systems

MDPI

Article

Deliberate Assignment Deferral for Multi-Agent Pickup and Delivery with Deadlines †

Taisei Hirayama [1,*], Kohei Yoshida [2], Hiroki Sakaji [1] and Itsuki Noda [1]

[1] Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0814, Japan; sakaji@ist.hokudai.ac.jp (H.S.); i.noda@ist.hokudai.ac.jp (I.N.)
[2] Toyota Industries Corporation, Kariya 474-8601, Japan
* Correspondence: hirayama.h77@gmail.com
† This paper is an extended and substantially revised version of the conference paper: Taisei Hirayama, Kohei Yoshida, Hiroki Sakaji, and Itsuki Noda. "Efficient Assignment of Immediate Tasks Using Deep Reinforcement Learning in Multi-Agent Pickup and Delivery", PAAMS 2025.

Abstract

Automated warehouses must coordinate fleets of mobile robots online while meeting order deadlines. In online Multi-Agent Pickup and Delivery with Deadlines (MAPD-D), committing to a feasible task immediately may restrict flexibility and increase downstream tardiness through congestion and reservation interactions. We propose Deliberate Assignment Deferral (DAD-θ), a one-parameter gate on top of deadline-aware Token Passing baselines (D-TP and D-TPTS). At each token turn, the token holder evaluates the baseline-defined assignable tasks using the baseline score (lower is better); it commits only if the best score is at most θ, and otherwise follows the baseline fallback. A safety override forces assignment once any assignable task reaches non-positive pickup slack. We also introduce a scale-normalized score that makes θ dimensionless for transfer across maps and deadline scales. In 100-seed paired simulations across four arrival/deadline regimes on a benchmark map, scenario-calibrated DAD-θ reduces cumulative tardiness by 9–58% and increases on-time completion by 1.3–10.0 percentage points relative to always-assign. We discuss how θ can be calibrated offline in a digital twin and monitored online via deferral and safety-trigger rates for service-level control.

Keywords: multi-agent path finding; pickup and delivery; deadlines; token passing; online task allocation; tardiness minimization; digital twin

Check for updates

Academic Editor: Emilio Larrodé

Received: 20 March 2026
Revised: 16 April 2026
Accepted: 24 April 2026
Published: 1 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Automated warehouses and fulfillment centers are cyber-physical logistics systems that must coordinate fleets of mobile robots in real time to meet service-level targets (e.g., order deadlines) at scale [1]. Because agents share space and resources, local task-assignment decisions can interact through congestion and reservation constraints, producing system-level effects that are not obvious from any single agent's perspective. Beyond travel efficiency, operators care about deadline compliance and tardiness because they directly affect order-level quality of service and downstream operations. This creates a tension between computationally heavy global optimization and lightweight online dispatch rules that can be deployed robustly under uncertainty.

In practical warehouse stacks, this tension appears across multiple layers. Upstream systems such as WMS/WES release orders and deadlines, while the fleet controller must repeatedly make fast local commitments under changing congestion, reservation conflicts,

and endpoint scarcity. Re-optimizing the whole system after every event is often too expensive or too brittle for online use, yet purely immediate assignment can be myopic: it can reserve long paths or consume valuable endpoints just before a more urgent task becomes available. This motivates *commitment timing* as a systems question in its own right, rather than treating assignment as an always-commit operation whenever a feasible task exists.

Multi-agent path finding (MAPF) and its pickup-and-delivery extensions are central problems in warehouse and logistics automation [1–5]. In online Multi-Agent Pickup and Delivery (MAPD), tasks arrive over time, and agents repeatedly receive new tasks after finishing current ones [4–6]. In many practical systems, each task also has a deadline; then minimizing tardiness becomes more relevant than only minimizing travel cost.

From a systems perspective, task allocation mediates *commitment under uncertainty*: committing too early can lock the system into a suboptimal trajectory, while delaying commitment can preserve optionality (the "value of postponement") at the risk of starvation if overused. This commitment–flexibility trade-off appears across operations and service systems, including postponement in supply chains and real-options views of delaying irreversible commitments [7,8].

Recent deadline-aware Token Passing methods for online MAPD-D, namely D-TP and D-TPTS, already provide lightweight and deployable online coordination by combining token-mediated reservations with a deadline-aware assignment score based on pickup slack and distance-to-pickup [9]. In such systems, however, assigning a task is not merely a local choice: it is an operational commitment that interacts with future token reservations, endpoint availability, and the set of tasks that other agents will be able to take next. This suggests a practical opportunity: when the currently best assignable task is not sufficiently attractive, deliberately postponing commitment by one token turn may preserve short-horizon flexibility and improve downstream deadline performance without replacing the underlying TP-family planner.

Intuitively, immediate assignment can lock an agent into a low-quality commitment (e.g., a far pickup with ample slack) at a moment when the system state is still evolving. If a more urgent or better-matched task becomes assignable shortly afterwards, the earlier commitment can force a late service by another agent or create longer detours due to already reserved paths and endpoint-occupancy constraints (the latter reflected in the assignable-set criterion in Equation (5)). A one-turn deferral gate aims to preserve short-horizon flexibility in these situations while retaining the simplicity of Token Passing.

Delaying irreversible commitments to preserve future flexibility has been explored in dynamic routing and related online service settings [10,11]. Bringing the same idea into TP-family MAPD-D is less straightforward than it may appear. In Token Passing, committing now or waiting one turn affects shared reservations, endpoint occupancy, and the future assignable set, while excessive waiting can induce starvation near deadlines. As a result, a useful TP-compatible deferral mechanism must satisfy several practical constraints simultaneously: it should preserve the baseline token semantics, add essentially no extra planning burden, and include an explicit guardrail against over-deferral. These requirements motivate a minimal threshold gate with normalization for portability and a safety override for deadline-critical cases.

This paper introduces Deliberate Assignment Deferral (DAD-θ), a minimal extension of deadline-aware Token Passing that makes *commitment timing* an explicit control variable in online MAPD-D. Rather than redefining task scores or path planning, DAD-θ acts only on the *decision of whether to commit now*: the token holder commits only when the best currently assignable option is sufficiently attractive under a threshold θ, and otherwise deliberately waits for one token turn to preserve short-horizon flexibility. This design is operationally

appealing because it leaves downstream path planning and low-level robot execution unchanged while exposing a single, explainable, operator-tunable control knob. In this sense, DAD-θ is not only a heuristic modification of TP-family allocation, but also a lightweight service-level control mechanism for cyber-physical warehouse coordination stacks.

Viewed more narrowly, DAD-θ turns commitment timing into a systems parameter that can be tuned against service-level targets while preserving TP-family semantics. This paper, therefore, asks a scoped question: within deadline-aware TP-family stacks, when does delaying commitment by one token turn improve deadline-oriented performance?

Our main contributions are:

- **Commitment timing as a control dimension:** we formalize *when to commit* as an explicit control variable in deadline-aware TP-family MAPD-D, and introduce DAD-θ as a minimal gate on top of D-TP and D-TPTS that preserves short-horizon flexibility without changing the underlying path planner or low-level control stack.
- **Deployable and portable design:** the proposed formulation retains the baseline candidate construction and score ordering, adds only a single threshold comparison and a parameter-free safety override, and supports scale-normalized thresholding so that θ is dimensionless and easier to transfer across maps and deadline regimes.
- **Strong empirical and operational evidence:** under a strict paired protocol with repeated seed-split holdout validation, ablations, transfer experiments, and larger-map generalization, we show robust reductions in cumulative tardiness, frequent improvements in on-time completion, often concurrent makespan gains, and negligible runtime impact, while also demonstrating how θ can be calibrated offline in a digital twin and monitored online via simple telemetry.

Figure 1 highlights the intended *systems role* of DAD-θ. Rather than replacing the warehouse coordination stack, DAD-θ acts as a lightweight commitment-control layer between order/deadline inputs and TP-family task allocation. This makes it possible to tune service-level behavior through a single operator-facing parameter, while leaving downstream path planning and robot execution unchanged. The same abstraction also supports an operational loop of offline calibration in a digital twin and online monitoring through low-cost telemetry, such as deferral and safety-trigger rates.

Note on Prior Publication.

This manuscript is an extended and substantially revised journal version of our PAAMS 2025 conference paper "Efficient Assignment of Immediate Tasks Using Deep Reinforcement Learning in Multi-Agent Pickup and Delivery" (ProcrastiNet) [12], but it differs from that paper in problem setting, method, objective, and empirical scope. The conference version studied MAPD with immediate tasks and used a learning-based policy to decide when to defer the assignment. In contrast, the present manuscript targets deadline-aware MAPD-D, replaces the learning stage with a deterministic TP-compatible threshold gate (DAD-θ), introduces scale normalization and a safety override, and evaluates the method under a substantially expanded experimental program centered on cumulative tardiness and deadline adherence. Thus, the present manuscript is not a journalized reproduction of the conference paper, but a different technical formulation and evaluation program built on the same high-level flexibility-preservation insight.

Compared with the conference version, this journal manuscript makes four substantive changes: (i) it changes the problem setting from immediate-task MAPD to deadline-aware MAPD-D; (ii) it changes the decision mechanism from a learned deferral policy to a deterministic and explainable TP-compatible threshold gate with normalization and a safety override; (iii) it changes the evaluation focus from general efficiency to deadline-oriented service-level outcomes such as cumulative tardiness, on-time completion, and tail

tardiness; and (iv) it substantially expands the empirical study with a strict paired protocol, seed-split holdout validation, comparator policies, ablations, transfer experiments, and larger-map generalization.

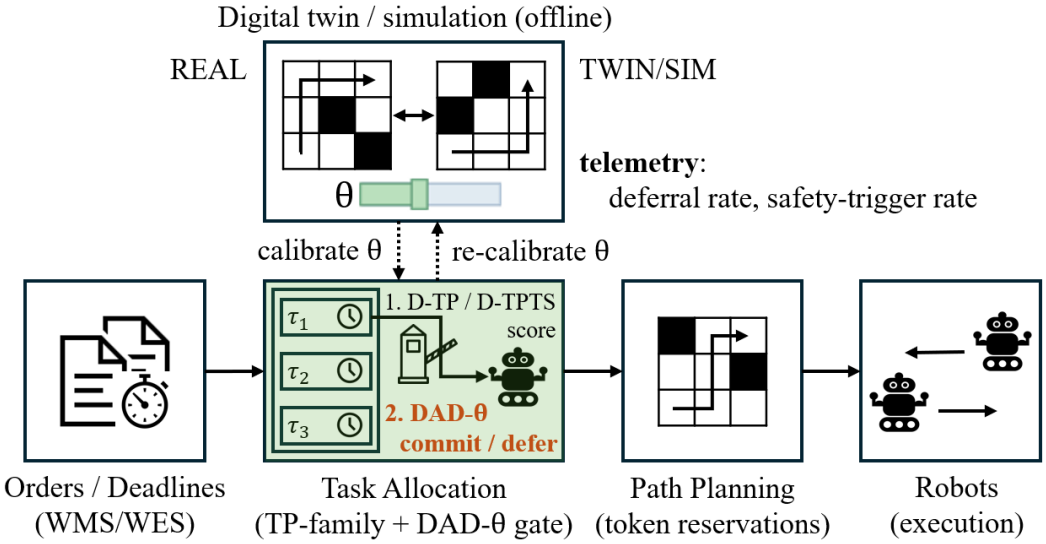


Figure 1. System-level overview of DAD-θ in a warehouse coordination stack. The figure highlights the functional role of DAD-θ as a lightweight *commit-versus-defer* gate placed between order/deadline inputs and downstream TP-family task allocation. DAD-θ leaves path planning and low-level robot control unchanged, while enabling offline calibration in a digital twin and online monitoring through telemetry, such as deferral and safety-trigger rates. Arrows indicate information/control flow, and colored blocks distinguish functional layers in the warehouse coordination stack. WMS/WES denote Warehouse Management System/Warehouse Execution System.

Paper Organization.

Section 2 reviews related work. Section 3 formulates the problem. Section 4 presents DAD-θ as a minimal gate on top of D-TP and D-TPTS, including normalized scoring and a safety override. Section 5 provides an interpretive one-step analysis that helps explain why intermediate thresholds can help. Section 6 describes the experimental setup and metrics. Section 7 reports results, including θ sweeps, holdout validation, and mechanism diagnostics. Section 8 discusses systems-level implications, limitations, and deployment considerations. Section 9 concludes.

2. Related Work

2.1. MAPF and MAPD

MAPF coordinates multiple agents to avoid vertex and edge-swap collisions while reaching goals [2]. MAPD generalizes this to lifelong online pickup-and-delivery tasks [4–6]. Token Passing (TP) and Token Passing with Task Swaps (TPTS) are standard online MAPD baselines for well-formed instances (see Section 3) [4]. Large-scale warehouse studies further validate MAPD relevance in production-style environments [13]. On the planning side, recent MAPF work has improved scalability with search-based solvers (e.g., LaCAM*) and explored throughput improvements for lifelong settings (e.g., guidance-graph optimization) [14–16]. Recent reviews also synthesize both beyond-classical MAPF generalizations and learning-enhanced MAPF pipelines, highlighting the growing range from search-based coordination to ML-assisted policies [17,18]. Recent MAPD variants have also moved closer to deployment constraints, for example, by considering dynamic environments with external moving agents [19,20] and capacity-enhanced lifelong pickup-and-delivery settings [21]. Other MAPD approaches integrate task assignment and collision-free planning

more tightly [22]; we focus on TP-family methods for well-formed online MAPD and their deadline-aware variants because DAD-θ is designed as a minimal extension of this family.

2.2. Deadlines in Multi-Agent Routing

Deadline-aware formulations have been studied in MAPF and MAPD-related settings, including MAPF with deadlines, multi-robot path planning with due times, and deadline-aware tour planning [23–25]. For MAPD, deadline-focused studies include offline/semi-offline settings and tardiness-oriented assignment approaches [26,27]. Online MAPD-D with Token Passing is addressed by D-TP and D-TPTS [9].

Related operations-research lines such as dynamic pickup-and-delivery/vehicle-routing with time constraints also motivate tardiness-sensitive decision making under online arrivals [28–30].

2.3. Deferral and Procrastination Strategies

Deferral-based policies have been investigated in neighboring online MAPD variants. In MAPD with immediate tasks, the ProcrastiNet study shows that deliberate postponement can improve system-level efficiency under uncertainty [12]. Our setting differs in objective and mechanism: we target MAPD-D tardiness and use a deterministic threshold gate on an existing deadline-aware score (no learning stage). In classical dynamic routing, deferral takes the form of *waiting strategies*—deciding where or how long vehicles should linger to preserve flexibility for future requests [10,11]. These vehicle-routing settings differ from TP-family MAPD-D: DAD-θ defers the assignment commitment at token turns under token-mediated reservations and endpoint-occupancy constraints, and it adds a portable (normalized) threshold and a safety override to prevent deferral-driven starvation. On the learning side of MAPF, recent work studies decentralized policies with richer social or behavioral conditioning to break symmetric conflicts [31]. This line is relevant as a neighboring alternative, but it operates on different action/state abstractions from the TP-family commit-or-defer gate studied here.

