

Stigmergic Graph Memory: An Environment-Aware Approach for Many-to-Many Multi-Agent Pickup and Delivery

Aditya Dutta, Joon-Seok Kim*

Emory University
201 Dowman Drive
Atlanta, GA 30322 USA
aditya.dutta@emory.edu, joonseok.kim@emory.edu
*Corresponding author

Abstract

Automated fulfillment warehouses must continuously assign and execute pickup-and-delivery work while avoiding congestion. In many-to-many Multi-Agent Pickup and Delivery (MAPD), a request specifies a stock-keeping unit rather than fixed endpoints, requiring the controller to select an agent, source, and destination before path planning. Existing graph-guidance methods primarily influence routing after goals are fixed, leaving endpoint instantiation uninformed by recent traffic. We introduce Stigmergic Graph Memory (SGM), a bounded, decaying memory layer that records recent execution signals on warehouse nodes and directed edges to rank feasible endpoints and route preferences without altering collision constraints or planner validity. Across paired request streams on five layouts, three load levels, and 25 seeds per condition, SGM outperforms two reconstructed many-to-many allocation baselines in all 15 map-load conditions, with paired throughput gains of 20.5–36.7%. These results show that recent execution memory can improve warehouse throughput by shaping which feasible goals enter the planner, not only how agents travel to already fixed goals.

1 Introduction

Automated warehouses increasingly rely on mobile-robot fleets for continuous pickup and delivery, making coordination essential for throughput (Wurman, D’Andrea, and Mountz 2008; Boysen, de Koster, and Weidinger 2019). This problem is commonly formulated as Multi-Agent Pickup and Delivery (MAPD), which combines multi-robot task allocation (MRTA) with collision-free Multi-Agent Path Finding (MAPF) in a shared environment (Stern et al. 2019; Ma et al. 2017; Xu et al. 2022; Korsah, Stentz, and Dias 2013). The underlying optimization variant of multi-robot path planning is NP-hard (Surynek 2010; Yu and LaValle 2013), making scalable coordination especially important in large, continuously operating warehouses. While most MAPD approaches assume fixed pickup and delivery locations (Ma et al. 2017; Liu et al. 2019), warehouse requests often identify a stock-keeping unit (SKU) available at multiple sources and deliverable to multiple destinations. Robots must therefore jointly choose an agent, pickup location, and delivery location, yielding a many-to-many MAPD problem.

Recent work on Many-to-Many Multi-Agent Pickup and Delivery (M2M) formulates this setting as a four-dimensional assignment problem over agents, tasks, pickup locations,

and delivery locations (Schneider et al. 2026). By exploiting inventory-location flexibility, M2M reports substantial gains over prior approaches, including up to 22,000 additional completed tasks in its eight-hour simulations (Schneider et al. 2026).

However, M2M evaluates assignments before the resulting traffic is realized by the path planner. Its objective relies on estimated travel and inventory-distribution costs and therefore does not directly account for congestion, waiting time, or contention for shared resources. As a result, assignments that appear inexpensive in isolation may repeatedly direct robots toward the same constrained aisles, intersections, or service locations. In dense warehouse environments, these locally attractive decisions can collectively generate traffic patterns that reduce overall system efficiency.

In this paper, we introduce **Stigmergic Graph Memory (SGM)**, a lightweight memory layer that exposes recent execution patterns to endpoint scoring and routing. Inspired by stigmergic coordination (Grassé 1959; Theraulaz and Bonabeau 1999), SGM maintains decaying signals on warehouse nodes and directed edges and maps them to endpoint and route preferences. This lets the controller account for recent traffic when selecting both feasible endpoints and routes.

We compare SGM with reconstructions of M2M and its SKU-aware variant under a common simulator, planner, and paired-request protocol. Across five layouts, three load levels, and 25 seeds per condition, SGM improves throughput in all 15 map-load conditions, with paired gains of 20.5–36.7%. Routing and matched-cap controls indicate that these gains are not explained by route costs or candidate budgets alone.

We make three contributions. **(1)** We frame endpoint instantiation in many-to-many MAPD as a congestion-control decision. **(2)** We introduce SGM, a decaying graph-memory layer that informs endpoint ranking and route preferences. **(3)** We evaluate SGM through paired benchmarks, routing controls, sensitivity analyses, component ablations, and matched-cap comparisons.

2 Related Work

MAPD and many-to-many task allocation. Multi-Agent Pickup and Delivery (MAPD) combines multi-robot task allocation (MRTA) with Multi-Agent Path Finding (MAPF) for lifelong task execution. MAPF computes collision-free paths on a graph subject to vertex and edge-conflict con-

straints (Stern et al. 2019), while lifelong MAPF repeatedly assigns new goals and replans over bounded horizons for scalability (Li et al. 2021). MAPD introduces online pickup-and-delivery tasks (Ma et al. 2017); related variants include integrated task assignment and path planning (Chen et al. 2021), multi-goal MAPD (Xu et al. 2022), and deadline-aware MAPD (Wu et al. 2021; Makino and Ito 2024). A broader survey situates MAPD among MAPF, learning-based coordination, and warehouse applications (Lau and Sengupta 2022).

Classical MAPF methods include reservation-based planning, conflict-based search, safe-interval planning, and scalable priority-based approaches (Silver 2005; Sharon et al. 2015; Phillips and Likhachev 2011; Okumura et al. 2022). Implementation-oriented MAPD variants also account for robot kinematics and continuous execution constraints (Hönig et al. 2016; Ma et al. 2019b). Integrated MAPD methods further use large-neighborhood search to improve coupled allocation and path-planning decisions (Chen et al. 2021; Ahuja et al. 2002; Pisinger and Røpke 2010).

Many-to-many MAPD generalizes fixed pickup-and-delivery tasks by allowing multiple feasible sources and destinations for each SKU request (Schneider et al. 2026). M2M addresses this problem by formulating task allocation as a four-dimensional assignment over agents, tasks, pickup locations, and delivery locations (Schneider et al. 2026), significantly improving throughput over traditional one-to-one MAPD approaches. We therefore use M2M and its SKU-aware variant, M2M-wSKU, as our primary many-to-many MAPD baselines.

Graph guidance and stigmergic coordination. Highways, traffic-flow costs, and guidance graphs improve lifelong MAPF by biasing movement toward favorable routes (Li and Sun 2023; Chen et al. 2024; Zhang et al. 2024; Zang et al. 2025; Zhang et al. 2026). These methods principally affect routing after task goals have been fixed. In contrast, SGM reads *recent execution history* when ranking feasible source–destination (or endpoint) instantiations, which makes our solution unique, as well as when assigning route preferences after endpoints are selected. Table 1 summarizes the resulting distinction across four aspects.

Table 1: Decision interfaces of method families in many-to-many MAPD.

Family	Source	Dest.	Recent exec.	Routing
M2M / M2M-wSKU allocation	Yes	Yes	No	No
Highways / guidance graphs	No	No	Usually no	Yes
Traffic-flow costs	No	No	Sometimes	Yes
SGM (ours)	Yes	Yes	Yes	Yes

Stigmergy describes indirect coordination through environmental traces (Grassé 1959; Theraulaz and Bonabeau 1999). Ant-colony optimization provides a familiar algorithmic use of decaying environmental traces (Dorigo and Gambardella 1997; Bonabeau, Dorigo, and Theraulaz 1999). SGM adopts stigmergy as a typed, decaying graph-memory

abstraction for many-to-many MAPD, rather than as a pheromone-based routing algorithm.

3 Problem Setting

We model the warehouse as an undirected graph $G = (V, E)$ with agents $\mathcal{R} = \{a_1, \dots, a_I\}$. At each discrete timestep, an agent either waits at its current vertex or moves along an adjacent edge. A joint motion plan must satisfy the standard MAPF constraints: no two agents may occupy the same vertex at the same timestep, and no two agents may traverse the same edge in opposite directions during the same timestep.

Tasks arrive online. Each request is $\tau_n = (q_n, \mathcal{S}_n, \mathcal{D}_n, r_n)$, where q_n is a stock-keeping unit (SKU), $\mathcal{S}_n \subseteq V$ is the set of feasible source locations containing q_n , $\mathcal{D}_n \subseteq V$ is the set of feasible delivery locations, and r_n is its release time. At time t , $\mathcal{T}(t)$ denotes the set of released but uncompleted requests. Instantiating τ_n selects an agent a_i , source $s \in \mathcal{S}_n$, and destination $d \in \mathcal{D}_n$, yielding an assignment (a_i, τ_n, s, d) . The assigned agent must visit s before d .

Let $C(H)$ denote the number of tasks completed by the end of planning horizon H . The objective is to maximize long-term throughput, $\frac{C(H)}{H}$, subject to task-feasibility and collision-avoidance constraints. The controller must therefore determine task instantiations and collision-free paths over time.

