Online Multi-Agent Pickup and Delivery with Task Deadlines

Hiroya Makino[1,*], and Seigo Ito[1]

***Abstract*— Managing delivery deadlines in automated warehouses and factories is crucial for maintaining customer satisfaction and ensuring seamless production. This study introduces the problem of online multi-agent pickup and delivery with task deadlines (MAPD-D), which is an advanced variant of the online MAPD problem incorporating delivery deadlines. MAPD-D presents a dynamic deadline-driven approach that includes task deadlines, with tasks being added at any time (online), thus challenging conventional MAPD frameworks. To tackle MAPD-D, we propose a novel algorithm named deadline-aware token passing (D-TP). The D-TP algorithm is designed to calculate pickup deadlines and assign tasks while balancing execution cost and deadline proximity. Additionally, we introduce the D-TP with task swaps (D-TPTS) method to further reduce task tardiness, enhancing flexibility and efficiency via task-swapping strategies. Numerical experiments were conducted in simulated warehouse environments to showcase the effectiveness of the proposed methods. Both D-TP and D-TPTS demonstrate significant reductions in task tardiness compared to existing methods, thereby contributing to efficient operations in automated warehouses and factories with delivery deadlines.**

I. Introduction

In the industrial automation landscape, the development of automated guided vehicles (AGVs) has revolutionized operational efficiencies. Enhancing multi-agent path finding (MAPF) to optimize the utilization of AGVs for more effective transportation solutions has been extensively researched [1], [2]. These advancements have been integrated into various domains, including logistics automation [3], [4], traffic control systems [5], automated valet parking [6], airport surface operations [7], and video games [8].

The multi-agent pickup and delivery (MAPD) problem is an extension of MAPF, wherein paths for multiple items are planned from pickup locations to delivery locations [9], [10], [11]. MAPD can be applied to various environments, including automated warehouses and factories [9], [12], [13]. In such settings, the importance of managing delivery deadlines cannot be overstated. Warehouses must tailor deadlines for individual orders to ensure customer satisfaction, whereas factories require timely deliveries to maintain seamless production. Therefore, satisfying the deadlines appropriately is essential for operational efficiency and economic success.

To consider deadlines in MAPD, we define the online multi-agent pickup and delivery with task deadlines (MAPD-D) as a new problem (Fig. 1). Existing studies [14], [15] have explored MAPD with deadlines in an *offline* setting, where all task information is provided in advance. However, to the best of our knowledge, no study has addressed deadline-aware MAPD in an *online* setting where tasks may be added at any time. Hence, this study introduces new problem definitions for MAPD-D.

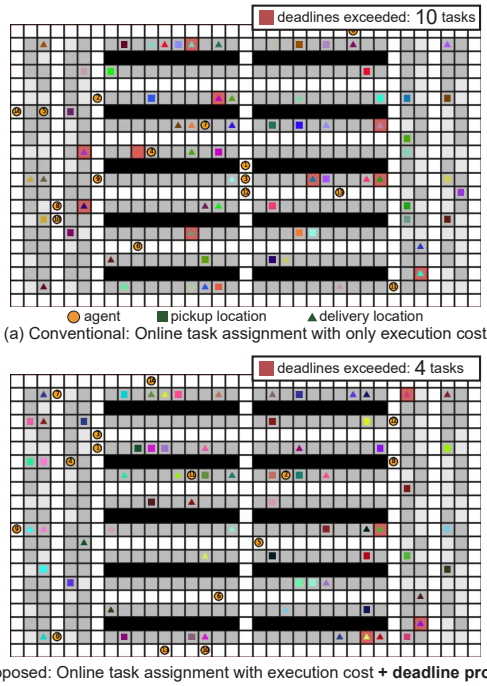


Fig. 1. Simulation examples in a four-neighbor grid environment [9]. The black cells represent blocked areas; dark gray cells indicate task endpoints; and light gray cells denote non-task endpoints. Each agent searches for a path from the pickup location to the delivery location for the assigned task. Unlike MAPD [9], a deadline is set for each task in MAPD-D.