2.4. Dispatching Rules in Scheduling Theory

The D-TP/D-TPTS score blends pickup slack and distance-to-pickup, which is conceptually related to due-date *dispatching rules* such as Apparent Tardiness Cost (ATC) and related slack-based indices [32,33]. DAD-θ adds a *commit-or-wait* option on top of this index (agents can wait at safe endpoints; see Section 3), with a safety override to prevent repeated postponement once tasks become urgent.

2.5. Digital Twins and Offline Calibration

Digital twins are widely used in cyber-physical and manufacturing systems to support what-if evaluation, parameter calibration, and monitoring by coupling physical operations with a virtual or simulation counterpart [34,35]. This aligns with calibrating θ offline by sweeping representative workloads in simulation and monitoring deferral/safety telemetry in deployment (e.g., deferral and safety-trigger rates) to detect workload drift and guide re-calibration.

Scope of Comparisons.

DAD-θ is designed as a minimal extension of the TP-family: it is defined on token turns, reuses TP's assignable-task criterion, and when it defers, it executes the TP-family fallback behavior (wait/park at endpoints). Therefore, we restrict experimental comparison to D-TP and D-TPTS under the same well-formed assumptions and simulator so as to isolate the effect of the deferral gate while keeping endpoint selection and reservation mechanics unchanged.

Our goal is not to claim universal superiority across MAPD architectures, but to isolate the effect of *commitment timing control* within the TP-family architecture class. This requires holding fixed the candidate-set semantics, endpoint-based feasibility constraints, token-mediated reservations, and fallback actions, because all of these directly shape the consequences of deferring versus committing. Comparing against methods built on substantially different planning architectures would confound the effect of DAD-style deferral with differences in feasibility definitions, reassignment semantics, and reservation models. Accordingly, the experimental question of this paper is deliberately scoped as follows: *within deadline-aware TP-family stacks, does explicit control of commitment timing improve deadline-oriented service-level performance?*

Alternative MAPD paradigms, such as integrated task assignment and path planning [22] or learning-based allocation [12], do not share the same token/reservation semantics or the same notion of an assignable set and fallback action; extending DAD-style deferral beyond TP therefore requires architecture-specific definitions of candidate sets, feasibility constraints, and the deferral action, which we leave for future work.

3. Problem Formulation

We consider an online MAPD-D instance on an undirected graph $G = (V, E)$ with agent set $A = \{a_1, \ldots, a_m\}$. Here, V is the set of vertices and E is the set of undirected edges. Time is discrete ($t \in \mathbb{Z}_{\geq 0}$). Each task τ_j is defined as follows:

$$\tau_j = (r_j, u_j, v_j, d_j^{\langle d \rangle}), \tag{1}$$

where r_j, $u_j \in V$, $v_j \in V$, and $d_j^{\langle d \rangle} \in \mathbb{Z}_{\geq 0}$ are the release time, pickup vertex, delivery vertex, and delivery deadline of task τ_j, respectively. We assume $u_j \neq v_j$ (tasks with pickup=delivery are excluded). (In our experiments, deadlines and offsets are integer-valued; the formulation extends to real-valued deadlines.) Let c_j be the completion time, i.e., the arrival time at v_j after pickup.

Modeling Assumptions.

We adopt the standard TP-family assumptions used in MAPD/MAPD-D benchmarks. Agents are homogeneous in kinematics, carrying capability, and task eligibility; tasks are homogeneous in service requirements except for their release times, pickup vertices, delivery vertices, and deadlines. Each agent carries at most one task at a time. Pickup and delivery themselves are modeled as instantaneous state transitions at arrival: if an agent reaches u_j at time t_p, the task moves from *assigned* to *in service* at that arrival, with no extra loading duration parameter; delivery completes upon arrival at v_j at time c_j. Consequently, urgency near pickup is represented through the reference pickup deadline $d_j^{\langle p \rangle}$ and the pickup slack $d_j^{\langle p \rangle} - t$ used at token-turn decision times.

Task States and Execution Model (Online, Collision-Free Paths).

Each task progresses through the states *unreleased*, *released*, *assigned*, *in service*, and *completed*. A task is unreleased for $t < r_j$. Once $t \geq r_j$ and no agent has committed to it, the task is released. After an agent commits to the task but before pickup, it is assigned. After pickup and before delivery, it is in service. After arrival at the delivery vertex at time c_j, it is completed.

At each time step, each agent either moves to an adjacent vertex or stays at the same vertex. We impose the standard MAPF collision constraints: no two agents occupy the same vertex at the same time, and no two agents traverse the same edge in opposite directions in the same time step. No task can be assigned before its release time r_j. In TP-family methods,

the set of tasks reconsidered at a token turn depends on the baseline allocator: D-TP uses released and currently unassigned tasks, whereas D-TPTS may also reconsider tasks that are already assigned but not yet picked up. We therefore use *assignable* for tasks in the baseline-defined candidate set that additionally satisfy the endpoint-occupancy condition introduced in Equation (5). Thus, a released task is not automatically assignable: at a given token turn, it becomes assignable to agent a_i only if it belongs to the baseline-defined candidate set for that turn *and* passes the endpoint-occupancy filter in Equation (5). This distinction matters in TP-family methods because task releases, endpoint occupancy, and token reservations evolve together over time. To complete τ_j, an agent must visit the pickup vertex u_j and then the delivery vertex v_j; following TP-family MAPD assumptions, each agent carries at most one task at a time. Collision-free paths that satisfy these constraints are generated by the underlying TP-family planner (via token-mediated reservations); this paper focuses on task-allocation decisions (which task to commit to at token turns) rather than on the path-planning algorithm itself.

Task tardiness is defined as:

$$\epsilon_j = \max\{0, c_j - d_j^{\langle d \rangle}\}. \tag{2}$$

We define cumulative tardiness and makespan as

$$T = \sum_j \epsilon_j. \tag{3}$$

$$M = \max_j c_j. \tag{4}$$

The online allocation objective studied in this paper is to minimize cumulative tardiness T. In the experiments, we report T as the main evaluation metric and makespan M as an additional completion-time metric. Makespan is not used as a tie-breaker or secondary objective inside the allocation rule.

TP-Family Endpoints and Well-Formedness.

TP-family MAPD planners designate a set of *endpoints* (safe resting locations) and require token paths to end at endpoints [4]. Endpoints include all pickup and delivery locations together with additional non-task endpoints (e.g., parking). Following prior TP-family work, we assume well-formed instances: a finite task set, enough non-task endpoints for parking (at least as many as agents), and endpoint-to-endpoint reachability without traversing other endpoints [4,36].

4. Method: DAD-θ on D-TP and D-TPTS

4.1. Overview

DAD-θ is designed as a minimal, explainable modification of the TP-family deadline-aware Token Passing (D-TP and D-TPTS). At each token turn, the underlying baseline first constructs the tasks it can currently consider for commitment and computes its score for each such task, where lower is better. DAD-θ leaves that candidate construction and scoring unchanged. It only adds one gate: after the best baseline score is identified, the token holder commits immediately only if that best score is at most θ; otherwise, it defers assignment for that token turn and follows the baseline fallback behavior. A parameter-free safety override disables deferral when any assignable task reaches non-positive pickup slack. Figure 2 summarizes the resulting token-turn logic.

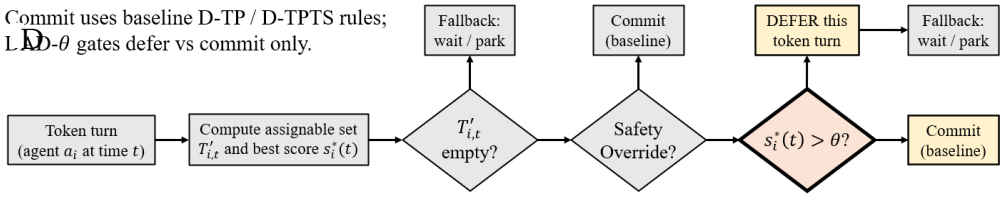


Figure 2. Flowchart summary of DAD-θ for one token turn. A token turn is triggered in the event-driven TP process when an agent reaches the end of its current token path and obtains the token for replanning; simultaneous requesters are then served sequentially by the implementation. For compactness, we write $s_i^*(t)$ for the best score $s_i^{\langle * \rangle}(t)$ in Equation (12). DAD-θ gates only defer-versus-commit; candidate construction, scoring, path planning, and swap/switch acceptance follow D-TP/D-TPTS. The safety override is based on pickup slack (see Equation (14)). Colors distinguish decision, action, and fallback/safety components in the token-turn logic.

Score Mode and Fair Comparisons.

DAD-θ can be applied using either the raw baseline score $s_{i,j}(t)$ (Equation (8)) or the scale-normalized score $\tilde{s}_{i,j}(t)$ (Equation (9)). In normalized mode, $\tilde{s}_{i,j}(t)$ replaces $s_{i,j}(t)$ throughout the underlying allocator (candidate ordering and DAD-θ gating), so normalization is an optional extension used for portability rather than part of the gate itself. All empirical comparisons keep the score mode fixed across the baseline and DAD-θ (and we report a raw-vs-normalized ablation in Appendix A.6 and Table A2).

Flowchart Summary.

Figure 2 summarizes the token turn logic and its interaction with the baseline commitment rules (including swap/switch acceptance in D-TPTS).

4.2. Background: Token Passing, Endpoints, and Assignable Tasks

Token Passing (TP) and Token Passing with Task Swaps (TPTS) are decoupled online MAPD algorithms for well-formed instances [4]. They use a synchronized shared-memory *token* that stores (i) the current planned paths of all agents, (ii) the task set, and (iii) current task assignments. At each time step, agents that reach the end of their current token path request the token and are served sequentially; the token holder plans a collision-free path that does not collide with already-reserved token paths and then updates the token [4].

Importantly, TP is *event-driven* rather than round-robin. An agent requests the token only when it reaches the end of its current token path and needs to update its plan. Hence, there is no fixed global ordering of agents across time steps: the effective queue emerges dynamically from agent availability, and any simultaneous requests are simply resolved sequentially by the implementation-level arbitration rule.

TP-family methods use *endpoints* as safe resting locations and enforce that token paths end at endpoints [4]. At a token turn for agent a_i at time t, let $\mathcal{T}_t$ denote the task set maintained by the underlying baseline allocator (unassigned tasks for TP/D-TP; all unexecuted tasks, including tasks assigned to other agents but not yet picked up, for TPTS/D-TPTS [4,9]). The token holder considers as candidates the subset

$$\mathcal{T}'_{i,t} = \{\tau_j \in \mathcal{T}_t \mid \text{no other token path ends at } u_j \text{ or } v_j\}. \tag{5}$$

This endpoint-occupancy filter is the standard *assignable* criterion in TP/TPTS and their deadline-aware variants [4,9]. When $\mathcal{T}'_{i,t} = \emptyset$, TP-family allocators apply a fallback: the agent keeps its current commitment and either waits at its endpoint (if it is not blocking any pending delivery location) or moves to an available non-task endpoint to avoid deadlock [4].

Illustrative Token-Turn Example.

Consider three agents a_1, a_2, a_3. Suppose a_1 and a_2 both reach the ends of their current token paths at time t, while a_3 is still traveling. Then only a_1 and a_2 request the token at time t, and they are served sequentially in some implementation-level order; the second requester plans against the reservations left by the first. Agent a_3 does not participate in that sequence until it later reaches the end of its own path and requests the token. Thus, the effective queue is induced by path completions and token availability, not by a fixed agent-ID ordering.

4.3. Baseline Deadline-Aware Score

D-TP/D-TPTS computes a pickup deadline $d_j^{\langle p \rangle}$ from each task deadline $d_j^{\langle d \rangle}$ by reverse-time planning from delivery to pickup [9]. Let $\pi_j^{\langle \mathrm{dum} \rangle}$ denote this reverse-time dummy path and let

$$\ell_j = |\pi_j^{\langle \mathrm{dum} \rangle}| - 1 \tag{6}$$

be its step length. Then

$$d_j^{\langle p \rangle} = d_j^{\langle d \rangle} - \ell_j. \tag{7}$$

Intuitively, ℓ_j is obtained by reverse-time planning under the current token paths and therefore serves as a proxy for the pickup-to-delivery travel time of task j given the current reservations. Thus, $d_j^{\langle p \rangle}$ is a reference to the latest pickup time (not a tight feasibility bound) used only for scoring.

For agent a_i at time t, define $\mathrm{loc}_i(t)$ as current location and $h(\cdot, \cdot)$ as shortest-path distance on G. For candidate task τ_j, the baseline score is

$$s_{i,j}(t) = \alpha \left(d_j^{\langle p \rangle} - t \right) + (1 - \alpha)\, h\big(\mathrm{loc}_i(t), u_j\big), \quad \alpha \in [0, 1]. \tag{8}$$

Lower is better. Because smaller pickup slack indicates higher urgency, the slack term favors urgent tasks while the distance term favors nearby pickups. The parameter α blends urgency and proximity: larger α increases weight on pickup slack $(d_j^{\langle p \rangle} - t)$ and thus emphasizes urgency, whereas smaller α increases weight on distance-to-pickup and emphasizes proximity. In particular, $\alpha = 1$ ranks candidates purely by picking up slack and $\alpha = 0$ ranks purely by distance. At $\alpha = 0$, the score ignores deadlines and reduces to the distance-based ordering used in TP/TPTS [4]. Operationally, the raw score can be read in three steps:

1. compute a reference latest pickup time $d_j^{\langle p \rangle}$ from the task deadline and the reverse-time pickup-to-delivery proxy;
2. evaluate the current pickup slack $d_j^{\langle p \rangle} - t$, which becomes smaller as the task approaches or exceeds that reference pickup time;
3. combine this urgency term with the current shortest-path distance to pickup, using α to determine how strongly urgency versus proximity should dominate the ranking.

Hence, a small raw score means that the task is urgent, nearby, or both, whereas a large raw score indicates a relatively unattractive commitment under the baseline rule.