The central challenge is that endpoint instantiation shapes future traffic before path planning begins. A static allocation objective can repeatedly select individually short source–destination pairs that collectively overload the same corridors, storage areas, or service locations. SGM addresses this decision point by using recent execution history to rank feasible endpoint choices before they become planner goals.

4 Stigmergic Graph Memory

SGM operates within an event-driven rolling-horizon controller based on Rolling-Horizon Collision Resolution (RHCR) (Li et al. 2021), using Priority-Based Search (PBS) as the MAPF backend (Ma et al. 2019a). Figure 1 shows a schematic of SGM architecture. At each control cycle, the system updates memory from recent execution events, instantiates eligible requests, computes collision-free paths for active goals, executes one action from each path, and updates pickup, delivery, and completion states. Replanning occurs when an agent completes a pickup or delivery segment, when a planned path is exhausted, or when execution requires a route repair.

SGM is a shared, decaying memory layer between task instantiation and path planning. It stores separate numerical *channels* for recent node and directed-edge events, which endpoint steering and route guidance read differently.

4.1 Execution Traces

SGM does not store a single generic congestion value. Instead, it maintains separate decaying memory channels for different kinds of recent execution information. Node channels track local congestion, endpoint demand, unplanned

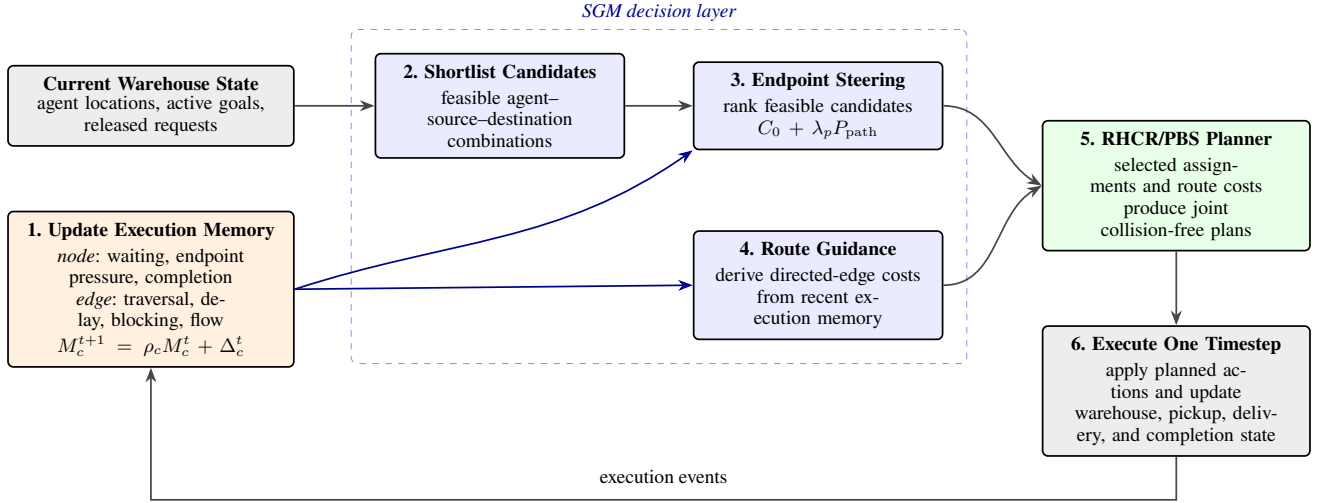


Figure 1: SGM execution cycle. The current warehouse state provides released requests, agent locations, and active goals. SGM updates typed execution memory, shortlists feasible agent–source–destination candidates, ranks endpoint choices, and derives route costs from recent execution history. RHCR/PBS computes a collision-free joint plan. Executing one timestep updates warehouse state and produces the events used in the next memory update.

waiting, pre-execution blocking, and successful task completion. Directed-edge channels track traversal, congestion, delay, pre-execution blocking, and successful directional flow.

Endpoint pressure measures recent demand for a location. When request τ_n is released, its demand is distributed over feasible endpoints: each source $s \in \mathcal{S}_n$ receives a source-pressure deposit proportional to $1/|\mathcal{S}_n|$, and each destination $d \in \mathcal{D}_n$ receives a destination-pressure deposit proportional to $1/|\mathcal{D}_n|$. Completion memory records successful deliveries associated with an endpoint, while waiting and delay memory record unplanned lack of progress. Planned waits in an otherwise valid route do not create waiting or delay deposits.

Pre-execution blocking pressure is not an executed collision. Before actions are applied, the execution layer checks proposed next moves. If a proposed move would create a vertex conflict or edge swap, the affected agent is held in place and the system records a blocking event at the relevant node and directed edge. The conflicting motion is therefore prevented before execution, so all executed states remain collision-free.

For each channel c and graph element x , the corresponding memory value evolves as

$$M_c^{t+1}(x) = \rho_c M_c^t(x) + \Delta_c^t(x), \quad (1)$$

where $\rho_c \in (0, 1)$ is the channel-specific retention factor and $\Delta_c^t(x) \geq 0$ is the deposit produced by events at timestep t . The empirically derived configuration uses retention values of 0.85 for waiting and delay, 0.90 for blocking and endpoint-pressure signals, 0.92 for congestion, traversal, and flow signals, and 0.95 for completion signals. An optional corridor adjustment increases retention only up to a cap of 0.99. Recent events therefore have greater influence than stale events.

Endpoint steering reads endpoint pressure, waiting, and blocking signals, whereas route guidance reads directional

traversal, delay, blocking, and flow signals. Keeping these channels separate distinguishes endpoint demand from movement difficulty and directional traffic.

4.2 Endpoint Steering

For each feasible many-to-many candidate (a_i, τ_n, s, d) , where $s \in \mathcal{S}_n$ and $d \in \mathcal{D}_n$, SGM starts from the M2M-style baseline allocation cost $C_0(a_i, \tau_n, s, d)$ (Schneider et al. 2026). In the SKU-aware variant, C_0 includes the M2M-wSKU inventory-distribution term. SGM then adds a memory-derived path penalty:

$$C(a_i, \tau_n, s, d) = C_0(a_i, \tau_n, s, d) + \lambda_p P_{\text{path}}(a_i, s, d). \quad (2)$$

For the reported SGM configuration, the baseline term is $C_0 = d(a_i, s) + d(s, d) + C_{\text{SKU}}(\tau_n, s, d)$. The memory term sums current directed-edge penalties along unit-cost shortest-path proxies from the agent to the source and from the source to the destination:

$$P_{\text{path}}(a_i, s, d) = \sum_{(u,v) \in \pi(a_i, s) \cup \pi(s, d)} r_t(u, v), \quad (3)$$

where π denotes the sampled unit-cost shortest path and $r_t(u, v) = \tilde{c}_t(u, v) - 1$ is the bounded directed-edge penalty defined in Eq. 4. These paths are scoring proxies; RHCR/PBS later computes the final joint plan.

To keep endpoint steering tractable, SGM evaluates SKU-compatible source and destination shortlists capped at 64 locations each, with at most 32 source–destination pairs per request. It can therefore steer among feasible, distance-comparable endpoint choices.

4.3 Route Guidance

For routing, SGM converts recent directed-edge memory into bounded positive traversal costs:

$$\tilde{c}_t(u, v) = 1 + \min\{\kappa, \lambda_r R_t(u, v)\}, \quad (4)$$

where (u, v) is a legal directed move, λ_r controls the influence of route guidance, and κ bounds the maximum increase over the unit free-flow cost. Thus, every legal edge retains positive cost, with $1 \leq \tilde{c}_t(u, v) \leq 1 + \kappa$.

The unbounded memory signal is

$$R_t(u, v) = \max\{0, \alpha_c c_t(v) + \alpha_f f_t(u, v) + \alpha_w w_t(u, v) + \alpha_b b_t(u, v) - \alpha_s q_t(u, v) + o_t(u, v)\},$$

where $c_t(v)$ is normalized congestion at the destination node, $f_t(u, v)$ is recent blocking pressure, $w_t(u, v)$ is recent delay, $b_t(u, v)$ measures reverse-direction flow, $q_t(u, v)$ measures successful flow in the forward direction, and $o_t(u, v)$ is the discounted marginal planned-occupancy cost over the guidance horizon. Positive terms discourage recently congested, delayed, blocked, or counter-flow movement, whereas the successful-flow term rewards directions that have recently supported productive traversal.

The reported configuration uses $\lambda_r = 0.2$ and $\kappa = 0.1$; Supplementary Tables S3–S4 specify the remaining channel coefficients and controller settings. RHCR (Li et al. 2021) uses PBS (Ma et al. 2019a) to compute a collision-free joint plan under these costs.