We propose the deadline-aware token passing (D-TP) algorithm to tackle MAPD-D. This algorithm is designed to calculate pickup deadlines and assign tasks while striking a balance between execution cost and deadline proximity. Additionally, we introduce the D-TP with task swaps (D-TPTS) method to reduce task tardiness. D-TPTS enhances flexibility and efficiency by employing task-swapping strategies among agents and within a single agent.

The primary contributions of this study can be summarized as follows:

- A new problem (MAPD-D) is defined considering delivery deadlines and the possibility of adding tasks at any time.
- A method for solving MAPD-D by calculating pickup deadlines and deadline-aware task assignments is pro-

[1] H. Makino and S. Ito are with the Toyota Central R&D Labs., Inc., 41-1, Yokomichi, Nagakute, Aichi, Japan.

[*] Corresponding author. hirom@mosk.tytlabs.co.jp

posed.
- Task-swapping methods are introduced to reduce task tardiness.

The remainder of this paper is structured as follows. Section II describes the related work on MAPD. Section III defines the MAPD-D problem, and Section IV describes the proposed algorithm for solving this problem. Section V presents the numerical experiments performed to evaluate the proposed method. Finally, Section VI summarizes the study findings and concludes the paper.

II. Related Work

MAPF is the problem of moving multiple agents to their respective destination locations without collisions. The MAPF task ends when all agents reach their destinations. In contrast, MAPD requires agents to attend to a stream of delivery tasks [9], [10], wherein each delivery task is assigned a starting point (pickup location) and a destination point (delivery location). The system assigns these tasks to agents, which then move to the delivery location via the pickup location. The agents receive new tasks after reaching the delivery location. Ma et al. [9], [10] described an *online* version of MAPD, in which tasks can be added to the set at any time. Conversely, Liu et al. [11] discussed an *offline* variant where tasks and their release times are generally predetermined.

Ma et al. [9] reported that online MAPD instances are solvable if they are well-formed. They introduced locations referred to as endpoints, where agents can remain without blocking other agents. Endpoints include all pickup and delivery locations along with the initial positions and designated parking locations. The pickup and delivery locations are referred to as task endpoints, whereas the other locations serve as non-task endpoints. An MAPD instance is well-formed if and only if (a) the number of tasks is finite, (b) the number of agents is not greater than non-task endpoints, and (c) a path exists between any two endpoints without traversing others.

To address online MAPD, Ma et al. [9] employed the token passing (TP) algorithm. Tokens are a type of shared memory that stores all agent paths as well as the task set and agent assignments. Agents sequentially access the token, are assigned tasks, and find paths without colliding with already reserved paths. The token is updated after identifying a path.

Several researchers have investigated MAPF and MAPD with deadlines. In this study, we classified related studies from the literature considering four perspectives (Table I). "Multi-task" indicates whether each agent is continuously assigned tasks; "Deadlines" denote whether each task is set with a deadline; "Individual release times" denote whether the release time of each task is the same; and "Online" indicate whether tasks are added at any time. As shown in Table I, Ma et al. [16], Wang and Chen [17], and Huang et al. [18] considered deadlines in the context of MAPF. Wu et al. [14] and Ramanathan et al. [15] introduced deadlines in MAPD. However, the perspective of online tasks in MAPD was not considered in [14], [15]. In this study, the proposed MAPD-D considers online tasks with deadlines, where tasks can be added at any time.

III. Problem Definition

In this section, we describe the shared definitions of MAPD [9] and MAPD-D and define the tasks specific to MAPD-D.

An instance of MAPD and MAPD-D consists of m agents in $A = a_1, \cdots, a_m$, a connected simple undirected graph $G = (V, E)$, and a set of unexecuted tasks $\mathcal{T} = \tau_1, \cdots, \tau_k$. Here, V represents the vertices, E denotes the edges, and $l_i(t) \in V$ indicates the location of agent a_i at timestep t. A path is a sequence of vertices associated with timesteps, indicating the vertex at which the agent is located at each timestep.