4.4. Scale-Normalized Score

To reduce the dependency of θ on map/deadline scale, we use the normalized score

$$\tilde{s}_{i,j}(t) = \alpha \, \frac{d_j^{\langle p \rangle} - t}{\Delta_d} + (1 - \alpha) \, \frac{h\big(\mathrm{loc}_i(t), u_j\big)}{\bar{h}}. \tag{9}$$

Both scales are computed deterministically at run start from scenario-level information (map topology and the deadline distribution):

$$\Delta_d = d^{\langle + \rangle} - d^{\langle - \rangle}, \tag{10}$$

where $[d^{\langle - \rangle}, d^{\langle + \rangle}]$ is the support of the deadline offset distribution $(d_j^{\langle d \rangle} - r_j)$. In our benchmark settings, the deadline offset support width is $\Delta_d = 60$ for both short and long regimes.

$$\bar{h} = \frac{1}{|\mathcal{Q}|} \sum_{(u,v) \in \mathcal{Q}} h(u,v), \quad \mathcal{Q} \subseteq \{(u,v) \mid u \in \mathcal{E}, v \in \mathcal{E}, u \neq v\}, \tag{11}$$

where $\mathcal{E} \subseteq V$ is the set of task endpoint cells (pickup/delivery candidates), and $\mathcal{Q}$ is a deterministic sample of at most 2000 endpoint pairs used to estimate the mean distance. We construct $\mathcal{Q}$ deterministically from the map only; Appendix C gives the procedure. In TP-family algorithms, assignment decisions are made when the token holder is at an endpoint, so the distance term is naturally comparable to an endpoint-to-endpoint scale. We therefore use the mean endpoint distance $\bar{h}$ as a map-only reference scale. The practical role of normalization is to make θ dimensionless and more comparable across maps and deadline regimes, thereby reducing scenario-specific tuning burden and stabilizing sweep ranges. Both scales are computed once at run start and add no per-time step overhead. For degenerate cases, we clamp any non-positive or unavailable scale to 1; this safeguard does not affect our benchmark settings. We evaluate the practical benefit of normalization empirically through transfer experiments and a raw-vs-normalized ablation (Appendix A.6). In normalized mode, the same two ingredients are retained, but each is divided by a scenario-level reference scale before the final weighted sum is formed: Δ_d converts pickup slack into units of deadline-range width, and $\bar{h}$ converts pickup distance into units of a typical endpoint-to-endpoint travel length. This makes the threshold θ easier to interpret across scenarios: values of similar magnitude correspond to similar *relative* commitment quality, rather than to raw time-step or distance units tied to one particular map and deadline scale.

4.5. DAD-θ Deferral Rule

Given the assignable set $\mathcal{T}'_{i,t}$ from Equation (5), define

$$s_i^{\langle * \rangle}(t) = \min_{\tau_j \in \mathcal{T}'_{i,t}} \hat{s}_{i,j}(t), \tag{12}$$

where $\hat{s}_{i,j}(t) = s_{i,j}(t)$ in raw mode and $\hat{s}_{i,j}(t) = \tilde{s}_{i,j}(t)$ in normalized mode. DAD-θ applies the gate

$$\text{defer in this token turn} \iff s_i^{\langle * \rangle}(t) > \theta. \tag{13}$$

$\theta = \infty$ removes the DAD gate and therefore recovers the corresponding baseline allocator under the same score mode and safety handling. When the gate does not trigger, the token holder proceeds with the underlying baseline commitment step using the baseline score $\hat{s}_{i,j}(t)$ (for D-TP this assigns the best-scored candidate; for D-TPTS, swaps/switches must satisfy the baseline acceptance rules). Unlike D-TP/D-TPTS, which attempt to commit whenever at least one task is assignable, DAD-θ may defer even when $\mathcal{T}'_{i,t} \neq \emptyset$; this gate is the only behavioral difference. When DAD-θ defers, the token holder takes no new task in that token turn and then executes the TP-family fallback used by the baseline (waiting at its endpoint or moving to an available non-task endpoint) [4]. DAD-θ adds only one

scalar comparison after computing baseline candidate scores and does not add additional path-planning calls beyond D-TP/D-TPTS.

4.6. Safety Override and Starvation Control

To mitigate deferral-driven starvation (repeated postponement of task assignment) near deadlines, deferral is disabled whenever an assignable candidate has non-positive pickup slack:

$$\exists \tau_j \in \mathcal{T}'_{i,t} \text{ such that } d_j^{\langle p \rangle} - t \leq 0 \Rightarrow \text{ignore } \theta \text{ (do not defer).} \tag{14}$$

This safety override is parameter-free. When Equation (14) holds, DAD-θ only disables deferral; the token holder's task choice still follows the underlying baseline commitment rule (best score and baseline acceptance rules). We choose the zero-slack trigger in Equation (14) as the default guardrail because it is aligned with the pickup-slack term already used in the baseline score and is easy to monitor operationally (Margin-based variants (e.g., $d_j^{\langle p \rangle} - t \leq \kappa$) are possible but introduce an additional tuning dimension). Its role is primarily that of a *stability and starvation-control safeguard*, not a tight feasibility test. Because Equation (14) is defined over the assignable set $\mathcal{T}'_{i,t}$, it specifically prevents *deferral-driven* starvation: once an urgent task is assignable to the current token holder, DAD-θ can no longer continue deferring in that token turn. If a task is temporarily unassignable due to token constraints (endpoint occupancy/reservation consistency), it cannot be assigned by any TP-family method at that time step; in well-formed settings, tokens and reservations evolve, and tasks become assignable in later steps.

Illustrative Safety-Override Example.

Suppose that, at a token turn, the assignable set contains two tasks with pickup slacks 3 and -1, respectively. Even if the best overall score still exceeds θ, the existence of the task with non-positive pickup slack activates Equation (14), so the token holder must follow the baseline commitment rule rather than defer again. This example illustrates that the safety override is triggered by the presence of an urgent assignable task, not by the score threshold itself.

4.7. D-TPTS Integration

For D-TPTS, DAD-θ is evaluated in the baseline task-selection routine before commitment, so the deferral/safety logic is shared with D-TP. The candidate set remains the D-TPTS task set (all unexecuted tasks, including tasks assigned to other agents but not yet picked up) filtered by the assignable criterion in Equation (5) [4,9]. If the best-scored candidate is already assigned to another agent, the swap is accepted only if the token holder reaches the pickup location earlier than the current assignee; otherwise, the reassignment is rejected, and D-TPTS continues to the next candidate in score order [4,9]. Separately, D-TPTS supports in-agent task switching while traveling to pickup; switching is allowed only under urgency-and-distance dominance:

$$d_{\text{new}}^{\langle p \rangle} < d_{\text{cur}}^{\langle p \rangle} \quad \wedge \quad h\big(\text{loc}_i(t), u_{\text{new}}\big) < h\big(\text{loc}_i(t), u_{\text{cur}}\big). \tag{15}$$

Here, cur denotes the agent's current task and new denotes the candidate new task; u_{cur} and u_{new} are their pickup vertices. DAD-θ does not modify these acceptance conditions; it only gates whether the token holder commits in the current token turn. For D-TPTS, the DAD gate (Equation (13)) is evaluated *before* the baseline swap/switch acceptance logic to keep DAD-θ as a minimal gate on the baseline score ordering shared with D-TP. As a result, even when the gate allows commitment, D-TPTS may reject the best-scored candidate

under its acceptance rules and then accept a higher-scored candidate (or none). Thus, θ gates the *opportunity to commit now* rather than imposing a hard constraint on the score of the eventual committed task in D-TPTS. Note that if $s_i^{\langle * \rangle}(t) > \theta$, then all candidates in $\mathcal{T}'_{i,t}$ have score $> \theta$, so deferring cannot forgo any below-threshold candidate in that token turn.

4.8. Operational Calibration and Monitoring

DAD-θ is intended as a lightweight, operator-tunable *commitment knob* for TP-family MAPD-D stacks: it changes only whether the token holder commits now or defers to preserve short-horizon flexibility. This subsection summarizes a practical workflow for selecting θ offline and maintaining it online.

Offline calibration (digital twin).

Given a fixed planning stack (map, endpoint set, path planner, and baseline allocator), we recommend:

- **Choose score mode.** Normalized scoring (Equation (9)) yields a dimensionless θ and is preferable when map scale or deadline scale may change. It uses the deterministic run-start scales Δ_d and $\bar{h}$ (Equations (10) and (11)); if the deadline-offset support is not fixed a priori, Δ_d can be estimated from a rolling window (with clipping) and refreshed during periodic re-calibration.
- **Tune baseline α under always-assign.** First select α for the baseline allocator with $\theta = \infty$ on representative workloads (as in our paired α sweep; Section 6.3). Then fix α and sweep θ.
- **Select θ against explicit service-level targets.** Besides cumulative tardiness T (Equation (3)), monitor on-time completion and tail tardiness (MaxTard; per-run maximum task tardiness) as service-level agreement (SLA) proxies. In our benchmark under normalized scoring, the selected best-θ values lie in $[0.7, 1.0]$, which provides a practical starting range.

As a concrete benchmark-style example, in Dense/Long with D-TP under normalized scoring, the selected setting $\theta^{\langle * \rangle} = 1$ reduced mean cumulative tardiness from 141.3 to 86.9 (a 38.5% reduction), improved on-time completion from 92.8% to 95.5%, and kept the safety-trigger rate at 5.5% (Tables 1, 2 and A1). This is the intended calibration logic: choose a point near the bottom of the sweep curve that delivers the target service-level gain while keeping safety-trigger frequency low enough that the gate remains genuinely discretionary rather than being dominated by the safety override.

Robust selection (band rather than point).

Rather than deploying a single best grid point, operators can choose any θ within a *joint-improvement band* around the selected $\theta^{\langle * \rangle}$: a contiguous interval on the sweep grid where both paired $\Delta T = T(\theta) - T(\infty)$ and $\Delta M = M(\theta) - M(\infty)$ relative to $\theta = \infty$ are negative, as visualized later in the theta-sweep results. This provides margin against finite-sample noise and workload variation while keeping DAD-θ active (nonzero deferrals at the selected $\theta^{\langle * \rangle}$). If deadline compliance and tail-risk control are the primary objectives, a lower-to-middle point of this band is often preferable because it preserves more deferral activity; if throughput or operational conservatism is prioritized, the upper end of the band provides a more baseline-like setting while still retaining the observed joint benefit.

Table 1. Best-θ summary on the benchmark (means over 100 seeds). For compactness, we denote $\alpha^{\langle * \rangle}$ and $\theta^{\langle * \rangle}$ by α^* and θ^*. Top: selected parameters and tardiness T with $\Delta T(\%) = 100 \cdot (T(\theta^*) - T(\infty))/T(\infty)$; bottom: makespan M, its change $\Delta M = M(\theta^*) - M(\infty)$, and DAD deferrals (Def; mean number of DAD deferrals per run; Section 6.4) at θ^*. Bold indicates values at θ^*.

Scenario	Algo	α^*	θ^*	$T(\infty)$	$T(\theta^*)$	ΔT (%)
Dense/Short	D-TP	0.2	**1**	1575.1	**1290.8**	**−18.1**
Dense/Short	D-TPTS	0.3	**0.9**	1399.3	**1270.7**	**−9.2**
Dense/Long	D-TP	0.5	**1**	141.3	**86.9**	**−38.5**
Dense/Long	D-TPTS	0.5	**1**	140.0	**88.2**	**−37.0**
Sparse/Short	D-TP	0	**1**	896.1	**603.3**	**−32.7**
Sparse/Short	D-TPTS	0.3	**0.7**	811.3	**565.9**	**−30.2**
Sparse/Long	D-TP	0.2	**1**	38.9	**16.7**	**−57.1**
Sparse/Long	D-TPTS	0.2	**1**	41.0	**17.3**	**−57.9**

Scenario	Algo	α^*	θ^*	$M(\infty)$	$M(\theta^*)$	ΔM	Def
Dense/Short	D-TP	0.2	**1**	372.6	**368.8**	**−3.8**	59.4
Dense/Short	D-TPTS	0.3	**0.9**	372.9	**371.5**	**−1.4**	81.8
Dense/Long	D-TP	0.5	**1**	381.7	**377.2**	**−4.5**	292.5
Dense/Long	D-TPTS	0.5	**1**	381.2	**377.3**	**−3.9**	292.7
Sparse/Short	D-TP	0	**1**	552.8	**547.9**	**−4.9**	363.8
Sparse/Short	D-TPTS	0.3	**0.7**	553.7	**547.8**	**−5.9**	1151.9
Sparse/Long	D-TP	0.2	**1**	555.6	**545.3**	**−10.3**	611.4
Sparse/Long	D-TPTS	0.2	**1**	556.3	**545.6**	**−10.7**	608.9

Table 2. Deadline adherence and tail tardiness at best-θ (benchmark). On-time rate is the fraction of tasks completed by their delivery deadlines; MaxTard is the per-run maximum task tardiness (steps), averaged over 100 seeds. Δ is the change in on-time rate in percentage points (pp) from $\theta = \infty$ to best-θ. Bold indicates values at best-θ.

Scenario	Algo	θ^*	On-Time (∞)	On-Time (θ^*)	Δ (pp)	MaxTard (∞)	MaxTard (θ^*)
Dense/Long	D-TP	**1**	92.8%	**95.5%**	**+2.7**	30.2	**26.2**
Dense/Long	D-TPTS	**1**	92.9%	**95.5%**	**+2.6**	30.4	**26.6**
Dense/Short	D-TP	**1**	53.0%	**57.9%**	**+4.9**	80.6	**74.1**
Dense/Short	D-TPTS	**0.9**	53.3%	**56.6%**	**+3.3**	70.6	**69.0**
Sparse/Long	D-TP	**1**	98.0%	**99.3%**	**+1.3**	22.0	**13.0**
Sparse/Long	D-TPTS	**1**	98.0%	**99.3%**	**+1.3**	22.4	**13.0**
Sparse/Short	D-TP	**1**	64.7%	**74.7%**	**+10.0**	63.0	**56.3**
Sparse/Short	D-TPTS	**0.7**	66.9%	**75.1%**	**+8.3**	59.5	**54.1**

Online monitoring and re-calibration triggers.

Because deferral interacts with token reservations and endpoint occupancy, log the DAD deferral rate and the safety-trigger rate (Section 6.4). Drift in these rates indicates a workload shift and motivates re-calibration of θ (and potentially upstream parameters such as α). Transfer results suggest that a threshold calibrated in a nearby regime can provide a strong warm start before target-side tuning (Appendix A.4).

5. An Interpretive One-Step Analysis of Deferral Thresholds

This section provides an *interpretive* one-step analysis intended to clarify why intermediate thresholds can be beneficial and why θ sweeps can exhibit a single-valley (U-shaped) sensitivity pattern. Because repeated deferrals alter future states, score distributions, and later decisions, we do not attempt a full performance theory for the deterministic multi-step policy. Instead, we study a one-step intervention surrogate that compares forced *defer* versus forced *assign* at a baseline decision point and then follows the baseline policy thereafter. This surrogate is meant only as a qualitative explanatory device: it suggests when deferral is likely to help and provides intuition for the observed sweep shapes, but

it is not a guarantee for arbitrary online workloads. In particular, it should not be read as an additive decomposition of the full coupled multi-agent dynamics induced by repeated token turns and evolving reservations.