4.4 Controller Variants

A controller determines how tasks are allocated, whether memory is used for endpoint steering and route guidance, when replanning occurs, and how queued work is handled after a route repair. We evaluate two controllers over the same SGM memory representation.

SGM is the primary throughput-oriented controller. At each allocation opportunity, it uses path-memory scoring to rank feasible endpoints and memory-derived route costs when planning active goals. It also preserves queued tasks across route repairs. A *route repair* recomputes paths for active goals after an agent completes a pickup or delivery segment, when an active route is exhausted, or when execution requires replanning. Queue preservation means that unstarted tasks already assigned to an agent remain in that agent’s queue rather than returning to the global task pool.

ASGM is a guarded, Adaptive variant of the same SGM memory layer. Under weak congestion evidence, it uses the M2M-wSKU-style allocation objective and unit-cost routing. As waiting, backlog, blocked movement, or reduced agent availability indicate sustained congestion, it can activate memory-guided endpoint steering and route guidance. Weighted routes are audited against unit-cost alternatives and retained only when they satisfy the guard criteria.

Table 2 summarizes the controller policies used in the main benchmark.

Table 2: Controller policies in the main benchmark.

Method	SKU	Endpoint memory	Route memory	Queue handling
M2M	No	No	No	Rebuild
M2M-wSKU	Yes	No	No	Rebuild
ASGM	Yes	Adaptive	Adaptive	Preserve
SGM	Yes	Continuous	Continuous	Preserve

4.5 Validity Boundary

Proposition (feasibility preservation). For any endpoint assignment selected by SGM, the RHCR/PBS backend receives the original warehouse graph, current agent states, task-feasibility constraints, and collision predicates. Therefore, every joint plan accepted and executed by the backend is collision-free and task-feasible under the original MAPD constraints.

Proof. SGM changes only scalar scores used to rank feasible endpoint candidates and bounded positive costs used to rank legal directed-edge traversals. It does not add or remove graph edges, alter agent states, relax endpoint feasibility, or modify vertex- and edge-conflict predicates. RHCR/PBS remains solely responsible for accepting a joint plan. Thus, any executed plan satisfies the original task-feasibility and collision-avoidance constraints.

At endpoint instantiation, memory terms are applied only while ranking a finite set of feasible candidates. At routing, SGM changes only the cost of legal directed moves; for every legal edge (u, v) ,

$$1 \leq \tilde{c}_t(u, v) \leq 1 + \kappa.$$

Because endpoint-memory contributions are clamped and routing costs remain positive and bounded, SGM cannot create infeasible assignments, negative edge costs, zero-cost cycles, or infinite-cost finite paths. It is therefore a bounded preference layer over feasible endpoint and routing decisions. $\square$

5 Experimental Protocol

We evaluate SGM and its adaptive variant, ASGM, against M2M and its SKU-aware variant, M2M-wSKU. The main small-map benchmark contains $5 \times 3 \times 25 \times 4 = 1500$ method–map–fleet-level–seed runs: five warehouse layouts (Figure 2), three parking-normalized fleet levels, and 25 paired seeds.

Each layout contains 93 designated parking cells. We evaluate low, medium, and high fleet levels with 28, 56, and 84 agents, corresponding to 30.1%, 60.2%, and 90.3% of parking capacity, respectively. These fleet sizes do not exceed the number of designated parking endpoints, consistent with the well-formed-infrastructure condition that provides at least one non-task endpoint per agent (Ma et al. 2017). Each run spans $H = 3600$ timesteps with paired request replay where the step size is one second, so all methods face the same release stream within each map–fleet-level–seed cell.

Let $C(t)$ denote cumulative completed tasks at timestep t . We report $C(H)$ as final completed tasks and define horizon throughput as $C(H)/H$. Because every main run uses the same horizon, these measures induce the same method ranking; we use *cumulative completed tasks* for temporal curves and *throughput* only for rates. We also report final-window activity, service time, blocked moves, and allocation cost.

Statistical analysis. For each map–fleet-level condition, we compare seed-matched final completed-task counts using two-sided Wilcoxon signed-rank tests (Wilcoxon 1945). We control the family-wise error rate across the 30 planned

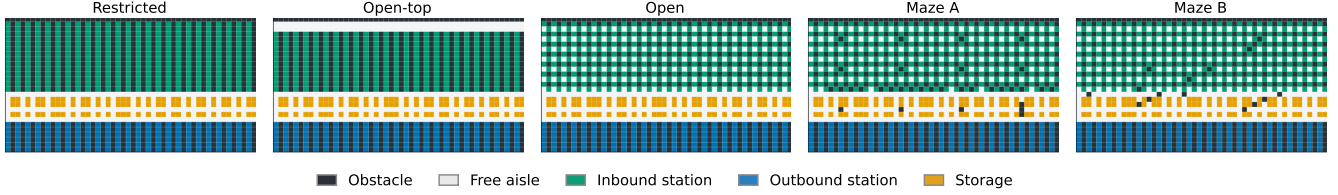


Figure 2: The five warehouse layouts, each 50×27 cells with 93 designated parking cells. Restricted, open-top, and open are benchmark warehouse layouts from the many-to-many MAPD setting; maze A and maze B provide structurally distinct corridor networks at the same scale. Supplementary Table S5 gives the per-layout cell counts.

comparisons of SGM with M2M and M2M-wSKU using the Holm step-down procedure (Holm 1979). Paired mean percentage gains and their 95% confidence intervals are reported as effect sizes; the tests assess whether paired differences are centered at zero. Supplementary Table S1 gives the complete test record.

Baseline provenance. Both baselines are many-to-many allocators reconstructed from the official repository of Schneider et al. (2026): M2M scores endpoints using travel-distance estimates, and M2M-wSKU adds the inventory-distribution term. We run all methods under the same RHCR/PBS planner, paired request replay, maps, and initial conditions. The baseline reconstructions rebuild queued tasks each timestep, whereas SGM and ASGM preserve queued tasks under event-driven replanning. Supplementary Tables S2 and S4 give the integration and method-specific settings.

We conduct three component ablations over the same $5 \times 3 \times 25$ map-fleet-level-seed cells. Endpoint-only retains path-memory endpoint steering but uses unit route costs; routing-only retains memory-weighted routing but removes endpoint-memory scoring; and no-queue-preserve retains both memory interfaces while rebuilding queued tasks each timestep. All ablations use the same paired request streams and RHCR/PBS planning stack as the main methods.

To isolate endpoint steering from routing guidance alone, we evaluate two M2M-wSKU routing controls on the three benchmark warehouse layouts: *M2M-wSKU+recent routing*, which uses SGM recent-traffic edge costs without endpoint memory, and *M2M-wSKU+static highway*, which uses a fixed parity highway. These controls use five paired seeds at the medium and high fleet levels. We also conduct one-factor sensitivity studies, matched-candidate-cap controls, and a five-seed high-fleet transfer study on medium open and open-top layouts. These additional studies support the main 25-seed small-map benchmark rather than replace it.

All main-method comparisons use the same RHCR/PBS planning stack with a planning window of 512, paired request replay, and the shared maps and initial conditions summarized in Table 2. For reproducibility, Supplementary Table S4 reports method-specific shortlist sizes, endpoint-pair limits, memory weights, guard criteria, and controller settings. Matched-cap controls test whether candidate-search budgets alone explain the observed gains.

6 Results

6.1 SGM improves completed-task throughput

Table 3 reports cumulative completed tasks at three checkpoints on the restricted, open-top, and open benchmark layouts. SGM is the best method in 26 of the 27 displayed map-fleet-checkpoint cells. The only exception is the earliest checkpoint on the high-fleet open layout, where M2M-wSKU is ahead at timestep 600; SGM overtakes it by timestep 1800 and remains ahead at the end of the horizon. This isolated early deficit reflects SGM’s deliberate warm-up behavior: before sufficient execution events accumulate, its decaying memory exerts limited influence, whereas M2M-wSKU can immediately apply its static SKU-aware allocation objective.

The temporal pattern is consistent across layouts. At timestep 600, SGM is already competitive with, and often exceeds, the stronger M2M baseline. Its larger advantage emerges as traffic accumulates: by timestep 1800, SGM improves on the stronger baseline by roughly 12–29% in the displayed conditions, and its final gains range from 19.6–36.2%. Thus, the benefit is not a transient start-up effect. Memory-guided endpoint selection progressively prevents the repeated concentration of work in the same constrained regions.

The full five-layout benchmark gives the same conclusion. Across all 15 map-fleet conditions, SGM exceeds both reconstructed many-to-many baselines under paired request replay, with paired final-throughput gains of 20.5–36.7%. Supplementary Table S6 gives the complete five-layout terminal outcomes. The planned paired tests in Section 5 evaluate these differences under the shared simulator and request streams; Table 3 shows how the final advantage develops over time rather than only reporting the terminal value.