Agents either remain at their current node $l_i(t) = l_i(t+1)$ or move to an adjacent node via an edge $(l_i(t), l_i(t+1)) \in E$ at each timestep. Agents must avoid collisions with each other. MAPD and MAPD-D define two types of collisions [1], [9]: (1) Vertex conflict, where two agents cannot occupy the same location at the same timestep, i.e., for all agents $a_i, a_j\ (i \neq j)$ and all timesteps t, $l_i(t) \neq l_j(t)$ must hold; and (2) Swapping conflict, where two agents cannot move along the same edge in opposite directions at the same timestep, i.e., for all agents $a_i, a_j\ (i \neq j)$ and all timesteps t, $l_i(t) \neq l_j(t+1)$ or $l_j(t) \neq l_i(t+1)$ must hold.

We further provide an extended definition of tasks in MAPD-D. Each task $\tau_j \in \mathcal{T}$ consists of a pickup location $v_j^{\mathrm{p}} \in V$, a delivery location $v_j^{\mathrm{d}} \in V$, and a delivery deadline d_j^{d}. New tasks can be added to the task set $\mathcal{T}$ at each timestep. When an agent assigned to a task reaches the delivery location via the pickup location, the task is completed, and its completion time is denoted as c_j. Once a task is completed, the agent is assigned to a new task.

The objective of MAPD is to identify paths that execute all tasks in a timely manner, whereas MAPD-D aims to minimize task tardiness. We define the tardiness for the j-th task as $\epsilon_j = \max(0, c_j - d_j^{\mathrm{d}})$. For each task τ_j, $\epsilon_j = 0$ indicates the success of the task, whereas $\epsilon_j > 0$ indicates task failure. The objective function of MAPD-D is selected from the following two options:

- Minimizing the number of task failures [14], [16], [18]:

$$\min \sum_{1 \leq j \leq k} U(\epsilon_j), \tag{1}$$

  where $U(\cdot)$ is a unit step function [1].
- Minimizing the cumulative tardiness [15], [17]:

$$\min \sum_{1 \leq j \leq k} \epsilon_j. \tag{2}$$

In this study, we used the cumulative tardiness (2) as the objective function because it provides a more detailed evaluation of tardiness compared to the objective function (1), which only considers success or failure.

[1]The unit step function.

$$U(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

TABLE I
RELATED WORK.

	Multi-task	Deadlines	Individual release times	Online
Lifelong multi-agent path finding for online pickup and delivery tasks (MAPD) [9]	✓	×	✓	✓
Multi-agent path finding with deadlines (MAPF-DL) [16]	×	✓	×	×
Multi-robot path planning with due times (MRPP-DT) [17]	×	✓	×	×
Deadline-aware multi-agent tour planning (DA-MATP) [18]	×	✓	×	×
Multi-agent pickup and delivery with task deadlines (MAPD-TD) [14]	✓	✓	×	×
Minimizing task tardiness for multi-agent pickup and delivery [15]	✓	✓	✓	×
Proposed (MAPD-D)	✓	✓	✓	✓

IV. PROPOSED METHOD

This section outlines the D-TP algorithm employed to address the MAPD-D problem, which is an extension of the existing TP method. Typically, TP assigns tasks to agents solely based on execution cost. To reduce task tardiness, we introduce enhancements in two key areas: the calculation of pickup deadlines and deadline-aware task assignment. Additionally, we present D-TPTS, which improves flexibility and efficiency through task-swapping strategies.

A. D-TP

1) Calculation of Pickup Deadlines: The pickup deadline is calculated based on the delivery deadline when a new task τ_j is added. We prepared a dummy agent, implemented using prioritized path planning [11]. D-TP calculates the path for the dummy agent to depart from the delivery location v_j^{d} at time d_j^{d} and move towards the pickup location v_j^{p} by reversing the timesteps. During the path calculation, the order of the pickup and delivery locations is reversed because the time required for transportation can vary depending on the paths of other agents in the environment. The proposed method searches for a path from the delivery location to the pickup location by reversing the order of time, thus calculating the latest possible path (dummy path) that meets the delivery deadline. The pickup deadline d_j^{p} represents the time obtained by subtracting the length (timesteps) of the dummy path from the delivery deadline d_j^{d}.