Takeaway.

- Intermediate thresholds can help by deferring only in regimes where the baseline score indicates a low-quality commitment, while avoiding pathological over-deferral.
- Under a mild single-crossing pattern—where the marginal value of deferring switches sign at most once as the score increases—expected performance as a function of θ tends to be single-valley/U-shaped.
- Naive deferral patterns (e.g., random/periodic) can defer in the wrong regime and are therefore not reliably beneficial, consistent with our comparator-policy experiments.

Interpretive summary.

At a safety-inactive token turn, deferring has two opposing effects: it incurs an immediate one-turn delay, but it can also preserve flexibility if new releases or reservation changes make a better task assignable soon after. When the current best assignable option is already good, the immediate delay is more likely to dominate; when the current best option is poor, waiting can be beneficial. This is the qualitative mechanism behind the empirical observation that intermediate thresholds tend to work best.

Under a mild single-crossing view of this trade-off, the benefit of deferral changes sign at most once as the best current score worsens, which yields a single-valley/U-shaped surrogate dependence on θ. This is the only claim we use in the main text: defer when the current option is sufficiently unattractive, but avoid indiscriminate waiting. The formal surrogate definition, single-crossing assumption, and proof sketch are moved to Appendix B.

These qualitative implications are consistent with two empirical findings: (i) the θ sweeps often show a single-valley sensitivity curve, and (ii) naive deferral patterns (random/periodic) are not reliably beneficial because they can defer even when the current best option is already good (Appendix A.3).

6. Experimental Setup

6.1. Implementation and Reproducibility

All experiments use a single containerized runtime environment for reproducibility (the accompanying codebase includes CPU/GPU Dockerfiles and a Docker Compose configuration). We pre-generate per-seed task schedules (with pre-generated deadlines) and replay the same schedules across all algorithms and θ values to enable strict paired comparisons (common random numbers). Each run outputs a structured JSON metrics file, and all tables and figures in this manuscript are generated from these per-run artifacts by fixed scripts in the accompanying codebase.

Space–Time Path Planning (Shared Across All Methods).

Collision-free paths are generated by a reservation-based space–time A* planner shared across all compared methods [4,9]. The planner runs A* search in a time-expanded graph with states (v, t) and move/wait transitions, treating other agents' current token paths as fixed reservations and enforcing vertex conflicts, edge-swap conflicts, and endpoint-occupancy constraints (agents rest at the final token location). Consistent with TP-family well-formed assumptions, intermediate traversal through endpoint cells (except the start and goal) is disallowed. DAD-θ does not modify the planner; it only gates commit-versus-defer decisions at token turns. Experiments were executed on Ubuntu 24.04.3 LTS

with an AMD Ryzen Threadripper PRO 7975WX CPU (Advanced Micro Devices, Inc., Santa Clara, CA, USA; 64 logical cores) and 251 GiB RAM; independent runs are executed in parallel with the number of worker processes matched to the available CPU cores. Reported runtimes are wall-clock times on this machine.

6.2. Map and Workload

We evaluate two well-formed grid maps: a benchmark map (37 × 23) and a larger map (48 × 35) preserving the same structural pattern. In both maps, the outer ring is walls, and interior start cells not selected as initial agent positions are treated as task endpoints, following the benchmark-style map construction used in prior TP-family MAPD studies [4,9]. Figure 3 visualizes both maps and the endpoint/travel structure.

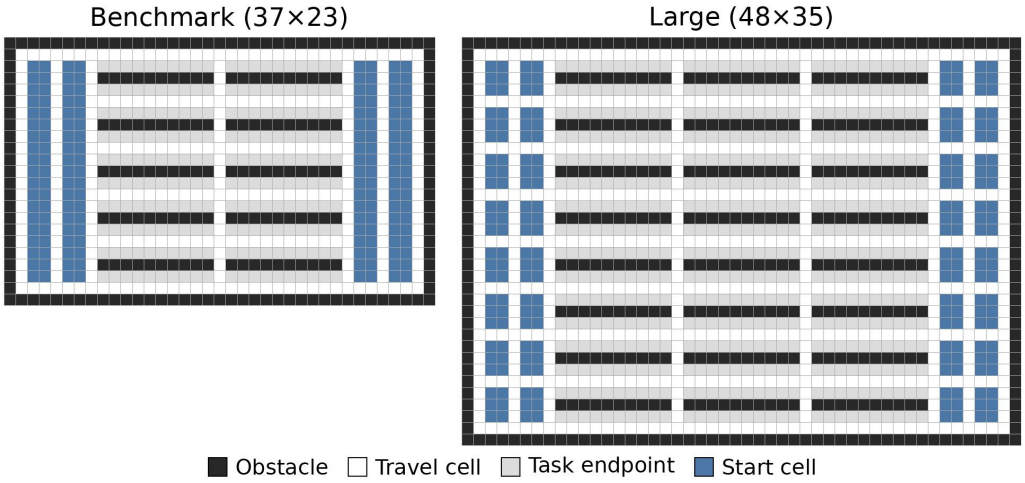


Figure 3. Well-formed maps used in experiments. **Left**: benchmark (37 × 23). **Right**: large (48 × 35). Colors denote tile types (legend shown in the figure): obstacles, travel cells, task endpoints, and start cells. Faint grid lines indicate individual cells.

Tasks are pre-generated with a fixed count and pre-generated deadlines. In the benchmark condition (base map), we use:

- Seeds: 1001–1100 (100 seeds).
- Tasks per run: exactly $n = 151$ (fixed across all runs; no task-count drift).
- Number of agents: $m = 15$.
- Scenarios: Dense/Short, Dense/Long, Sparse/Short, Sparse/Long.
- Release windows: dense $[0, 300]$, sparse $[0, 500]$.
- Deadline offsets: short $[20, 80]$, long $[60, 120]$ (uniform).

Dense/Short is the most time-critical regime (highest arrival intensity and tightest deadline offsets), while Sparse/Long is the most relaxed. We use a fixed task count to avoid confounding algorithm comparisons with episode length. The specific value 151 follows the MAPD-D benchmark-style configuration used with D-TP/D-TPTS [9] and is not otherwise special. The dense/sparse release windows and short/long deadline offsets form a compact 2 × 2 scenario design that spans both congestion pressure and deadline pressure without changing the underlying map topology, making it easier to isolate the effect of commitment timing across more and less time-critical regimes. This finite-task evaluation corresponds to optimizing a fixed order window (e.g., a batch or shift) and is standard in MAPD-D benchmarks. In continuous operations with ongoing arrivals, DAD-θ can be applied online in the same token turn manner; deferral and safety-trigger rates provide simple telemetry for monitoring stability and workload drift.

For additional map-scale generalization on the larger map, we evaluate two conditions:

- **C1 (map-only):** $m = 15$ agents and $n = 151$ tasks.
- **C2 (map + load):** $m = 30$ agents and $n = 304$ tasks (scaled by free-cell ratio and rounded).

Workload Generation (Pre-Generated Task Files).

For each scenario, seed, map, and agent count, we pre-generate a task schedule and replay it for all algorithms and θ values. The generation procedure mirrors prior MAPD-D endpoint sampling [9] and is deterministic given the random seed: we sample the m initial agent locations uniformly without replacement from the start cells; treat remaining start cells as task endpoints (removing them from the non-task endpoint pool) [4,9]; generate exactly n tasks by sampling release times uniformly from the scenario window and sampling ordered pickup–delivery endpoint pairs uniformly (rejecting pickup=delivery and duplicate ordered pairs to avoid repeated identical tasks); set delivery deadlines by $d_j^{\langle d \rangle} = r_j + \delta_j$ with δ_j uniform on the scenario offset interval; and sort tasks by release time (breaking ties by generation order) for replay. Pickups and deliveries are drawn from the same endpoint pool.

6.3. Protocol

The benchmark evaluation uses a four-stage paired protocol:

1. α **sweep (baseline):** $\theta = \infty$, $\alpha \in \{0, 0.025, 0.05, 0.1, 0.2, 0.3, 0.5\}$ for D-TP and D-TPTS. We restrict this baseline grid to $\alpha \leq 0.5$ to keep the sweep lightweight and because our focus is on the effect of θ rather than on urgency-dominant ranking; exploring $\alpha > 0.5$ is left for future work. The grid is denser near zero because small-to-moderate urgency weights are the practically relevant regime in our benchmark settings.
2. θ **sweep (DAD):** fix $\alpha = \alpha^{\langle * \rangle}$ from phase 1 per scenario–algorithm; evaluate $\theta \in \{-1.0, -0.9, \ldots, 2.0\}$ (step 0.1 on $[-1, 2]$), using the corresponding $\theta = \infty$ run from phase 1 (at $\alpha^{\langle * \rangle}$) as the paired always-assign baseline. Negative θ values are allowed because pickup slack can be negative. In normalized units, this range spans aggressive deferral through near-baseline behavior, and the selected best-θ values reported later lie well inside it.
3. **Policy sweep:** evaluate threshold/random/periodic policies under the same assignment/safety framework. Threshold settings are $\theta \in \{\infty, -0.5, 0, 0.25, 0.5, 1.0\}$; random defers use Bernoulli probability $q \in \{0.05, 0.1, 0.2\}$; periodic defers trigger when $(t + i) \bmod k = 0$ with $k \in \{2, 3, 4, 5\}$ (decision time t, agent index i).
4. θ **transfer:** for each source scenario and algorithm, select source-best $\theta^{\langle * \rangle}$ by mean tardiness, then apply it to all target scenarios without target-side re-tuning.

Unless stated otherwise, the main protocol uses normalized score mode (Equation (9)) with scenario-based scale estimation (Equations (10) and (11)); $\alpha^{\langle * \rangle}$ is selected in phase 1 and fixed thereafter. This separation of baseline tuning (α) from deferral tuning (θ) is intentional: it keeps the underlying task-ranking rule fixed while isolating the marginal effect of commitment timing. The overall experimental total is 101,600 runs (benchmark + large-map generalization).

6.4. Validity and Metrics

A run is counted as successful only if:

- all tasks are completed,
- simulation terminates because all tasks are completed,
- vertex collisions, edge collisions, and teleportations (non-adjacent moves within one time step) are all zero,

- no stall timeout is detected (we terminate if no pickup or delivery is completed by any agent for 1000 consecutive time steps).

The stall timeout is a conservative guardrail to detect pathological non-progress loops; it never triggers in the main benchmark and large-map protocols (100% validity) and is only used to flag failures in stress settings such as safety-OFF under aggressive deferral (Appendix A.5).

Our main reported metric is cumulative tardiness T (Equation (3)). Additional reported metrics are makespan M (Equation (4)), deadline adherence (on-time rate; fraction of tasks with $\epsilon_j = 0$), tail tardiness (MaxTard; per-run maximum task tardiness), DAD deferrals (Def), safety-trigger rate, and runtime.

Terminology.

In TP-family algorithms, a *token turn* is a time step at which an agent holds the token and updates its plan in the shared token. In each token turn, the allocator considers a set of candidate tasks and the *assignable set* $\mathcal{T}'_{i,t}$ after the endpoint-occupancy filter (Equation (5)). We use:

- **Assignment decision:** $\mathcal{T}'_{i,t} \neq \emptyset$.
- **No-assignable turn:** at least one candidate task exists but $\mathcal{T}'_{i,t} = \emptyset$.
- **Deferral event:** an assignment decision where safety is inactive and the DAD gate triggers (Equation (13)).
- **Safety-trigger decision:** an assignment decision where the safety condition holds (Equation (14)), so deferral is disabled.

Deferrals and Safety Triggers.

In DAD-θ, a *deferral* event occurs at an assignment decision when the safety override is inactive (Equation (14) does not hold) and the gate triggers (Equation (13)), causing the token holder to skip the commitment step and take the fallback action for that token turn. We report Def as the per-run count of such deferral events (means over seeds in tables/figures). We additionally compute the *deferral rate* as this count divided by the number of safety-inactive assignment decisions. The *safety-trigger rate* is the fraction of assignment decisions at which the safety condition in Equation (14) is met (i.e., an urgent assignable task exists), which is the subset of decisions where deferral is forcibly disabled. For paired comparisons, we use two-sided Wilcoxon signed-rank tests on per-seed paired differences (zero differences dropped) and report Holm step-down corrected p-values at the 0.05 significance level within each experiment block and metric [37]. For best-θ comparisons, we additionally report rank-biserial correlation r_{rb} and percentile bootstrap confidence intervals for the mean paired difference. Unless noted, error bands labeled "mean $\pm$ 95% CI" are two-sided t-based confidence intervals for the mean over seeds.

Mechanism Telemetry.

To probe *why* DAD-θ can reduce tardiness under TP-family coordination, we log additional run-level diagnostics that reflect two key shared-constraint mechanisms: (i) *assignable-set shrinkage* due to the endpoint-occupancy filter in Equation (5), and (ii) *reservation-induced routing inflation* in space–time planning. Concretely, in each token turn, we count how many candidate tasks are considered by the baseline routine before applying the endpoint-occupancy filter and how many of those candidates are blocked by the filter. We summarize this as a *blocked fraction* (blocked candidates divided by total considered candidates) and a *no-assignable rate* (fraction of token turns that have at least one candidate task but yield zero assignable candidates after filtering). Separately, we record a *detour ratio* from the space–time planner calls as the aggregate relative inflation of planned steps versus reservation-free

shortest-path distance on G, which serves as a proxy for waiting and detours caused by reservations and congestion. These diagnostics are reported as descriptive paired differences between best-θ and $\theta = \infty$.

7. Results

Executive Summary.

Across the benchmark conditions, best-θ consistently improves cumulative tardiness over the always-assign baseline and also improves on-time completion, typically with concurrent makespan gains and negligible runtime impact (Tables 1–3). Repeated seed-split holdout validation further shows that the tardiness benefit generalizes strongly across seeds (tardiness-improvement rate ≥ 0.905 in all scenario–algorithm pairs; Table 4). The supplementary analyses then show that these gains are specific to thresholded deferral, remain robust under transfer and larger-map generalization, and are not explained solely by the safety override (Appendix A).

Table 3. Runtime comparison between always-assign ($\theta = \infty$) and best-θ (from Table 1). Time is mean wall-clock seconds per run; ms/step is mean milliseconds per time step computed as time divided by makespan (100 seeds).