6.2 The effect persists across distinct warehouse structures

Figure 3 extends the time-resolved view to the two maze layouts. These maps differ from the benchmark warehouses in their corridor topology and available detours, yet SGM remains ahead of the stronger M2M baseline at medium and high fleet levels.

The routing controls isolate the distinction between learned execution memory and static routing biases. On the three benchmark warehouse layouts at medium and high fleet levels, adding recent-traffic routing alone to M2M-wSKU

Table 3: Cumulative completed tasks at three checkpoints on the benchmark layouts (mean over 25 paired seeds). *Best M2M* is the stronger of M2M and M2M-wSKU at that checkpoint; its label appears below the value. Parenthetical values for ASGM and SGM are percentage differences from that baseline. Bold indicates the highest value among all four methods in a condition-checkpoint cell.

Layout	Fleet	Timestep 600			Timestep 1800			Timestep 3600		
		Best M2M	ASGM	SGM	Best M2M	ASGM	SGM	Best M2M	ASGM	SGM
Restricted	Low	1140.9	1125.4	1144.6	2736.0	2727.2	3273.6	4610.2	4647.6	6122.7
		wSKU	−1.4%	+0.3%	wSKU	−0.3%	+19.6%	wSKU	+0.8%	+32.8%
		1529.0	1566.7	1697.3	3346.6	3412.4	4329.1	5542.2	5628.4	7551.2
	Medium	wSKU	+2.5%	+11.0%	wSKU	+2.0%	+29.4%	wSKU	+1.6%	+36.2%
		1808.0	1702.2	1877.8	3796.2	4125.8	4570.5	6280.5	6954.8	7820.6
	High	wSKU	−5.9%	+3.9%	wSKU	+8.7%	+20.4%	wSKU	+10.7%	+24.5%
Open-top	Low	1091.4	1116.8	1149.2	2639.8	2683.5	3231.2	4313.3	4345.8	5770.4
		wSKU	+2.3%	+5.3%	wSKU	+1.7%	+22.4%	wSKU	+0.8%	+33.8%
		1412.6	1502.3	1655.2	3199.8	3323.5	4152.4	5061.9	5255.4	6877.9
	Medium	wSKU	+6.4%	+17.2%	wSKU	+3.9%	+29.8%	M2M	+3.8%	+35.9%
		1632.6	1569.1	1809.4	3445.5	3875.5	4393.2	5573.0	6156.4	7243.4
	High	wSKU	−3.9%	+10.8%	wSKU	+12.5%	+27.5%	M2M	+10.5%	+30.0%
Open	Low	1106.4	1061.9	1111.0	2547.9	2494.8	3076.1	4386.1	4423.4	5820.2
		wSKU	−4.0%	+0.4%	wSKU	−2.1%	+20.7%	wSKU	+0.9%	+32.7%
		1468.0	1428.0	1527.1	3156.8	3178.0	3919.6	5269.4	5346.8	6949.2
	Medium	wSKU	−2.7%	+4.0%	wSKU	+0.7%	+24.2%	wSKU	+1.5%	+31.9%
		1805.2	1524.1	1706.9	3734.0	3672.0	4165.3	6286.8	6512.2	7516.7
	High	wSKU	−15.6%	−5.4%	wSKU	−1.7%	+11.5%	wSKU	+3.6%	+19.6%

changes throughput by $-0.1 \pm 2.1\%$, while a fixed parity highway changes it by $-1.4 \pm 2.5\%$. In contrast, SGM improves throughput by $29.2 \pm 5.7\%$ in the same comparison. Routing guidance alone is therefore insufficient: the dominant improvement comes from using recent execution to choose which feasible source-destination pair becomes the task goal.

Additional controls support this interpretation. One-factor sensitivity sweeps retain substantial gains across the examined memory settings, and matched-candidate-cap experiments show that the advantage is not explained by evaluating more endpoint pairs (Supplementary Table S9). On the medium-scale transfer study, SGM also improves completed tasks by $7.3 \pm 3.3\%$, indicating that the controller remains beneficial outside the primary small-map benchmark (Supplementary Table S10).

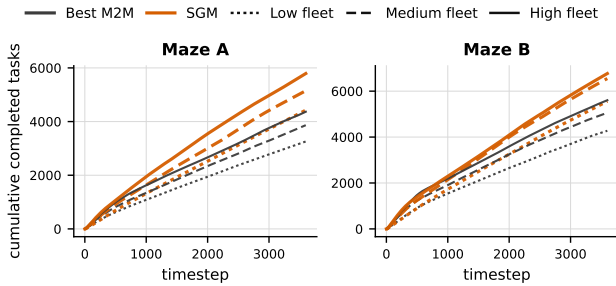


Figure 3: Cumulative completed tasks on maze A and maze B. Curves average 25 paired seeds. Charcoal denotes the stronger M2M baseline in each map-fleet cell, vermilion denotes SGM, and line style denotes fleet level.

6.3 Endpoint steering drives throughput; route guidance improves execution

The component ablations identify the roles of the two SGM interfaces. Endpoint-only retains memory-guided endpoint scoring but uses unit routing costs. It matches full SGM in final completed tasks across the small-map ablation benchmark: averaged over the 15 map-fleet conditions, endpoint-only completes 6395.2 tasks and full SGM completes 6394.3. This is a direct mechanistic result, not a weakness of the method: it establishes endpoint instantiation as the primary throughput-control interface.

Route guidance improves the quality and cost of execution while preserving that throughput. Relative to endpoint-only, full SGM lowers total planner time in every ablation condition, reduces waiting in 13 of 15 conditions, and reduces blocked moves and blocked-motion replans in 12 conditions. Thus, endpoint steering determines which work enters the planner, while route guidance reduces the execution burden of serving that work.

The remaining ablations reinforce this decomposition. Routing-only performs substantially worse than controllers with endpoint memory, showing that routing cannot recover from poor endpoint instantiation after goals are fixed. Removing queue preservation also reduces completed tasks and increases congestion, because repeated reconstruction discards useful allocation continuity. Together, these results show that SGM's throughput gain is produced by memory-guided endpoint steering, while its route guidance and queue-preserving execution policy make that throughput more efficient to plan and execute.

ASGM serves a different operating point. It reduces allocation overhead relative to the full controller, but sacrifices absolute completed-task throughput. We therefore treat ASGM as an adaptive compute-throughput trade-off, whereas SGM is the primary controller for the main performance claims.

Table 4: Full SGM versus endpoint-only, averaged over the 15 small-map map–fleet conditions. Lower is better for planner time, waiting, blocked moves, and blocked-motion replans.

Metric	Endpoint-only	SGM	Change
Final completed tasks	6395.2	6394.3	−0.9
Planner time (s)	295.6	268.8	−9.1%
Waiting events / step	27.24	27.03	−0.8%
Blocked moves	19.1	16.9	−11.5%
Blocked-motion replans	8.84	7.90	−10.6%

7 Discussion

Endpoint instantiation is the primary throughput-control interface. The central result is that many-to-many MAPD congestion can be shaped before a path is planned. SGM uses recent execution history to avoid repeatedly committing new tasks to source–destination combinations associated with waiting, blocking, and concentrated traffic. The endpoint-only ablation identifies this as the dominant throughput mechanism: it retains 99.3–100.5% of full SGM throughput across the evaluated small-map conditions. Thus, most of the performance gain comes from reading graph memory when feasible endpoints are instantiated, rather than only after goals have been fixed.

Why the complete controller retains route guidance. Endpoint-only is deliberately a throughput-focused ablation, not the recommended deployment controller. Although it matches full SGM in final completed tasks, route guidance improves the operational cost of achieving that throughput. Across the 15 small-map map–fleet conditions, full SGM reduces planner time by 9.1% in every condition, reduces waiting events in 13 conditions, and reduces blocked moves and blocked-motion replans in 12 conditions. These effects are consistent with the intended role of route guidance: after endpoint steering selects where work should be served, directed-edge memory helps the planner serve that work with less contention. Endpoint steering controls which goals enter the planner; route guidance reduces the execution and planning burden of serving them.

Controller choice and computational cost. SGM is the primary controller when absolute completed work is the objective: it has the highest completed-task count at every fleet level. ASGM provides a distinct adaptive operating point. It limits allocation computation under high load and achieves the strongest compute-normalized throughput at the high fleet level, while retaining a substantial improvement over both M2M baselines. Thus, SGM is the appropriate choice when throughput is paramount, whereas ASGM is useful when allocation computation is a tighter operational constraint.

