Random 10%	10,000	54.15	54.23	54.19	53.35	53.44	53.40
Random 10%	12,000	64.03	64.14	64.09	62.43	62.60	62.53

Table 18: Full and allocation-removed policies on sparse and dense random layouts. Throughput averages eight cases per training seed, over all 2500 steps.