Scenario	Algo	θ^*	Time (∞)	Time (θ^*)	Δ%	ms/step (∞)	ms/step (θ^*)
Dense/Short	D-TP	1	1.637	1.618	−1.2	4.397	4.398
Dense/Short	D-TPTS	0.9	1.920	1.881	−2.0	5.156	5.056
Dense/Long	D-TP	1	4.087	4.012	−1.8	10.736	10.669
Dense/Long	D-TPTS	1	4.343	4.184	−3.7	11.427	11.121
Sparse/Short	D-TP	1	2.098	2.095	−0.1	3.797	3.818
Sparse/Short	D-TPTS	0.7	2.493	2.473	−0.8	4.505	4.519
Sparse/Long	D-TP	1	3.734	3.697	−1.0	6.727	6.794
Sparse/Long	D-TPTS	1	4.034	3.842	−4.7	7.250	7.056

$\Delta\% = 100 \cdot (\text{Time}(\theta^*) - \text{Time}(\infty))/\text{Time}(\infty)$.

Table 4. Seed-split holdout validation on the benchmark (5-fold, repeated 40 times; 200 test folds per scenario–algorithm pair). In each split, $\theta^{\langle * \rangle}$ is selected on training seeds by minimum mean cumulative tardiness among finite θ values, then evaluated on held-out test seeds as paired differences ΔT (tardiness) and ΔM (makespan) versus $\theta = \infty$. "Joint" is the fraction of test folds where both test-fold mean $\Delta T < 0$ and $\Delta M < 0$; "Tard" and "Mk" are the marginal rates. "Avg" denotes the average across test folds of the test-fold mean; "Worst" denotes the maximum across test folds of the test-fold mean. θ mode is the most frequently selected $\theta^{\langle * \rangle}$ (frequency in parentheses). For readability, we bold the larger of the two algorithms' rates within each scenario for Joint/Tard/Mk (ties are in bold).

Scenario	Algo	Joint	Tard	Mk	Avg ΔT	Avg ΔM	Worst ΔT	Worst ΔM	θ Mode
Dense/Short	D-TP	**0.930**	**1.000**	**0.930**	−283.15	−3.76	−79.40	+5.55	1.0 (0.99)
Dense/Short	D-TPTS	0.445	0.905	0.490	−90.51	+0.22	+100.90	+13.35	0.9 (0.47)
Dense/Long	D-TP	**0.940**	**0.990**	**0.950**	−54.07	−4.47	+3.70	+4.65	1.0 (0.99)
Dense/Long	D-TPTS	0.905	**0.990**	0.915	−50.68	−3.83	+4.25	+5.30	1.0 (0.98)
Sparse/Short	D-TP	**0.985**	**1.000**	**0.985**	−292.19	−4.94	−210.60	+1.80	1.0 (0.99)
Sparse/Short	D-TPTS	0.975	**1.000**	0.975	−245.35	−5.84	−175.55	+1.00	0.7 (1.00)
Sparse/Long	D-TP	**1.000**	**1.000**	**1.000**	−21.73	−10.44	−2.90	−3.30	1.0 (0.94)
Sparse/Long	D-TPTS	**1.000**	**1.000**	**1.000**	−22.48	-10.64	−7.90	−4.70	1.0 (0.79)

7.1. Best-θ Summary

Figure 4 and Table 1 summarize gains from $\theta = \infty$ to best-θ and show that best-θ reduces tardiness in all pairs, reduces makespan in mean, and induces nonzero deferrals (Def > 0; Table 1).

Table 2 reports complementary service-level indicators at $\theta = \infty$ and best-θ: the on-time completion rate (fraction of tasks with zero tardiness) and a tail indicator (MaxTard;

per-run maximum task tardiness, mean over seeds). DAD-θ increases the on-time rate and reduces the tail in all 8 scenario–algorithm pairs.

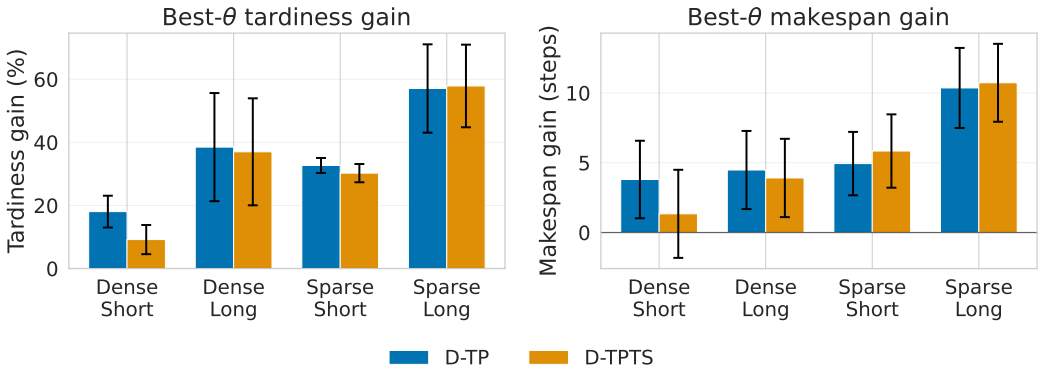


Figure 4. Benchmark best-θ gains over the always-assign baseline $\theta = \infty$ for each scenario–algorithm pair (mean $\pm$95% CI over 100 paired seeds). Each pair compares the selected thresholded DAD-θ setting against the same baseline allocator under the same score mode.

To make the baseline-versus-optimized comparison more concrete, Figure 5 shows direct side-by-side values for two representative scenario–algorithm pairs: Dense/Long with D-TP and Sparse/Short with D-TPTS. In both examples, the selected best-θ setting improves not only cumulative tardiness but also on-time completion and tail tardiness relative to the always-assign baseline.

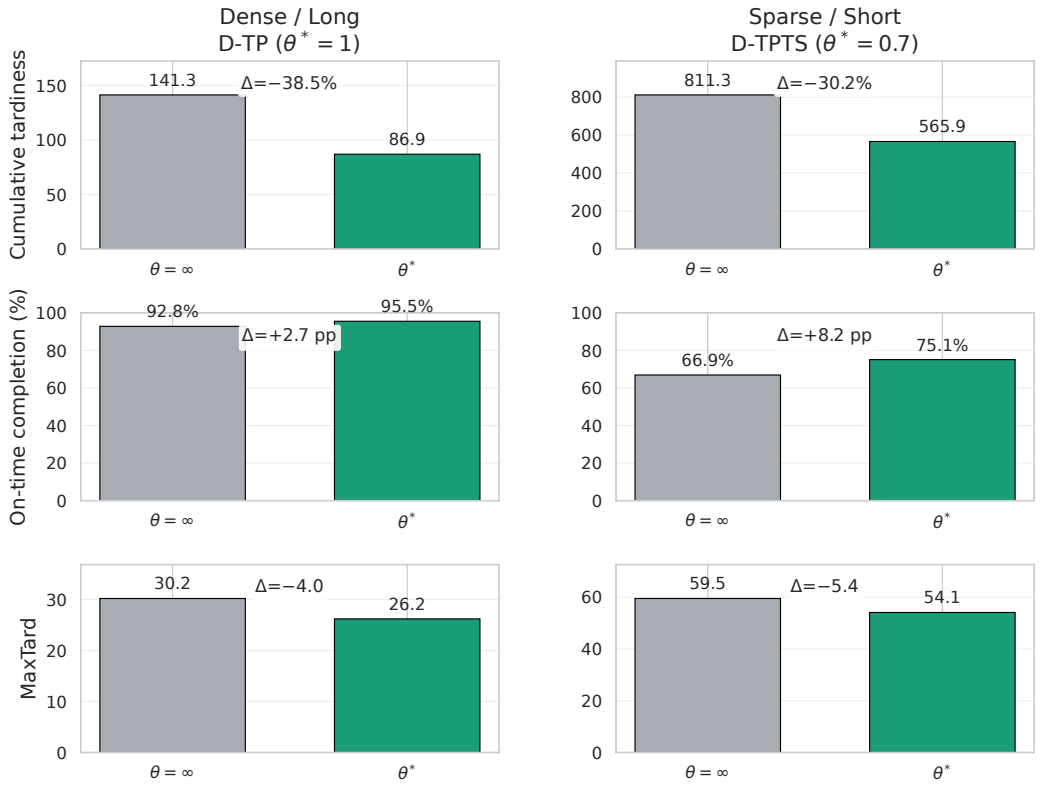


Figure 5. Direct side-by-side comparison between the always-assign baseline $\theta = \infty$ and the selected best-θ policy for two representative scenario–algorithm pairs. **Left** column: Dense/Long with D-TP ($\theta^* = 1$). **Right** column: Sparse/Short with D-TPTS ($\theta^* = 0.7$). The three rows report cumulative tardiness, on-time completion rate, and MaxTard. These representative examples complement the aggregate summaries in Figure 4 and Tables 1 and 2 by showing the baseline and optimized policies on the same axis.

7.2. DAD-θ Sweep

Figure 6 shows paired Δ tardiness versus $\theta = \infty$ (mean ±95% CI) across the sweep grid. Negative paired delta indicates improvement versus always-assign.

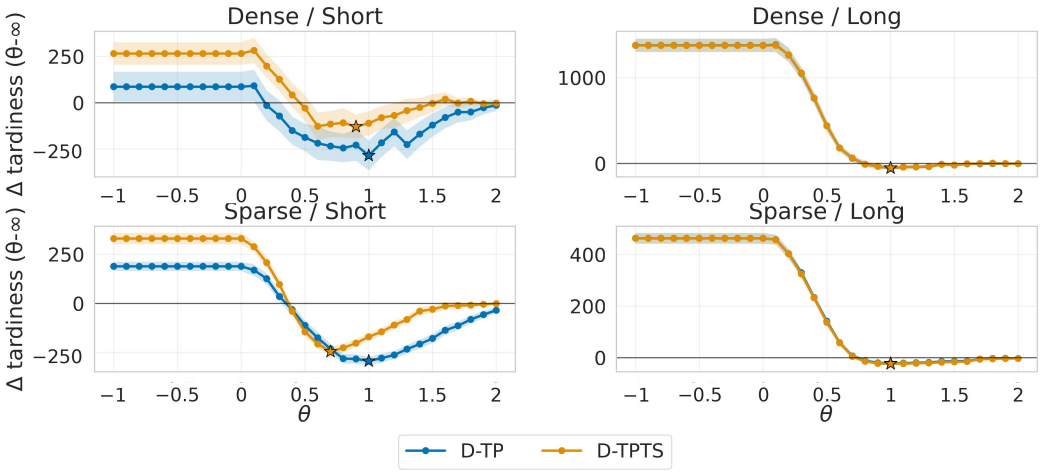


Figure 6. Benchmark paired Δ tardiness versus the always-assign baseline $\theta = \infty$ across the normalized θ sweep (mean ±95% CI). Each panel is a scenario; each curve is one TP-family baseline. Negative is better. Stars mark the selected best-θ values, black horizontal lines indicate zero difference from the always-assign baseline, and shaded bands show 95% confidence intervals.

Figure 7 shows corresponding paired Δ makespan behavior. At best-θ, all scenario–algorithm pairs have lower mean makespan than the $\theta = \infty$ baseline (Table 1).

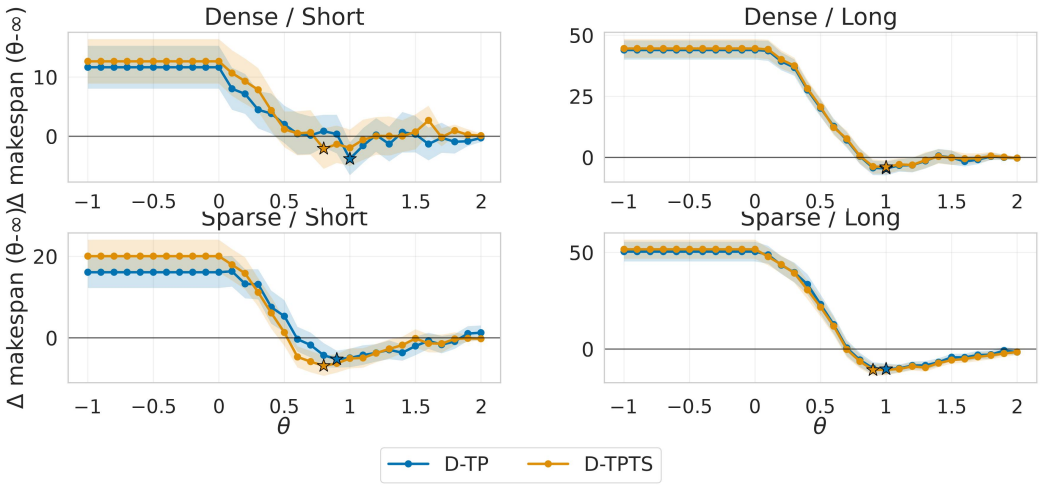


Figure 7. Benchmark paired Δ makespan versus the always-assign baseline $\theta = \infty$ across the normalized θ sweep (mean ±95% CI). Each panel is a scenario; each curve is one TP-family baseline. Negative is better. Stars mark the selected best-θ values, black horizontal lines indicate zero difference from the always-assign baseline, and shaded bands show 95% confidence intervals.

Taken together with Table 2, the sweep supports a practical calibration view. As θ increases, DAD-θ becomes less active and approaches the always-assign baseline; as θ becomes too small, the method over-defers and loses performance. The useful operating regime is the intermediate band where tardiness improves while on-time completion and MaxTard also improve, which is the regime that the operational workflow in Section 4.8 is designed to target.

Joint-Improvement Bands (Not a Single Point).

To summarize robustness, we define a mean-based *joint-improvement band*: the maximal contiguous interval on the sweep grid containing the selected $\theta^{\langle * \rangle}$ (minimum mean tardiness) for which the mean paired Δ tardiness and Δ makespan versus $\theta = \infty$ are both negative (Figures 6 and 7). This band is descriptive (not a significant region); on the benchmark, bands contain 2–15 grid points (step 0.1).

7.3. Runtime

We report simulator wall-clock computation time for $\theta = \infty$ and the corresponding best-θ settings in Table 3. DAD-θ has negligible runtime impact in our measured settings: total-time changes are within 5% (all are small decreases) and per-time-step changes are within 0.31 ms (increases by at most 0.07 ms).