2) Deadline-aware Task Assignment: The disparity between the calculated pickup deadline d_j^{p} and the current time t indicates the temporal margin of the deadline. In TP, the system assigns tasks to minimize the execution cost at the moment of assignment. In contrast, in D-TP, the system assigns tasks to agents in a manner that minimizes the weighted sum of the execution cost and the temporal margin relative to the deadline.

$$\underset{\tau_j \in \mathcal{T}'}{\operatorname{argmin}} \left(\alpha \cdot (d_j^{\mathrm{p}} - t) + (1 - \alpha) \cdot h\left(loc(a_i), v_j^{\mathrm{p}}\right) \right), \quad (3)$$

where $0 \leq \alpha \leq 1$ and $\mathcal{T}'$ denotes the set of tasks that can be assigned to agent a_i. The first term $(d_j^{\mathrm{p}} - t)$ denotes the temporal margin for the deadline of task τ_j; the second term $h(loc(a_i), v_j^{\mathrm{p}})$ indicates the h-value from the current location of agent a_i to the pickup location of the task, which represents the execution cost; and the parameter α indicates the weight for the urgency of the pickup deadline. When $\alpha = 0$, it is equivalent to the existing method [9] that does not take the deadline into account.

3) Algorithm: Algorithm 1 provides the pseudocode for D-TP; parts that differ from TP [9] are indicated in red. In lines 17–20, we define the UpdatePickupDeadline function. In line 19, a dummy agent is prepared to calculate the dummy path from the delivery location v_j^{d} to the pickup location v_j^{p} using the reversed-path finding function $\mathcal{RP}(dummy, v_j^{\mathrm{d}}, v_j^{\mathrm{p}}, token, d_j^{\mathrm{d}})$. In line 20, the pickup deadline d_j^{p} is calculated by subtracting the length of the dummy path. Here, $|\mathcal{P}|$ represents the length of the path, that is, the number of steps required to move.

In line 1, the token is initialized with trivial paths where all agents remain in their initial locations. At each timestep, the system adds all new tasks to the task set $\mathcal{T}$ (line 3). Subsequently, in line 4, the pickup deadline is calculated for the newly added tasks. Lines 5–15 handle the task assignment process. If one or more tasks are assignable, the system assigns a task considering both the execution cost and the margin until the pickup deadline d_j^{p}. The path from v_j^{p} to v_j^{d} is calculated by the function $\mathcal{P}(a_i, v_j^{\mathrm{p}}, v_j^{\mathrm{d}}, token, t)$. In cases where no tasks are assignable, the system handles deadlock resolution or maintains the current position of agents, as outlined in [9]. If new task assignments overwrite the dummy paths, the pickup deadline is recalculated (line 15). Finally, agents proceed along their paths in the token (line 16).

B. D-TPTS

In this section, we introduce two methods for task swapping that incorporate deadlines in MAPD-D: *Task swapping among agents* and *task switching*. *Task swapping among agents* involves swapping tasks between agents, while *task switching* focuses on swapping tasks within a single agent.

1) Task Swapping Among Agents: Ma et al. [9] proposed the token passing with task swaps (TPTS) algorithm as a solution to MAPD. In TP with task swapping, agents can be assigned not only "unassigned" tasks but also "tasks assigned to a different agent but not yet picked up." This flexibility can be advantageous, particularly when one agent can pick up a task faster than another. In such cases, the system reassigns the task from one agent to another, allowing for more efficient task completion.

In contrast to TPTS, which focuses solely on execution cost considerations, the proposed method modifies this approach to incorporate a weighted sum of the execution cost and temporal margin for deadlines, as expressed in (3).

Algorithm 1 Token passing for tasks with deadlines (TP-D)