Robustness, controls, and transfer. The control studies distinguish SGM from two common routing-only alternatives. Adding recent routing costs or a static highway after M2M-wSKU has fixed endpoints leaves throughput near the M2M-wSKU baseline, whereas SGM gains substantially on the same cells. Matched-candidate-cap studies show that the

Table 5: Throughput–compute trade-off over five layouts and 25 seeds per condition. */M cand.* is aggregate completed tasks divided by aggregate candidate evaluations in millions; */comp. s* is aggregate completed tasks divided by aggregate allocator wall-clock seconds.

Fleet	Method	Completed	/M cand.	/comp. s
Low	M2M	3893.9	1.96	3.07
	M2M-wSKU	4168.5	2.96	4.01
	ASGM	4207.7	2.51	5.13
	SGM	5539.7	4.11	6.00
Medium	M2M	4614.2	1.54	2.28
	M2M-wSKU	4950.4	2.42	3.19
	ASGM	5039.4	2.09	4.31
	SGM	6615.1	3.22	4.76
High	M2M	5157.9	1.03	1.46
	M2M-wSKU	5591.7	1.62	2.10
	ASGM	6113.0	3.79	5.85
	SGM	7027.9	2.08	3.14

result is not explained by evaluating more endpoint pairs, and one-factor sweeps show that the improvement persists across a broad range of memory settings. The medium open and open-top transfer study further shows that SGM remains beneficial beyond the primary small-map benchmark.

Scope of the claim. The evidence establishes SGM as an effective congestion-control layer for many-to-many warehouse MAPD under shared rolling-horizon RHCR/PBS planning. These results support endpoint instantiation as a robust control interface for shaping congestion before planner goals are fixed. Future work includes nonstationary demand, richer kinematics, and physical robot deployments.

8 Conclusion

We introduced SGM, a bounded graph-memory layer for many-to-many MAPD that uses recent execution to guide endpoint instantiation and route preferences. Across paired benchmarks, SGM improves completed-task throughput in every evaluated map–fleet condition relative to both reconstructed many-to-many baselines. The endpoint-only ablation shows that memory-guided endpoint instantiation is the primary source of this gain.

The complete SGM controller retains this throughput while reducing waiting, blocked motion, blocked-motion replanning, and planner runtime. These results show that many-to-many MAPD offers a high-leverage congestion-control interface before path planning begins: the controller can shape which goals enter the planner, not only how agents travel to goals already selected. Future work includes extending SGM to nonstationary warehouse environments and physical robot deployments.

References

- Ahuja, R. K.; Ergun, Ö.; Orlin, J. B.; and Punnen, A. P. 2002. A Survey of Very Large-Scale Neighborhood Search Techniques. *Discrete Applied Mathematics*, 123: 75–102.
- Bonabeau, E.; Dorigo, M.; and Theraulaz, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.
- Boysen, N.; de Koster, R.; and Weidinger, F. 2019. Warehousing in the E-Commerce Era: A Survey. *European Journal of Operational Research*, 277: 396–411.
- Chen, Z.; Alonso-Mora, J.; Bai, X.; Harabor, D. D.; and Stuckey, P. J. 2021. Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery. *IEEE Robotics and Automation Letters*, 6(3): 5816–5823.
- Chen, Z.; Harabor, D.; Li, J.; and Stuckey, P. J. 2024. Traffic Flow Optimisation for Lifelong Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20674–20682.
- Dorigo, M.; and Gambardella, L. M. 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1): 53–66.
- Grassé, P.-P. 1959. La Reconstruction du Nid et les Coordinations Interindividuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. La Théorie de la Stigmergie: Essai d'Interprétation du Comportement des Termites Constructeurs. *Insectes Sociaux*, 6: 41–80.
- Holm, S. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6: 65–70.
- Hönig, W.; Kumar, T. K. S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Multi-Agent Path Finding with Kinematic Constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 26, 477–485.
- Korsah, G. A.; Stentz, A.; and Dias, M. B. 2013. A Comprehensive Taxonomy for Multi-Robot Task Allocation. *The International Journal of Robotics Research*, 32: 1495–1512.
- Lau, T. T.-K.; and Sengupta, B. 2022. The Multi-Agent Pickup and Delivery Problem: MAPF, MARL and Its Warehouse Applications. arXiv:2203.07092.
- Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. K. S.; and Koenig, S. 2021. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11272–11281.
- Li, M.-F.; and Sun, M. 2023. The Study of Highway for Lifelong Multi-Agent Path Finding. arXiv:2304.04217.
- Liu, M.; Ma, H.; Li, J.; and Koenig, S. 2019. Task and Path Planning for Multi-Agent Pickup and Delivery. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 1152–1160.
- Ma, H.; Harabor, D.; Stuckey, P. J.; Li, J.; and Koenig, S. 2019a. Searching with Consistent Prioritization for Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7643–7650.
- Ma, H.; Hönig, W.; Kumar, T. K. S.; Ayanian, N.; and Koenig, S. 2019b. Lifelong Path Planning with Kinematic Constraints for Multi-Agent Pickup and Delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7651–7658.
- Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 837–845.
- Makino, H.; and Ito, S. 2024. Online Multi-Agent Pickup and Delivery with Task Deadlines. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8428–8434.
- Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority Inheritance with Backtracking for Iterative Multi-Agent Path Finding. *Artificial Intelligence*, 310: 103752.
- Phillips, M.; and Likhachev, M. 2011. SIPP: Safe Interval Path Planning for Dynamic Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 5628–5635.
- Pisinger, D.; and Røpke, S. 2010. Large Neighborhood Search. In Gendreau, M.; and Potvin, J.-Y., eds., *Handbook of Metaheuristics*, 399–420. Springer.
- Schneider, E.; Chen, J.; Gu, T.; Lian, K.; Hutchinson, S.; and Chernova, S. 2026. Many-to-Many Multi-Agent Pickup and Delivery. arXiv:2605.07835.
- Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 219: 40–66.
- Silver, D. 2005. Cooperative Pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, volume 1, 117–122.
- Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Barták, R.; and Boyarski, E. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, volume 10, 151–158.
- Surynek, P. 2010. An Optimization Variant of Multi-Robot Path Planning Is Intractable. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 1261–1263.
- Theraulaz, G.; and Bonabeau, E. 1999. A Brief History of Stigmergy. *Artificial Life*, 5(2): 97–116.
- Wilcoxon, F. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1: 80–83.
- Wu, X.; Liu, Y.; Tang, X.; Cai, W.; Bai, F.; Khonstantine, G.; and Zhao, G. 2021. Multi-Agent Pickup and Delivery with Task Deadlines. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, volume 12, 206–208.
- Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine*, 29: 9.

- Xu, Q.; Li, J.; Koenig, S.; and Ma, H. 2022. Multi-Goal Multi-Agent Pickup and Delivery. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9964–9971.
- Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, 1443–1449.
- Zang, H.; Zhang, Y.; Jiang, H.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Li, J. 2025. Online Guidance Graph Optimization for Lifelong Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 14726–14735.
- Zhang, Y.; Bhatt, V.; Fontaine, M. C.; Nikolaidis, S.; and Li, J. 2026. Optimization of Edge Directions and Weights for Mixed Guidance Graphs in Lifelong Multi-Agent Path Finding. arXiv:2602.23468.
- Zhang, Y.; Jiang, H.; Bhatt, V.; Nikolaidis, S.; and Li, J. 2024. Guidance Graph Optimization for Lifelong Multi-Agent Path Finding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 311–320.

Purpose. This technical supplement provides the complete experimental record supporting the paper: controller definitions, memory configuration, feasibility preservation, full benchmark outcomes, ablations, robustness controls, transfer evidence, and spatial diagnostics.

S1 Reproducibility Protocol and Statistical Analysis

The primary benchmark comprises $5 \times 3 \times 25 \times 4 = 1500$ method–layout–fleet–seed runs. The five layouts are restricted, open-top, open, maze A, and maze B. Low, medium, and high fleet levels use 28, 56, and 84 agents, respectively; these are 30.1%, 60.2%, and 90.3% of the 93 designated parking cells. Every run lasts $H = 3600$ timesteps. Within each layout–fleet–seed cell, every method receives the same map, initial state, and replayed request stream.

The primary outcome is the cumulative number of completed tasks, $C(H)$; final throughput is $C(H)/H$. We also record time-resolved completions, final-window activity, blocked moves, waiting events, planner time, allocation time, service time, and candidate evaluations. All main comparisons use the same RHCR/PBS backend with a planning window of 512. The allocator sequence depth is three; the 30-second allocator and 10-second planner limits are safety budgets.

Computing environment. Experiments ran on a cluster node that has two Intel Xeon Gold 6248 processors (48 physical cores and 96 hardware threads total). The experiment campaign used 88 CPU threads and a 1 TB memory allocation; the simulation and planner are CPU-bound, so no GPU was used. Recorded runs used Python 3.10.12 on Linux 5.15.0-179-generic (glibc 2.35), with GCC 11.4.0 for the native planner extension. The Python environment pins PyYAML 6.0.3, pybind11 3.0.4, NumPy 2.4.6, SciPy 1.17.1, pandas 3.0.3, Matplotlib 3.11.0, and statsmodels 0.14.6.