7.4. Seed-Split Holdout Validation

Best-θ in the main protocol is selected by minimizing mean tardiness on the same seed set used for evaluation. To reduce selection bias concerns, we perform a repeated seed-split holdout validation on the benchmark: we repeatedly partition the 100 seeds into 5 folds (40 repeats), select $\theta^{\langle * \rangle}$ on training seeds only by minimum mean cumulative tardiness among finite θ values, and evaluate the paired differences ΔT (tardiness) and ΔM (makespan) versus $\theta = \infty$ on held-out test seeds. We restrict selection to finite θ values to evaluate generalization of deferral thresholds; $\theta = \infty$ is reserved as the paired baseline. Table 4 summarizes the resulting rates and worst-case test-fold means. Tardiness improvements generalize strongly across seeds (tardiness-improvement rate (Tard) ≥ 0.905 for all pairs), while makespan improvements are typically but not universally robust (Dense/Short with D-TPTS is the weakest case). In this validation, we keep α fixed at the scenario–algorithm-specific $\alpha^{\langle * \rangle}$ selected in phase 1 of the main protocol, so the holdout primarily isolates generalization of θ conditional on a fixed baseline tuning.

7.5. Paired Statistical Test at Best-θ

Because best-θ is selected by minimizing mean tardiness on the same seed set used for evaluation, the tests and confidence intervals in this subsection should be interpreted as descriptive post-selection summaries. The seed-split holdout validation above provides the main evidence that threshold selection generalizes across seeds. Using paired Wilcoxon tests versus $\theta = \infty$:

- **Tardiness:** all 8 scenario–algorithm pairs improve, and all 8 are significant both before and after Holm correction.
- **Makespan:** all 8 pairs improve in mean; 7 of 8 are significant at raw $p < 0.05$, and 7 of 8 remain significant after Holm correction.

After Holm correction, the only non-significant makespan case is Dense/Short (D-TPTS); all others remain significant improvements. Detailed paired statistics (mean Δ with 95% CI, effect sizes, and raw/Holm-adjusted p-values) are reported in Table 5. For interpretation, $r_{\text{rb}} \in [-1, 1]$ summarizes the dominance of paired improvements: $r_{\text{rb}} = -1$ means all non-tied pairs improve, $r_{\text{rb}} = 0$ indicates a balance of improvements and degradations, and $r_{\text{rb}} = 1$ means all non-tied pairs worsen.

Table 5. Paired Wilcoxon signed-rank tests for best-θ (selected by mean tardiness) versus always-assign ($\theta = \infty$). We report mean paired differences Δ (candidate minus baseline) with 95% percentile bootstrap confidence intervals for the mean (20,000 resamples), rank-biserial correlation r_{rb} (Wilcoxon effect size), raw two-sided p-values, and Holm-adjusted p-values within each metric family (8 tests). Top: tardiness (ΔT); bottom: makespan (ΔM). Negative Δ and negative r_{rb} indicate improvement over $\theta = \infty$. In the makespan panel, bold indicates Holm-significant improvements ($p_{\mathrm{Holm}} < 0.05$); all tardiness cases are Holm-significant.

Scenario	Algo	θ^*	ΔT (Mean [95% CI])	r_{rb}	p	p_{Holm}
Dense/Short	D-TP	1	−284.33 [−364.53, −208.76]	−0.75	7.1×10^{-11}	2.8×10^{-10}
Dense/Short	D-TPTS	0.9	−128.63 [−191.97, −64.49]	−0.48	3.7×10^{-5}	3.7×10^{-5}
Dense/Long	D-TP	1	−54.41 [−79.60, −32.18]	−0.59	3.1×10^{-7}	9.3×10^{-7}
Dense/Long	D-TPTS	1	−51.80 [−76.33, −29.89]	−0.57	8.9×10^{-7}	1.8×10^{-6}
Sparse/Short	D-TP	1	−292.85 [−313.58, −271.87]	−1	$< 1 \times 10^{-16}$	$< 1 \times 10^{-16}$
Sparse/Short	D-TPTS	0.7	−245.35 [−268.47, −221.90]	−1	$< 1 \times 10^{-16}$	$< 1 \times 10^{-16}$
Sparse/Long	D-TP	1	−22.20 [−27.52, −16.90]	−0.82	3.6×10^{-12}	1.8×10^{-11}
Sparse/Long	D-TPTS	1	−23.71 [−29.01, −18.45]	−0.87	2.5×10^{-13}	1.5×10^{-12}
Scenario	Algo	θ^*	ΔM (mean [95% CI])	r_{rb}	p	p_{Holm}
Dense/Short	D-TP	1	**−3.80 [−6.56, −1.13]**	−0.26	0.02447	**0.04894**
Dense/Short	D-TPTS	0.9	−1.34 [−4.43, 1.85]	−0.16	0.1739	0.1739
Dense/Long	D-TP	1	**−4.48 [−7.25, −1.75]**	−0.34	0.003686	**0.01475**
Dense/Long	D-TPTS	1	**−3.91 [−6.68, −1.23]**	−0.29	0.01454	**0.04363**
Sparse/Short	D-TP	1	**−4.94 [−7.15, −2.64]**	−0.5	1.9×10^{-5}	**0.0001122**
Sparse/Short	D-TPTS	0.7	**−5.84 [−8.52, −3.30]**	−0.5	2.7×10^{-5}	**0.000133**
Sparse/Long	D-TP	1	**−10.36 [−12.96, −7.32]**	−0.85	2.3×10^{-13}	1.8×10^{-12}
Sparse/Long	D-TPTS	1	**−10.73 [−13.24, −7.80]**	−0.84	5.3×10^{-13}	3.7×10^{-12}

7.6. Mechanism Diagnostics: Assignable-Set Availability and Planning Detours

Figure 8 visualizes the mechanism telemetry defined in Section 6.4 as paired mean differences between best-θ and $\theta = \infty$ (100 seeds). Across all scenario–algorithm pairs, best-θ reduces the *no-assignable rate* (token turns with candidate tasks but no assignable candidates), with especially large and consistent reductions for D-TPTS in Dense/Long and in sparse regimes. Endpoint blocking at the candidate-filter level also drops substantially in sparse regimes (blocked fraction reductions of 0.08–0.31), consistent with the interpretation that deferring low-quality commitments can preserve assignable-set availability under endpoint occupancy constraints. Blocking is not uniformly reduced in the most time-critical dense setting: Dense/Short with D-TP shows a small increase in blocked fraction despite improvements in tardiness. Finally, the *detour ratio* typically decreases for D-TP (and for Sparse/Long with D-TPTS), indicating modest reductions in reservation- and congestion-driven path inflation at best-θ. Taken together, these diagnostics support the mechanism interpretation used throughout the paper: deferring a low-quality commitment can preserve assignable-set availability for later urgent tasks and can reduce downstream routing friction caused by reservations and endpoint blocking.

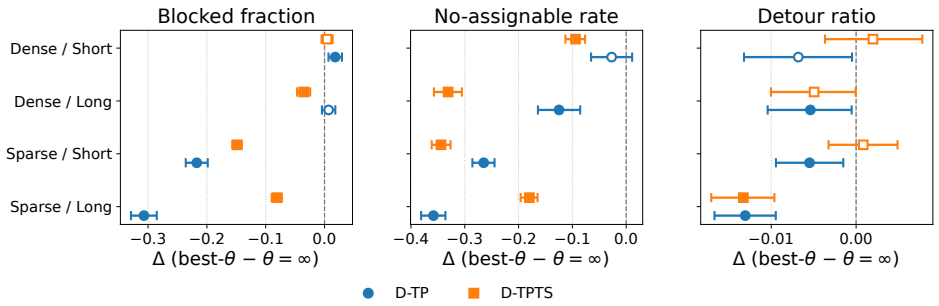


Figure 8. Mechanism diagnostics at best-θ: paired mean differences (best-$\theta - \theta = \infty$) with 95% bootstrap CIs (100 paired seeds). The three panels report blocked-fraction change, no-assignable-rate change, and detour-ratio change. Filled markers indicate raw two-sided Wilcoxon $p < 0.05$ (descriptive), open markers indicate non-significant differences at raw $p \geq 0.05$, and dashed vertical lines indicate zero difference from the always-assign baseline. Negative Δ indicates reduced blocking/no-assignable turns and reduced detours.

Supplementary analyses.

Additional analyses (run validity, comparator policies, transfer experiments, ablations, and large-map generalization) are provided in Appendix A.

8. Discussion

The experiments consistently reveal a *commitment-timing trade-off*: intermediate thresholds improve tardiness by preserving flexibility when immediate commitment is low-value, whereas very small θ over-defers and very large θ approaches the always-assign baseline. In this sense, DAD-θ is best interpreted not simply as another dispatching heuristic, but as a lightweight mechanism for controlling the timing of commitment in a shared, reservation-constrained multi-agent system.

Normalized scoring (Equation (9)) makes θ dimensionless and improves portability; transfer and large-map generalization show frequent gains versus always-assign, while target-side tuning remains best. Seed-split holdout validation indicates that θ selected on training seeds generalizes in tardiness, whereas makespan gains are less robust in the most time-critical setting. Comparator policies and the safety ablation further suggest that gains are specific to thresholded deferral and that the safety override mainly acts as a validity guardrail under aggressive deferral. Although DAD-θ is tuned for tardiness, we often observe concurrent makespan reductions. We interpret this as consistent with reduced endpoint blocking and reservation conflicts under better-timed commitments, but we use makespan only as an additional evaluation metric and do not claim a general guarantee. From a service-level perspective, the accompanying gains in on-time completion and reductions in MaxTard indicate that the method is not merely shifting delay among tasks: in the tested settings, the intermediate-threshold regime also improves deadline adherence and tail behavior. This is important operationally because warehouse operators often care about missed-deadline frequency and worst-case tardiness in addition to aggregate tardiness.

8.1. Systems-Level Implications and Deployment Considerations

From a systems perspective (see Figure 1), DAD-θ provides an operator-facing tuning knob for balancing deadline compliance against other operational objectives (e.g., makespan/throughput) within a larger warehouse coordination stack that uses a TP-family task allocator. Normalized scoring makes θ dimensionless and supports transfer across related regimes, which is useful when only map scale or deadline scale changes; however, as shown by our transfer results, target-side tuning remains best for tardiness. Importantly, the commit-or-defer decision can have system-level consequences because commitments interact through shared-space constraints (congestion, endpoint occupancy, and reservation conflicts), creating feedback loops that shape system-level tardiness.

Emergent Behavior and Feedback Loops.

In Token Passing, committing to a task reserves a path and occupies an endpoint in the shared token; these reservations directly affect which tasks are assignable to other agents (the assignable set in Equation (5)). Deferring a low-quality commitment can therefore have *indirect* benefits: it can keep endpoints and reservations available for imminent urgent tasks, reduce congestion, and improve the system's ability to route work to better-matched agents. These benefits are emergent in the sense that they arise from many local commit-or-wait decisions interacting through shared constraints. Consistent with this interpretation, best-θ reduces the rate of token turns that have candidate tasks but no assignable candidates, with additional reductions in endpoint blocking in sparse regimes and reductions in planning detours primarily for D-TP (Figure 8).

At the same time, excessive deferral can create starvation and instability. The safety override in Equation (14) acts as a guardrail: once an urgent task becomes assignable to the current token holder, DAD-θ must commit regardless of θ. In deployment, two low-cost telemetry signals—the DAD deferral rate and the safety-trigger rate—provide operational visibility into this feedback loop and can indicate when workload characteristics shift and re-tuning is warranted.

Calibration and Integration.

In practice, θ can be calibrated offline on historical or simulated order streams (e.g., in a digital twin) against explicit service-level targets such as tardiness, on-time completion, and tail risk, and then monitored online via low-cost telemetry. Section 4.8 summarizes a workflow based on threshold sweeps, margin-aware selection, and re-calibration triggers derived from deferral and safety-trigger rates. Because DAD-θ is implemented purely as a gate on top of an existing TP allocator, it can be introduced without changing endpoint-selection logic, reservation semantics, or downstream robot control, which makes it attractive as a drop-in operational control parameter in hierarchical warehouse coordination architectures.

Beyond Warehouses.

The same commitment–flexibility tension appears in other cyber-physical and service systems with online arrivals, such as dynamic vehicle routing and on-demand ride-sharing [29,30]. While DAD-θ is defined here as a gate on TP-family allocators, similar commit/deferral controls could be explored in other planning architectures; doing so would require architecture-specific definitions of feasibility constraints, candidate sets, and the deferral action.

8.2. Limitations and Future Work

Our two evaluated maps share the same obstacle-block motif, and the agent-to-task ratio remains close to 1:10 in both map conditions. We expect corridor-heavy layouts to amplify reservation externalities and endpoint contention, which could make the useful θ band narrower or shift it toward more conservative settings because a single premature commitment can block more downstream traffic. Conversely, more open layouts may weaken these effects. Likewise, at ratios closer to 1:20, scarce agents are likely to reduce tolerance for deferral because urgent tasks must be picked up promptly, whereas at ratios closer to 1:5, extra fleet slack may make DAD-style postponement function more as congestion smoothing than urgency triage. These expectations are plausible but untested here, which is why broader layout and load-ratio sweeps remain future work.

1. This study focuses on well-formed grid maps with static obstacles; broader map diversity (e.g., irregular layouts, corridor-heavy designs, or non-grid topologies) may change congestion dynamics and could introduce multiple local optima in the θ landscape.
2. The formulation assumes homogeneous agents and homogeneous task service requirements apart from release times, locations, and deadlines; extending the commit-or-defer idea to heterogeneous capabilities, loading times, or task classes is future work.
3. Although we evaluate two map scales and two load scales, the agent-to-task ratio remains similar (approximately 15:151 and 30:304, i.e., $\approx$1:10); future work should vary this ratio to test high-congestion and extreme-scarcity regimes.
4. Deadlines are sampled from uniform offset intervals; non-uniform or correlated deadline structures (e.g., priority orders or batch deadlines) may affect θ selection and the efficacy of normalization.

5. While DAD-θ has negligible overhead in our measured settings, broader wall-clock scalability studies under larger fleets and task pools (e.g., 100+ agents) are needed.
6. We focus on lightweight TP-compatible baselines. Comparisons to methods with different assumptions and heavier computation budgets (e.g., global optimization, auctions, or learning-based allocators) are important, but they would introduce different candidate-set semantics, training requirements, or reservation models and are therefore left for future work.
7. While full-run success is empirical here, formal guarantees beyond the finite-deadline benchmark settings studied in this paper are future work.
8. Our holdout validation focuses on selection of θ conditional on fixed baseline tuning ($\alpha^{\langle * \rangle}$); future work should evaluate generalization of joint (α, θ) selection under nested validation.
9. Fixed θ still requires scenario-level tuning for best tardiness, motivating lightweight adaptive threshold control as future work.

9. Conclusions

We presented DAD-θ, a minimal and explainable mechanism that turns *commitment timing* into an explicit control variable for deadline-aware TP-family MAPD-D. DAD-θ adds one threshold gate with a parameter-free safety override near pickup deadlines, and our normalized scoring makes threshold values dimensionless. Under strict paired experiments with 100% valid runs (101,600 runs in total), DAD-θ reduced cumulative tardiness in all tested scenario–algorithm pairs in both benchmark and larger-map settings. Seed-split holdout validation on the benchmark further confirms that selecting θ on training seeds generalizes in tardiness to held-out seeds. We emphasize that inferential claims for best-θ comparisons are supported primarily by the seed-split holdout validation, while same-seed best-θ paired summaries are reported as descriptive post-selection statistics.

Best-θ also improved makespan in mean on all benchmarks and additional large-map pairs, though makespan gains are less robust under seed-split holdout for Dense/Short with D-TPTS. Paired significance tests with Holm correction and comparator-policy experiments show that the benefit is specific to thresholded deferral, not to arbitrary defer patterns. Transfer experiments further show frequent improvements over always-assign without target-side re-tuning, while target-tuned thresholds remain best for tardiness. Additional ablations show that (i) the safety override improves validity under aggressive deferral, and (ii) score normalization improves transfer robustness relative to raw scoring. Overall, DAD-θ offers a lightweight, operator-tunable commitment-control mechanism for deadline-aware Token Passing in logistics systems: it can be calibrated offline against explicit service-level targets, monitored online through simple telemetry, and deployed without modifying the underlying planner or robot-control stack. The main current limitations are the homogeneous well-formed TP-family setting, finite-task benchmark windows, and comparison scope restricted to TP-compatible baselines, which together define the next steps for broader heterogeneous and cross-architecture evaluation.

Author Contributions: T.H. performed conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing—original draft preparation, and visualization. K.Y., H.S. and I.N. provided advice throughout the research and contributed to writing—review and editing. H.S. and I.N. supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request. The source code, Docker configuration, and scripts used to

generate the results will also be made available by the authors on request. The code was run in a Docker-based Python 3.11 environment. The main analysis dependencies include NumPy 1.24.3, pandas 2.0.3, SciPy 1.11.3, scikit-learn 1.3.0, Matplotlib 3.7.2, and NetworkX 3.3.

Conflicts of Interest: Author Kohei Yoshida was employed by Toyota Industries Corporation. A patent application related to the method described in this manuscript has been filed. The authors declare no other commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Supplementary Results

This appendix collects secondary results referenced in the main text.

Appendix A.1. Run Validity

All runs in the main benchmark and additional large-map protocols satisfy the strict success conditions above (100% validity). Disabling the safety override can reduce validity under aggressive deferral (Appendix A.5).

Appendix A.2. Baseline α Selection

Baseline $\alpha^{\langle * \rangle}$ values are selected under $\theta = \infty$ in phase 1 of the main protocol (Section 6.3) and reported (as α^*) in Table 1. We then fix $\alpha^{\langle * \rangle}$ for the subsequent θ sweep and holdout validation.

Appendix A.3. Comparator Deferral Policies

The policy sweep executes 13 settings per scenario–algorithm pair (6 threshold, 3 random, 4 periodic). For comparator analysis, we focus on the 56 non-threshold settings (random and periodic only) under the same candidate set, scoring, safety override, and fallback behavior as thresholded DAD-θ; only the deferral trigger differs:

- **Tardiness:** 18/56 settings improve mean tardiness over always-assign; 5/56 are significant at raw $p < 0.05$, and 0/56 remain significant after Holm correction.
- **Makespan:** 19/56 settings improve mean makespan; 7/56 are significant at raw $p < 0.05$, and 0/56 remain significant after Holm correction.

Thus, naive defer patterns are not competitive with thresholded DAD-θ for tardiness optimization.

Appendix A.4. θ Transferability

In transfer experiments (32 source–target–algorithm combinations), source-selected $\theta^{\langle * \rangle}$ is applied to targets without re-tuning.

- **Against always-assign** ($\theta = \infty$ on target): transfer improves tardiness in mean in 30/32 combinations and improves makespan in mean in 30/32. After Holm correction, 30/32 (tardiness) and 15/32 (makespan) remain significant improvements, with 1/32 Holm-significant degradation in each metric.
- **Against target-tuned best-θ:** transfer never improves tardiness (0/32), and 5/32 cases are Holm-significantly worse. Transfer improves makespan in mean in 4/32 combinations, but none of these gains are significant after Holm correction; 2/32 are Holm-significantly worse.

For improvements versus always-assign, this pattern holds for three source scenarios (8/8 target pairs each), while Sparse/Short is weaker (6/8).

Appendix A.5. Ablation: Safety Override

The safety override in Equation (14) forces assignment when any assignable task reaches non-positive pickup slack. To isolate its contribution, we rerun the normalized θ sweep at $\theta \in \{\infty, 0.7, 0.9, 1\}$ with the safety override disabled on the same pre-generated workloads. Table A1 compares safety ON/OFF at the best-θ values from the main protocol. Completion remains 100% at these tuned $\theta^{\langle * \rangle}$ values, and mean tardiness/makespan remain comparable between safety ON/OFF (with some pairs slightly better without safety), indicating that the best-θ improvements are not solely due to the safety override. However, disabling safety can reduce validity under aggressive deferral: for Sparse/Short with D-TP and $\alpha^{\langle * \rangle} = 0$, only 75/100 runs complete at $\theta = 0.7$ (stall timeout). Safety trigger rates in the benchmark range from 1.9% to 48.6% of assignment decisions depending on scenario and algorithm (Table A1).

Table A1. Safety override ablation on the benchmark. "Safety (on)" is the mean fraction of assignment decisions where the safety condition (non-positive pickup slack for some assignable task) is met under safety ON.

Scenario	Algo	θ^*	ok (on)	ok (off)	Tard (on)	Tard (off)	Mk (on)	Mk (off)	Safety (on)
Dense/Short	D-TP	1	100/100	100/100	1290.8	1192.3	368.8	366.8	48.6
Dense/Short	D-TPTS	0.9	100/100	100/100	1270.7	1155.5	371.5	368.5	42.3
Dense/Long	D-TP	1	100/100	100/100	86.9	88.3	377.2	377.2	5.5
Dense/Long	D-TPTS	1	100/100	100/100	88.2	89.6	377.3	377.2	5.5
Sparse/Short	D-TP	1	100/100	100/100	603.3	470.6	547.9	542.8	5.8
Sparse/Short	D-TPTS	0.7	100/100	100/100	565.9	473.9	547.8	544.8	2.5
Sparse/Long	D-TP	1	100/100	100/100	16.7	11.0	545.3	543.4	1.9
Sparse/Long	D-TPTS	1	100/100	100/100	17.3	11.6	545.6	543.3	1.9

To illustrate the safety override as a validity guardrail, we also run an additional safety-OFF sweep on the most sensitive case (Sparse/Short) over $\theta \in [-1, 2]$. For Sparse/Short with D-TP, completion reaches 100% only for $\theta \geq 1.4$ when safety is OFF, whereas D-TPTS remains at 100% across the grid.

Appendix A.6. Ablation: Raw vs. Normalized Scoring

Normalized scoring (Equation (9)) is designed to make θ dimensionless and portable. We compare it against raw scoring by rerunning the benchmark protocol in raw mode (separate α sweep, θ sweep, and θ transfer on the same workloads; all raw-mode runs satisfy the success conditions). In raw mode we sweep $\theta \in \{0, 5, 10, 15, 18, 20, 25, 30, 35, 40\}$ (plus $\theta = \infty$ baseline). Table A2 shows that raw optimal thresholds are an order of magnitude larger (in our benchmark, $\theta^{\langle * \rangle}_{\text{raw}} \in [18, 25]$, i.e., interior points of the raw grid) and vary more across scenarios than normalized $\theta^{\langle * \rangle}$, which concentrates in $[0.7, 1]$. We use this coarse grid (mostly 5-step spacing, with a few additional values near 20) to resolve the typical raw-threshold regime in our benchmark without incurring a large sweep cost. Transfer robustness is also stronger with normalization (Table A3): against $\theta = \infty$, normalized transfer improves mean tardiness in 30/32 combinations (Holm: +30/−1) versus 27/32 (Holm: +25/−3) in raw mode, and it yields fewer Holm-significant makespan degradations (-1 versus -5).

Table A2. Raw vs. normalized best-θ portability on the benchmark. Bold indicates normalized θ^*_{norm}.

Scenario	Algo	θ^*_{raw}	θ^*_{norm}
Dense/Short	D-TP	20	**1**
Dense/Short	D-TPTS	18	**0.9**
Dense/Long	D-TP	25	**1**
Dense/Long	D-TPTS	25	**1**
Sparse/Short	D-TP	18	**1**
Sparse/Short	D-TPTS	18	**0.7**
Sparse/Long	D-TP	20	**1**
Sparse/Long	D-TPTS	25	**1**

Table A3. Raw vs. normalized transfer robustness summary on the benchmark (32 source–target–algorithm combinations). "Holm (+/−)" counts Holm-significant improvements/degradations versus $\theta = \infty$. Bold indicates the normalized setting.

Metric (vs. $\theta = \infty$)	Mean Improve	Mean Worsen	Holm (+/−)	Mode
Tardiness	27/32	5/32	+25/−3	raw
Tardiness	**30/32**	**2/32**	**+30/−1**	**normalized**
Makespan	25/32	7/32	+19/−5	raw
Makespan	**30/32**	**2/32**	**+15/−1**	**normalized**

Appendix A.7. Additional Map/Scale Generalization on a Larger Map

We next evaluate the larger 48×35 map under C1 (map-only: 15 agents, 151 tasks) and C2 (map+load: 30 agents, 304 tasks). Both conditions use the same four scenarios and normalized scoring, with $\alpha^{\langle * \rangle}$ transferred from the benchmark setting. The summaries below are computed from complete C1/C2 run sets (25600 θ-sweep runs and 3200 transfer runs per condition), not from sampled subsets.

At best-θ versus $\theta = \infty$:

- **C1:** tardiness improves in 8/8 pairs (raw 8/8 significant; Holm 8/8). Makespan improves in mean in 8/8 pairs (raw 7/8 significant; Holm 7/8). Tardiness reduction ranges from 5.8% to 47.9%.
- **C2:** tardiness improves in 8/8 pairs (raw 8/8 significant; Holm 8/8). Makespan improves in mean in 8/8 pairs (raw 6/8 significant; Holm 5/8). Tardiness reduction ranges from 1.8% to 9.7%.

DAD-θ remains active in most finite-θ runs in both conditions.

For transfer on the larger map (32 source–target–algorithm combinations per condition):

- **Against always-assign:** tardiness improves in mean in 32/32 (C1) and 32/32 (C2), and makespan improves in mean in 32/32 (C1) and 28/32 (C2). After Holm correction, significant improvements are 24/32 (tardiness) and 10/32 (makespan) for C1 and 26/32 (tardiness) and 12/32 (makespan) for C2, with zero Holm-significant degradations in both conditions.
- **Against target-tuned best-θ:** transfer is never better in tardiness (0/32 for both C1 and C2) and is Holm-significantly worse in 6/32 for both. For makespan, no Holm-significant improvements occur; 6/32 are Holm-significantly worse in C1 and 2/32 in C2.

These large-map results confirm robust gains versus always-assign while preserving the expected gap to target-side θ tuning. Figure A1 visualizes per-pair best-θ gains for C1/C2.

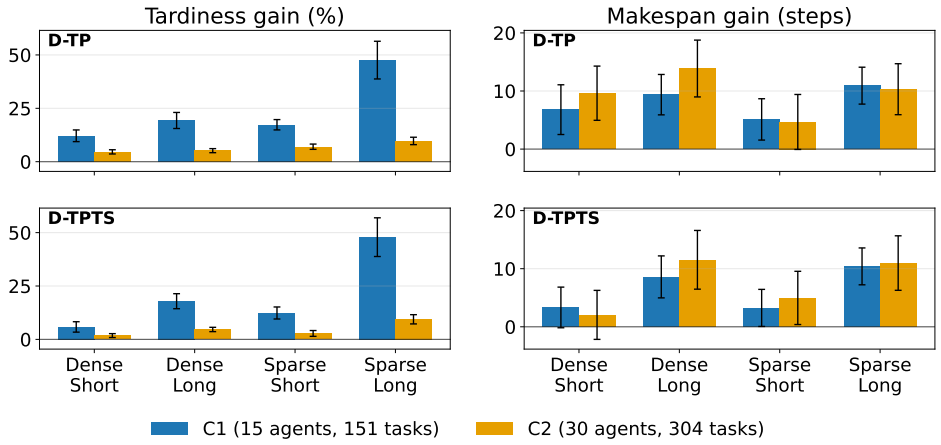


Figure A1. Additional large-map best-θ gains for C1 (map-only) and C2 (map+load). **Left**: tardiness gain (%). **Right**: makespan gain (steps). Positive values indicate improvement over $\theta = \infty$. Bars show mean $\pm$ 95% CI over 100 paired seeds.

Appendix B. One-Step Surrogate Details

This appendix records the formal surrogate definitions underlying the qualitative summary in Section 5.

Appendix B.1. One-Step Intervention Surrogate

Intuitively, deferring at a decision point incurs an immediate one-step delay cost, but it can be beneficial if waiting allows a better task to become assignable (e.g., due to new releases or token-reservation evolution). Our surrogate isolates this marginal trade-off by comparing forced *defer* versus forced *assign* at a single baseline decision point, while following the baseline policy thereafter. Let π_∞ denote the baseline always-assign policy ($\theta = \infty$ in Equation (13)) with the same safety override (Equation (14)). Consider a *decision point* under π_∞ where (i) at least one task is assignable ($\mathcal{T}'_{i,t} \neq \emptyset$) and (ii) the safety override is inactive. Let S be the best candidate score at that decision point, i.e., $S = s_i^{\langle * \rangle}(t)$ from Equation (12).

Throughout this appendix, X denotes the eventual run-level metric value; when we write $X(\cdot)$ with a policy argument (e.g., $X(\infty)$), it denotes the corresponding expected metric value under that policy. For a performance metric X (we use T and M as defined in Equations (3) and (4)), define the one-step defer advantage at score s as

$$D_X(s) = \mathbb{E}\Big[X^{\langle \text{defer} \rangle} - X^{\langle \text{assign} \rangle} \mid S = s; \pi_\infty\Big], \tag{A1}$$

where $X^{\langle \text{defer} \rangle}$ (resp. $X^{\langle \text{assign} \rangle}$) is the eventual metric value if we force *defer* (resp. *assign*) at that decision point and then follow π_∞ afterwards. Thus, $D_X(s) < 0$ means that deferring at score s is beneficial for metric X under this one-step intervention view. In our discrete-time experiments (integer shortest-path distances and integer-valued deadlines/offsets), $h(\cdot,\cdot)$ and $(d_j^{\langle p \rangle} - t)$ are integer-valued and $\tilde{s}_{i,j}(t)$ is a deterministic rational rescaling; therefore, S takes values on a discrete grid at decision points, and Equation (A1) can be read as an ordinary conditional expectation. For continuous variants (or real-valued deadlines), $D_X(s)$ is interpreted via a regular conditional expectation.