For each layout–fleet condition, we compare seed-matched final completed-task counts using two-sided Wilcoxon signed-rank tests. Holm–Bonferroni correction controls the family-wise error rate across the 30 planned comparisons of SGM with M2M and M2M-wSKU. Table S1 gives the complete test record. All adjusted p -values are below 6.14×10^{-5} .

Paired inference. Pairing makes the seed-level throughput difference, rather than an unpaired aggregate, the unit of inference. Each comparison therefore holds the layout, fleet size, initial configuration, and entire request sequence fixed while varying only the controller. The reported paired percentage gains summarize effect magnitude, and the signed-rank tests establish that the observed advantages are consistently positive across the replayed streams.

Table S1: Two-sided Wilcoxon signed-rank tests for final completed tasks ($n = 25$ paired seeds per cell). Each comparison is SGM versus the named baseline; p_{Holm} is adjusted across all 30 planned tests.

Layout	Fleet	W_{M2M}	p_{Holm}	W_{wSKU}	p_{Holm}
Restricted	Low	0	1.79×10^{-6}	0	1.79×10^{-6}
Restricted	Medium	0	1.79×10^{-6}	0	1.79×10^{-6}
Restricted	High	1	1.79×10^{-6}	0	1.79×10^{-6}
Open-top	Low	0	1.79×10^{-6}	0	1.79×10^{-6}
Open-top	Medium	15	4.90×10^{-5}	0	1.79×10^{-6}
Open-top	High	17	6.14×10^{-5}	0	1.79×10^{-6}
Open	Low	0	1.79×10^{-6}	0	1.79×10^{-6}
Open	Medium	1	1.79×10^{-6}	0	6.14×10^{-5}
Open	High	0	1.79×10^{-6}	2	6.14×10^{-5}
Maze A	Low	0	6.14×10^{-5}	0	6.14×10^{-5}
Maze A	Medium	0	1.79×10^{-6}	0	1.79×10^{-6}
Maze A	High	0	1.79×10^{-6}	0	1.79×10^{-6}
Maze B	Low	0	1.79×10^{-6}	0	1.79×10^{-6}
Maze B	Medium	0	1.79×10^{-6}	0	1.79×10^{-6}
Maze B	High	0	1.79×10^{-6}	3	2.09×10^{-6}

S2 Controllers and Common Evaluation Harness

M2M and M2M-wSKU are released-code-faithful reconstructions of the allocation objectives in Schneider et al. (2026), integrated into the common simulator, request-replay protocol, RHCR/PBS backend, and validation layer. This controlled integration isolates the allocation decision under one planner boundary. M2M scores candidates from travel-distance estimates. M2M-wSKU adds the released inventory-distribution term, with base travel weight $w_b = 1.0$ and SKU weight $w_s = 0.25$. Both reconstructions rebuild queued tasks at each allocation cycle.

SGM is the primary controller. It continuously applies path-memory endpoint scoring, supplies bounded memory-derived route costs, and preserves queued work across event-driven repairs. ASGM uses the same memory representation but activates memory-guided scoring and routing only under sustained congestion evidence; otherwise it uses the M2M-wSKU-style objective and unit route costs. Table S2 summarizes the four main methods.

Table S2: Controller implementation details under the common evaluation harness. All methods use the same simulator, replayed requests, RHCR/PBS backend, and feasibility validation.

Method	Endpoint objective	Route costs	Memory activation	Queue update
M2M	Travel distance	Unit	None	Rebuild
M2M-wSKU	$w_b d + w_s I$; $w_b = 1.0, w_s = 0.25$	Unit	None	Rebuild
ASGM	M2M-wSKU or memory-scored	Unit or bounded memory-derived	Sustained congestion	Preserve
SGM	Path-memory scored	Bounded memory-derived	Continuous	Preserve

S3 Memory Configuration and Validity Boundary

S3.1 Execution traces and deposits

SGM maintains typed, decaying signals rather than a single congestion value. Node channels record waiting, endpoint pressure, and completion. Directed-edge channels record traversal, delay, blocking, and directional flow. When a request is released, one unit of source or destination pressure is distributed uniformly over its feasible endpoints. Unplanned waiting deposits waiting and delay; planned waiting on a

valid route does not. Before execution, proposed moves are checked for vertex conflicts and edge swaps. A move that would violate either constraint is held and deposits a *pre-execution blocking* signal; it is not an executed collision.

For channel c and graph element x , memory evolves as

$$M_c^{t+1}(x) = \rho_c M_c^t(x) + \Delta_c^t(x). \quad (S1)$$

Waiting and delay use $\rho_c = 0.85$; blocking and endpoint pressure use 0.90; congestion, traversal, and flow use 0.92; and completion uses 0.95. A corridor modifier can increase retention by 0.05 on degree-at-most-two corridors, capped at 0.99.

Table S3: Active memory channels in the reported configuration.

Channel	Element	Deposit event	Retention
Traversal / successful flow	Directed edge	Executed traversal	.92
Waiting	Node	Unplanned wait	.85
Delay	Directed edge	Unplanned wait	.85
Blocking	Node / directed edge	Blocked proposed move	.90
Endpoint pressure	Node	Request release	.90
Completion	Endpoint node	Completed delivery	.95

S3.2 Endpoint and routing interfaces

For a feasible candidate (a_i, τ, s, d) , SGM augments the M2M-wSKU-style baseline score with sampled path memory:

$$C(a_i, \tau, s, d) = C_0(a_i, \tau, s, d) + 1.75 P_{\text{path}}(a_i, s, d). \quad (S2)$$

The path term sums current directed penalties along sampled unit-cost shortest paths from a_i to s and from s to d . Source and destination shortlists are capped at 64 locations each, with at most 32 endpoint pairs per request. These sampled paths are scoring proxies: RHCR/PBS computes the final joint plan.

For a legal directed move (u, v) , routing uses

$$\tilde{c}_t(u, v) = 1 + \min\{0.1, 0.2 R_t(u, v)\}, \quad (S3)$$

where R_t aggregates normalized congestion, blocking, delay, contraflow, successful forward flow, and discounted planned occupancy over 32 timesteps. Thus every legal move has positive, bounded cost: $1 \leq \tilde{c}_t(u, v) \leq 1.1$.

Table S4: Remaining reported controller settings.

Setting	Value
Planning horizon	3600 timesteps
Paired seeds	25 (10–34)
RHCR/PBS window	512
Allocation sequence depth	3
Path-memory weight	1.75
Routing-memory weight / cap	0.2 / 0.1
Guidance horizon	32
SGM source / destination / pair caps	64 / 64 / 32
M2M source / destination / pair caps	100 / 100 / 128
ASGM fallback pair cap	160

S3.3 Feasibility preservation

Proposition. For every endpoint assignment selected by SGM, the RHCR/PBS backend receives the original graph, agent states, task-feasibility constraints, and vertex- and edge-conflict predicates. Every accepted and executed plan is therefore collision-free and task-feasible under the original MAPD constraints.

Justification. SGM changes only scalar scores used to rank a finite set of feasible candidates and scalar costs assigned to legal moves. It does not add or remove graph edges, alter agent states, relax endpoint feasibility, or modify conflict predicates. RHCR/PBS alone accepts the joint plan. Positive bounded route costs and finite candidate ranking cannot create an infeasible assignment, a negative edge cost, a zero-cost cycle, or an infinite cost for a finite legal path.

S4 Layouts and Complete Benchmark Outcomes

Each layout is a 50×27 grid with 1350 cells and 93 designated storage/parking cells. Restricted, open-top, and open are the warehouse layouts used for the primary benchmark; maze A and maze B preserve the scale while changing corridor structure.

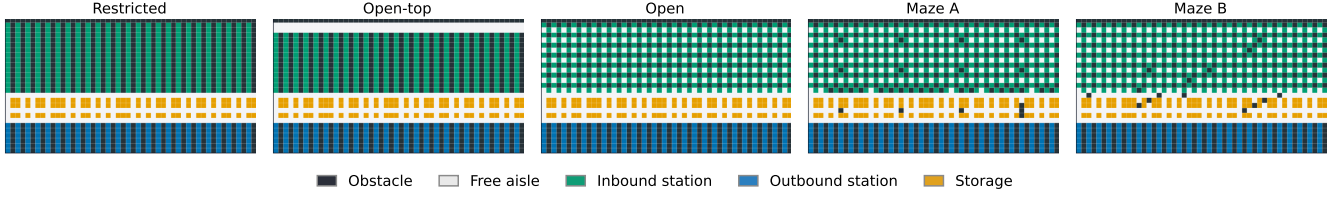


Figure S1: The five 50×27 primary-benchmark layouts at full scale. Dark cells are obstacles; light cells are traversable aisles; colored cells denote inbound stations, outbound stations, and storage/parking locations. Restricted, open-top, and open are the benchmark warehouse families; maze A and maze B hold scale fixed while changing corridor topology.

Table S5: Cell composition of the five layouts (counts out of 1350).

Layout	Obstacle	Aisle	Inbound	Outbound	Storage
Restricted	550	207	350	150	93
Open-top	500	307	300	150	93
Open	375	382	350	150	93
Maze A	408	349	350	150	93
Maze B	390	367	350	150	93

Table S6 gives all primary-benchmark terminal outcomes. The final column is the paired SGM percentage improvement over the stronger M2M or M2M-wSKU baseline in the corresponding cell.

Table S6: Complete primary benchmark: final completed tasks (mean over 25 paired seeds). Gain is the paired SGM improvement over the stronger M2M baseline, with a 95% confidence interval.

Layout	Fleet	M2M	M2M-wSKU	ASGM	SGM	Gain (%)
Restricted	Low	4275	4610	4648	6123	33.2 ± 5.5
Restricted	Medium	5063	5542	5628	7551	36.7 ± 6.9
Restricted	High	5746	6280	6955	7821	24.3 ± 4.8
Open-top	Low	4221	4313	4346	5770	34.6 ± 6.7
Open-top	Medium	5062	5016	5255	6878	34.9 ± 11.0
Open-top	High	5573	5420	6156	7243	28.8 ± 9.3
Open	Low	4126	4386	4423	5820	33.2 ± 4.6
Open	Medium	4920	5269	5347	6949	32.5 ± 5.4
Open	High	5595	6287	6512	7517	20.5 ± 5.8
Maze A	Low	2899	3258	3323	4425	36.1 ± 2.9
Maze A	Medium	3346	3866	3898	5149	33.4 ± 3.3
Maze A	High	3843	4367	4862	5791	32.7 ± 4.2
Maze B	Low	3949	4275	4299	5560	30.4 ± 4.4
Maze B	Medium	4681	5059	5069	6548	29.6 ± 5.3
Maze B	High	5033	5604	6080	6768	21.1 ± 4.8

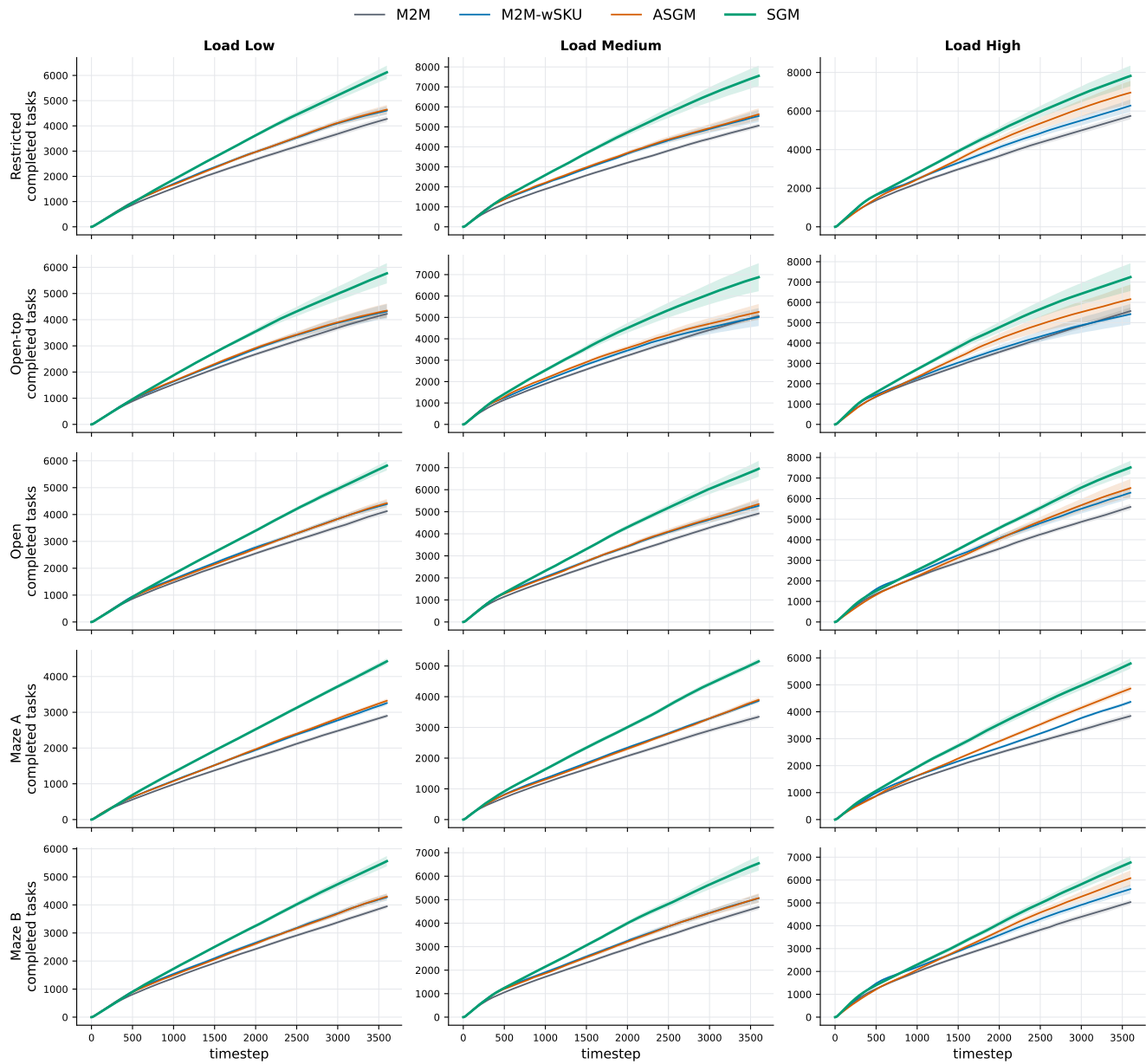


Figure S2: Cumulative completed tasks throughout the 3600-timestep horizon for every primary-benchmark layout and fleet level. Curves are means over 25 paired seeds; shaded bands show 95% confidence intervals. SGM separates from both M2M baselines early and maintains the advantage through the terminal horizon in all 15 conditions.

Temporal separation. The trajectories show that SGM’s advantage is not a terminal-window artifact. The methods begin near parity while agents first receive and instantiate requests, then the SGM curves separate as the shared system accumulates traffic. That separation is sustained through the 3600-timestep horizon in every layout–fleet condition, including the two maze topologies. The result is consistent with the proposed mechanism: endpoint selection changes the congestion pattern before repeated assignments commit traffic to the same constrained regions.

S5 Ablations and Robustness Controls

Endpoint-only retains path-memory endpoint steering with unit routing costs. Routing-only retains memory-weighted routing without endpoint-memory scoring. No-queue-preserve keeps both memory interfaces but rebuilds queued work at every allocation cycle. The ablation cleanly identifies endpoint steering as the dominant throughput interface: endpoint-only retains 99.3–100.5% of full SGM throughput across fleet levels. Table S7 and Figure S3 report the same conclusion in tabular and visual form.

Table S7: Component ablations aggregated over five layouts and 25 seeds per fleet level. “% SGM” is relative to full SGM throughput; Δ wSKU is relative to M2M-wSKU.

Fleet	Method	Completed	% SGM	Δ wSKU
Low	SGM	5540	100.0	+32.9
	Endpoint-only	5556	100.3	+33.3
	Routing-only	4347	78.5	+4.3
	No queue preserve	5326	96.1	+27.8
Medium	SGM	6615	100.0	+33.6
	Endpoint-only	6570	99.3	+32.7
	Routing-only	4968	75.1	+0.4
	No queue preserve	6460	97.6	+30.5
High	SGM	7028	100.0	+25.7
	Endpoint-only	7060	100.5	+26.3
	Routing-only	5537	78.8	−1.0
	No queue preserve	6913	98.4	+23.6

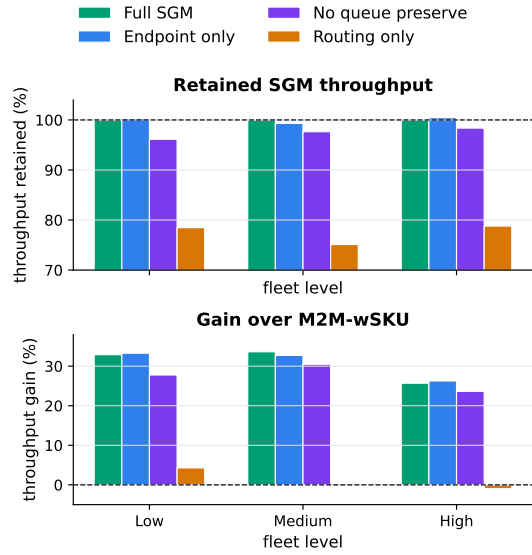


Figure S3: Component attribution aggregated over the five primary layouts and 25 paired seeds per fleet level. Endpoint-only preserves full SGM throughput, whereas routing-only does not; routing and queue preservation contribute the operational efficiency retained by the full controller.