For a run under π_∞, let N_{dp} be the number of safety-inactive decision points (i.e., token turns with $\mathcal{T}'_{i,t} \neq \emptyset$ where the safety override is inactive) and index them by $k = 1, \ldots, N_{\text{dp}}$

in temporal order. Let S_k be the best candidate score at decision point k (so S_k takes values on the same grid as S). We define the one-step intervention surrogate for threshold θ as

$$\tilde{X}(\theta) = X(\infty) + \mathbb{E}\left[\sum_{k=1}^{N_{\mathrm{dp}}} D_X(S_k)\,\mathbf{1}\{S_k > \theta\}\right]. \tag{A2}$$

where $X(\infty)$ denotes the expected metric value under π_∞. Equivalently, for any $\theta_1 < \theta_2$,

$$\tilde{X}(\theta_1) - \tilde{X}(\theta_2) = \mathbb{E}\left[\sum_{k=1}^{N_{\mathrm{dp}}} D_X(S_k)\,\mathbf{1}\{\theta_1 < S_k \le \theta_2\}\right]. \tag{A3}$$

Remark A1 (Surrogate versus full policy)**.** *In general, the actual policy performance $X(\theta)$ need not match the surrogate $\tilde{X}(\theta)$, because changing θ affects later states, future score distributions, and interactions among repeated deferrals. We therefore use the surrogate only as a qualitative explanatory lens and treat the observed empirical sweep shapes as the main check on its relevance.*

Appendix B.2. Single-Crossing Assumption and Implications

Assumption A1 (Single-Crossing)**.** *For each $X \in \{T, M\}$, assume the conditional one-step defer advantage $D_X(s)$ is non-increasing in the score s (deferring becomes weakly less harmful as the best currently assignable option worsens). Define the* crossing interval *boundary points*

$$s_X^- := \sup\{s \in \mathbb{R} : D_X(s) > 0\}, \qquad s_X^+ := \inf\{s \in \mathbb{R} : D_X(s) < 0\}, \tag{A4}$$

with the convention $\sup \emptyset = -\infty$ *and* $\inf \emptyset = +\infty$*. Under monotonicity, these satisfy $s_X^- \le s_X^+$.*

$$D_X(s) > 0 \text{ for } s < s_X^- \quad \text{and} \quad D_X(s) < 0 \text{ for } s > s_X^+. \tag{A5}$$

Because $D_X(s)$ need not be continuous (especially on discrete score grids), its value at the boundary points s_X^- and s_X^+ can have either sign; therefore Equation (A5) is stated with strict inequalities (off the boundary). In particular, in discrete/grid settings $D_X(s)$ can be exactly zero on a nontrivial crossing interval $s \in (s_X^-, s_X^+)$. Intuitively, deferring is harmful when the best available option is already good (low s) but can be beneficial when the best available option is less attractive under the baseline score (high s). The term "single-crossing" is used in monotone comparative statics [38]; here we adopt it as a sufficient-condition assumption on the conditional one-step defer advantage $D_X(s)$.

Toy Example (Waiting Value).

A simple motivation for monotonicity is to decompose the one-step defer advantage as $D_X(s) = H_X - W_X(s)$, where $H_X > 0$ is the one-step delay penalty and $W_X(s) \ge 0$ is the expected value of waiting. If waiting becomes more valuable when the current best option is worse (i.e., $W_X(s)$ is non-decreasing in s), then $D_X(s)$ is non-increasing, yielding the single-crossing structure in Equation (A5).

Proposition A1 (Interpretive single-valley surrogate)**.** *Under the single-crossing assumption for metric X, the surrogate $\tilde{X}(\theta)$ is non-increasing for $\theta < s_X^-$ and non-decreasing for $\theta \ge s_X^+$, i.e., it exhibits a single-valley (stepwise, possibly with a flat minimum interval) dependence on θ. We emphasize that this statement concerns the one-step surrogate, not the full deterministic multi-step DAD-θ policy.*

Proof. Follows directly from Equation (A3) and the sign pattern in Equation (A5). □

Definition A1 (Jointly Improving Threshold; Surrogate)**.** *We call a threshold* θ jointly improving *for the surrogate if it improves both metrics relative to the baseline:* $\tilde{T}(\theta) < T(\infty)$ *and* $\tilde{M}(\theta) < M(\infty)$.

The above analysis is consistent with two empirical findings: (i) θ sweeps often show a single-valley sensitivity curve (Figures 6 and 7), and (ii) naive deferral patterns (random/periodic) are not reliably beneficial, because they can defer even in low-score regimes where $D_X(s) \geq 0$ (Appendix A.3).

Remark A2 (How to Empirically Probe Single-Crossing)**.** *Although counterfactual,* $D_X(s)$ *could be estimated via paired assign/defer interventions on bin-matched safety-inactive decision points under our common-random-numbers protocol.*

Appendix C. Deterministic Construction of $\mathcal{Q}$ for $\bar{h}$

To estimate the mean endpoint distance $\bar{h}$ in Equation (11) without using random numbers or future workload information, we construct $\mathcal{Q}$ deterministically from the map topology. Let $(e_0, \ldots, e_{|\mathcal{E}|-1})$ be the task endpoints in a fixed order (row-major by coordinates in our grid implementation). We use $N_p = \min\{16, |\mathcal{E}|\}$ anchors and set $L = \lceil 2000/N_p \rceil$. We cap $|\mathcal{Q}|$ at 2000; multiple anchors spread the sampled pairs across the endpoint set, while L targets a roughly even per-anchor contribution. The fixed prime 997 defines a deterministic anchor-specific starting index, and subsequent striding by s fills in additional pairs. For each $k \in \{0, \ldots, N_p - 1\}$, choose an anchor $u_k = e_{\lfloor k|\mathcal{E}|/N_p \rfloor}$ and then generate candidate pair indices by cycling through

$$j \leftarrow (997k) \bmod |\mathcal{E}|, \qquad j \leftarrow (j+s) \bmod |\mathcal{E}|, \qquad s = \max\{1, \lfloor |\mathcal{E}|/L \rfloor\},$$

visiting at most $|\mathcal{E}|$ indices per anchor and adding the first L pairs (u_k, e_j) with $e_j \neq u_k$ (skipping unreachable pairs), stopping early if 2000 pairs have been collected.

References

1. Wurman, P.R.; D'Andrea, R.; Mountz, M. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Mag.* **2008**, *29*, 9–20.
2. Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.K.S.; et al. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2019; Volume 10, pp. 151–158.
3. Salzman, O.; Stern, R. Research Challenges and Opportunities in Multi-Agent Path Finding and Multi-Agent Pickup and Delivery Problems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*; The Association for Computing Machinery (ACM): New York, NY, USA, 2020; pp. 1711–1715.
4. Ma, H.; Li, J.; Kumar, T.K.S.; Koenig, S. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*; The Association for Computing Machinery (ACM): New York, NY, USA, 2017; pp. 837–845.
5. Liu, M.; Ma, H.; Li, J.; Koenig, S. Task and Path Planning for Multi-Agent Pickup and Delivery. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*; The Association for Computing Machinery (ACM): New York, NY, USA, 2019; pp. 1152–1160.
6. Ma, H.; Hönig, W.; Kumar, T.K.S.; Ayanian, N.; Koenig, S. Lifelong Path Planning with Kinematic Constraints for Multi-Agent Pickup and Delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*; The Association for Computing Machinery (ACM): New York, NY, USA, 2019; pp. 7651–7658.
7. Van Mieghem, J.A.; Dada, M. Price Versus Production Postponement: Capacity and Competition. *Manag. Sci.* **1999**, *45*, 1639–1649. [CrossRef]
8. Dixit, A.K.; Pindyck, R.S. *Investment Under Uncertainty*; Princeton University Press: Princeton, NJ, USA, 1994.
9. Makino, H.; Ito, S. Online Multi-Agent Pickup and Delivery with Task Deadlines. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; The Association for Computing Machinery (ACM): New York, NY, USA, 2024; pp. 8428–8434.

10. Mitrović-Minić, S.; Laporte, G. Waiting Strategies for the Dynamic Pickup and Delivery Problem with Time Windows. *Transp. Res. Part B Methodol.* **2004**, *38*, 635–655. [CrossRef]
11. Branke, J.; Middendorf, M.; Noeth, G.; Dessouky, M. Waiting Strategies for Dynamic Vehicle Routing. *Transp. Sci.* **2005**, *39*, 298–312. [CrossRef]
12. Hirayama, T.; Yoshida, K.; Sakaji, H.; Noda, I. Efficient Assignment of Immediate Tasks Using Deep Reinforcement Learning in Multi-Agent Pickup and Delivery. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Computational Social Science: The PAAMS Collection*; Mathieu, P., De la Prieta, F., Eds.; PAAMS Workshop 2025; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2026; Volume 16031. [CrossRef]
13. Li, J.; Tinka, A.; Kiesel, S.; Durham, J.W.; Kumar, T.K.S.; Koenig, S. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*; The Association for Computing Machinery (ACM): New York, NY, USA, 2021; pp. 11272–11281.
14. Okumura, K. Improving LaCAM for Scalable Eventually Optimal Multi-Agent Pathfinding. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*; The Association for Computing Machinery (ACM): New York, NY, USA, 2023; pp. 243–251.
15. Liang, R.; Veerapaneni, R.; Harabor, D.; Li, J.; Likhachev, M. Real-Time LaCAM for Real-Time MAPF. In *Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2025; Volume 18, pp. 196–200.
16. Zhang, Y.; Jiang, H.; Bhatt, V.; Nikolaidis, S.; Li, J. Guidance Graph Optimization for Lifelong Multi-Agent Path Finding. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*; The Association for Computing Machinery (ACM): New York, NY, USA, 2024; pp. 311–320.
17. Gao, J.; Li, Y.; Li, X.; Yan, K.; Lin, K.; Wu, X. A Review of Graph-Based Multi-Agent Pathfinding Solvers: From Classical to Beyond Classical. *Knowl.-Based Syst.* **2024**, *283*, 111121. [CrossRef]
18. Alkazzi, J.-M.; Okumura, K. A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding. *IEEE Access* **2024**, *12*, 57390–57409. [CrossRef]
19. Flammini, B.; Azzalini, D.; Amigoni, F. Preventing Deadlocks for Multi-Agent Pickup and Delivery in Dynamic Environments. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*; The Association for Computing Machinery (ACM): New York, NY, USA, 2024; pp. 580–588.
20. Bonalumi, L.; Flammini, B.; Azzalini, D.; Amigoni, F. Multi-Agent Pickup and Delivery with External Agents. *Robot. Auton. Syst.* **2025**, *191*, 105000. [CrossRef]
21. Çilden, E.; Polat, F. Task Assignment Strategies for Capacitated Agents Engaged in Lifelong Pickup and Delivery Tasks. *Knowl.-Based Syst.* **2025**, *329*, 114281. [CrossRef]
22. Chen, Z.; Alonso-Mora, J.; Bai, X.; Harabor, D.D.; Stuckey, P.J. Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5816–5823. [CrossRef]
23. Ma, H.; Wagner, G.; Felner, A.; Li, J.; Kumar, T.K.S.; Koenig, S. Multi-Agent Path Finding with Deadlines. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*; The Association for Computing Machinery (ACM): New York, NY, USA, 2018; pp. 417–423.
24. Wang, H.; Chen, W. Multi-Robot Path Planning with Due Times. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4829–4836. [CrossRef]
25. Huang, T.; Shivashankar, V.; Caldara, M.; Durham, J.; Li, J.; Dilkina, B.; Koenig, S. Deadline-Aware Multi-Agent Tour Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*; The Association for Computing Machinery (ACM): New York, NY, USA, 2023; pp. 189–197.
26. Wu, X.; Liu, Y.; Tang, X.; Cai, W.; Bai, F.; Khonstantine, G.; Zhao, G. Multi-Agent Pickup and Delivery with Task Deadlines. In *Proceedings of the International Symposium on Combinatorial Search*; The Association for Computing Machinery (ACM): New York, NY, USA, 2021; pp. 206–208.
27. Ramanathan, S.; Liu, Y.; Tang, X.; Cai, W.; Li, J. Minimising Task Tardiness for Multi-Agent Pickup and Delivery. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*; The Association for Computing Machinery (ACM): New York, NY, USA, 2023; pp. 2349–2351.
28. Berbeglia, G.; Cordeau, J.-F.; Laporte, G. Dynamic Pickup and Delivery Problems. *Eur. J. Oper. Res.* **2010**, *202*, 8–15. [CrossRef]
29. Pillac, V.; Gendreau, M.; Guéret, C.; Medaglia, A.L. A Review of Dynamic Vehicle Routing Problems. *Eur. J. Oper. Res.* **2013**, *225*, 1–11. [CrossRef]
30. Alonso-Mora, J.; Samaranayake, S.; Wallar, A.; Frazzoli, E.; Rus, D. On-Demand High-Capacity Ride-Sharing via Dynamic Trip-Vehicle Assignment. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 462–467. [CrossRef] [PubMed]
31. He, C.; Duhan, T.; Tulsyan, P.; Kim, P.; Sartoretti, G. Social Behavior as a Key to Learning-Based Multi-Agent Pathfinding Dilemmas. *Artif. Intell.* **2025**, *348*, 104397. [CrossRef]
32. Vepsalainen, A.P.J.; Morton, T.E. Priority Rules for Job Shops with Weighted Tardiness Costs. *Manag. Sci.* **1987**, *33*, 1035–1047. [CrossRef]

33. Morton, T.E.; Pentico, D.W. *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*; Wiley: Hoboken, NJ, USA, 1993.
34. Tao, F.; Zhang, M. Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing. *IEEE Access* **2017**, *5*, 20418–20427. [CrossRef]
35. Kritzinger, W.; Karner, M.; Traar, G.; Henjes, J.; Sihn, W. Digital Twin in Manufacturing: A Categorical Literature Review and Classification. *IFAC-PapersOnLine* **2018**, *51*, 1016–1022. [CrossRef]
36. Čap, M.; Vokřinek, J.; Kleiner, A. Complete Decentralized Method for On-Line Multi-Robot Trajectory Planning in Well-Formed Infrastructures. In *Proceedings of the International Conference on Automated Planning and Scheduling*; The Association for Computing Machinery (ACM): New York, NY, USA, 2015; pp. 324–332.
37. Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
38. Milgrom, P.; Shannon, C. Monotone Comparative Statics. *Econometrica* **1994**, *62*, 157–180. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.