Operational value of route guidance. Endpoint-only isolates the throughput effect of endpoint steering. Full SGM retains that throughput while making its execution less costly: it lowers planner time in every small-map condition and reduces waiting, blocked moves, and blocked-motion replans in most conditions (Table S8).

Table S8: Full SGM versus endpoint-only, averaged over the 15 small-map layout–fleet conditions. Lower is better except for completed tasks.

Metric	Endpoint-only	SGM	Change	SGM favored
Final completed tasks	6395.2	6394.3	−0.9	–
Planner time (s)	295.6	268.8	−9.1%	15 / 15
Waiting events / step	27.24	27.03	−0.8%	13 / 15
Blocked moves	19.1	16.9	−11.5%	12 / 15
Blocked-motion replans	8.84	7.90	−10.6%	12 / 15

Routing controls hold M2M-wSKU endpoint allocation fixed and vary only the routing costs. They show that post-commitment route guidance alone does not produce the main effect. Sensitivity controls vary one setting at a time; matched-cap controls reverse the default candidate caps. Table S9 gives the complete control record on restricted, open-top, and open layouts at medium and high fleet levels (five paired seeds per cell).

Table S9: Complete routing, sensitivity, and candidate-cap controls. Each row pools 30 matched layout–fleet–seed runs. Δ is the paired percentage change relative to M2M-wSKU.

Control	Setting	Completed tasks	Δ (%)
<i>Routing controls: M2M-wSKU endpoint allocation is held fixed.</i>			
M2M-wSKU	Unit routing costs	5749.6 ± 326.3	0.0 ± 0.0
M2M-wSKU	Recent-traffic routing	5737.3 ± 328.6	-0.1 ± 2.1
M2M-wSKU	Static highway routing	5661.5 ± 338.3	-1.4 ± 2.5
SGM	Endpoint and route memory	7391.1 ± 484.3	$+29.2 \pm 5.7$
<i>One-factor sensitivity: full SGM with one setting varied.</i>			
Path-memory weight	.875	7427.7 ± 496.9	$+29.7 \pm 5.6$
	1.75 (reported)	7391.1 ± 484.3	$+29.2 \pm 5.7$
	3.5	7384.5 ± 499.6	$+29.1 \pm 6.1$
Memory retention	.90	7400.4 ± 505.4	$+29.2 \pm 5.9$
	.95	7381.2 ± 496.0	$+28.9 \pm 5.6$
	.98	7351.0 ± 492.0	$+28.5 \pm 5.8$
Endpoint-pair cap	16	6603.4 ± 360.7	$+16.0 \pm 5.2$
	32 (reported)	7391.1 ± 484.3	$+29.2 \pm 5.7$
	64	8084.8 ± 671.7	$+40.5 \pm 7.5$
Route-memory weight	0.0	7582.8 ± 507.9	$+32.5 \pm 5.9$
	0.2 (reported)	7391.1 ± 484.3	$+29.2 \pm 5.7$
	0.4	7411.4 ± 501.0	$+29.4 \pm 5.7$
<i>Matched candidate caps: source / destination / pair caps are exchanged.</i>			
M2M-wSKU	64/64/32	5687.7 ± 321.7	-0.9 ± 1.9
SGM	100/100/128	8351.8 ± 754.6	$+44.8 \pm 8.5$

S6 Transfer and Spatial Evidence

The medium-scale transfer study uses open and open-top layouts at the high fleet level, five paired seeds per layout, and the same 3600-timestep horizon. SGM improves final completed tasks over M2M-wSKU by $7.3 \pm 3.3\%$. The transfer result confirms that the endpoint-memory interface remains effective beyond the primary small layouts.

Table S10: Medium-scale high-fleet transfer results on open and open-top layouts (10 paired runs: two layouts by five seeds). Δ is relative to M2M-wSKU.

Method	n	Completed tasks	Δ (%)	Range (%)
M2M-wSKU	10	25764.6 ± 1204.2	0.0 ± 0.0	0.0 to 0.0
Recent routing	10	25490.4 ± 1310.0	-1.1 ± 0.7	-3.0 to 1.4
Static highway	10	24828.6 ± 1233.8	-3.6 ± 1.5	-8.6 to -0.6
Endpoint-only	10	23150.9 ± 5040.5	-8.6 ± 21.0	-77.4 to 17.9
SGM	10	27576.0 ± 795.9	$+7.3 \pm 3.3$	$+1.4$ to 14.8

The transfer comparison makes the role of the full controller explicit. Endpoint-only is competitive on the primary small maps, but it does not retain the full controller’s performance on these larger layouts. The routing-only controls remain near or below M2M-wSKU in both benchmark regimes, whereas full SGM remains positive under the changed scale and geometry.

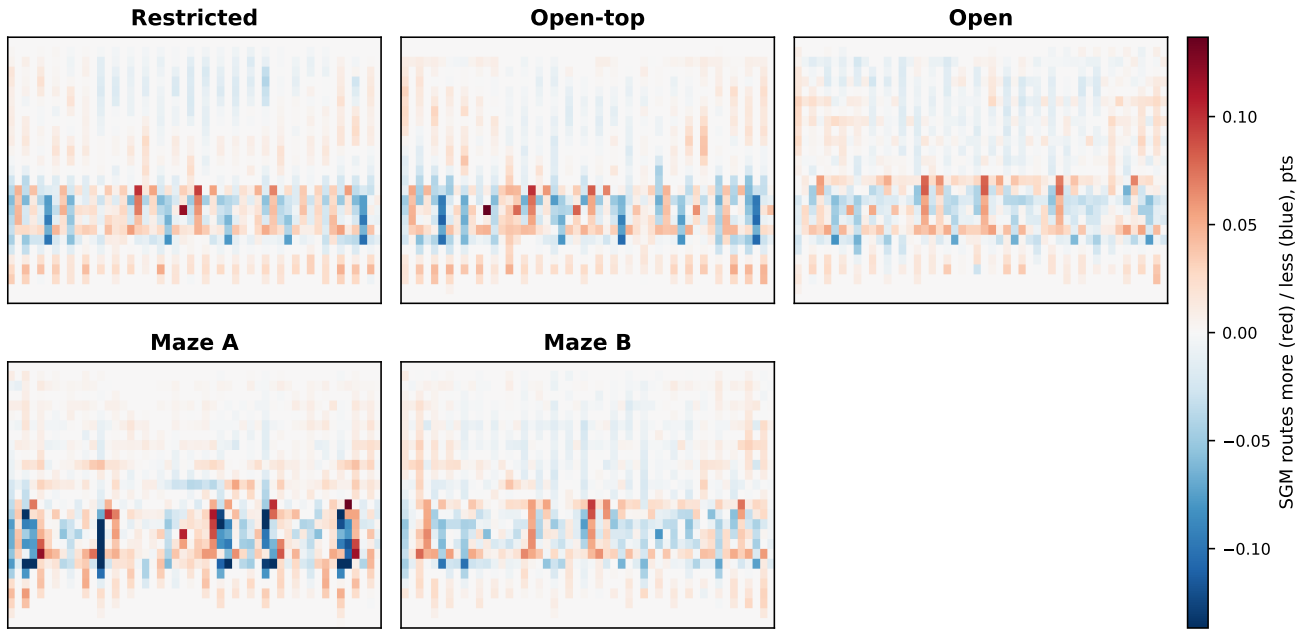


Figure S4: Change in final-window traversal share at high fleet level: SGM minus M2M-wSKU. Red cells receive a larger share of traversals under SGM; blue cells receive a smaller share. Each panel is normalized by its method's total traversals, so the diagnostic measures redistribution rather than the higher absolute task volume produced by SGM.

S6.1 Spatial traffic redistribution

Figure S4 complements terminal counts with a traffic-distribution diagnostic at high fleet level. It visualizes the change in each cell's share of traversals relative to M2M-wSKU during the final measurement window. The redistribution is topology-specific: SGM redirects traffic away from the locally overloaded regions induced by each layout.

The diagnostic does not require any particular corridor to receive more or less traffic. Instead, it shows that the redistribution follows each layout's local geometry: under SGM, relative traversal decreases in some congested service regions and increases on alternative accessible routes. Together with the transfer table, this evidence supports a controller that responds to the available warehouse structure rather than a fixed global traffic pattern.