1: Initialize $token$ with the (trivial) path $[loc(a_i)]$ for each agent a_i
2: **while** true **do**
3: Add all new tasks, if any, to task set $\mathcal{T}$
4: UPDATEPICKUPDEADLINE(new tasks, $token$)
5: **while** agent a_i that requests $token$ exists **do**
6: $\mathcal{T}' \leftarrow \{\tau_j \in \mathcal{T} \mid$ no other path in $token$ ends in v_j^{p} or $v_j^{\mathrm{d}}\}$
7: **if** $\mathcal{T}' \neq \phi$ **then**
8: $t \leftarrow$ current timestep
9: $\tau_{j^*} \leftarrow \underset{\tau_j \in \mathcal{T}'}{\operatorname{argmin}} \left(\alpha \cdot (d_j^{\mathrm{p}} - t) + (1-\alpha) \cdot h(loc(a_i), v_j^{\mathrm{p}})\right)$
10: Assign a_i to τ_{j^*}
11: Remove τ_{j^*} from $\mathcal{T}$
12: Update a_i's path in $token$ with $\mathcal{P}(a_i, v_{j^*}^{\mathrm{p}}, v_{j^*}^{\mathrm{d}}, token, t)$
13: **else**
14: Remove deadlock or stay
15: UPDATEPICKUPDEADLINE($\mathcal{T}$ whose dummy path is overwritten, $token$)
16: All agents move along their paths in token for one timestep
17: **function** UPDATEPICKUPDEADLINE($tasks, token$)
18: **for** $\tau_j \in tasks$ **do**
19: Update τ_j's dummy path in $token$ with $\mathcal{RP}(dummy, v_j^{\mathrm{d}}, v_j^{\mathrm{p}}, token, d_j^{\mathrm{d}})$
20: Update τ_j's pickup deadline d_j^{p} with $d_j^{\mathrm{d}} - |\mathcal{RP}(dummy, v_j^{\mathrm{d}}, v_j^{\mathrm{p}}, token, d_j^{\mathrm{d}})|$

2) Task Switching: In this approach, the agent is allowed to abandon its current task and undertake a more urgent task if a task with higher urgency appears closer when an agent is en route to the pickup location. We anticipate that task switching can reduce tardiness by prioritizing tasks with higher urgency.

An agent will abandon its current task if both of the following conditions are met:

- The urgency of the new task is higher than that of the current task.
- The execution cost of the new task is lower than that of the current task.

In other words, the following inequalities should hold simultaneously:

$$d_{\mathrm{new}}^{\mathrm{p}} < d_{\mathrm{cur}}^{\mathrm{p}} \tag{4}$$

$$h\left(loc(a_i), v_{\mathrm{new}}^{\mathrm{p}}\right) < h\left(loc(a_i), v_{\mathrm{cur}}^{\mathrm{p}}\right), \tag{5}$$

where cur denotes the index of the current task of the agent a_i and new represents the index of the new task.

3) Algorithm: Algorithm 2 provides the pseudocode for D-TPTS. The overall flow mirrors that of TPTS [9], with differences highlighted in red. Lines 5–10 implement task switching. When an agent is en route to the pickup locations, it abandons its current task and accepts a different task if the new task has an earlier pickup deadline and a lower execution cost than the current task. Task swapping is performed in the function GetTask (lines 15–32). As indicated in line 19, the system considers both the temporal margin of the pickup deadline and the weighted sum of the task execution cost.

Algorithm 2 Deadline-aware token passing with task swaps (D-TPTS)

1: Initialize $token$ with the (trivial) path $[loc(a_i)]$ for each agent a_i
2: **while** true **do**
3: Add all new tasks, if any, to task set $\mathcal{T}$
4: UPDATEPICKUPDEADLINE(new tasks, $token$)
5: **for** $\tau_j \in$ new tasks **do**
6: **for** agent a_i that is moving to the pickup location **do**
7: $\tau_{j'} \leftarrow$ task that a_i is executing
8: **if** $d_j^{\mathrm{p}} < d_{j'}^{\mathrm{p}}$ and $h(loc(a_i), v_j^{\mathrm{p}}) < h(loc(a_i), v_{j'}^{\mathrm{p}})$ **then**
9: Unassign a_i' from $\tau_{j'}$
10: Remove a_i''s path from $token$
11: **while** agent a_i that requests $token$ exists **do**
12: GETTASK($a_i, token$)
13: All agents move along their paths in token for one timestep
14: Remove tasks from $\mathcal{T}$ when agents start to execute them
15: **function** GETTASK($a_i, token$)
16: $\mathcal{T}' \leftarrow \{\tau_j \in \mathcal{T} \mid$ no other path in $token$ ends in v_j^{p} or $v_j^{\mathrm{d}}\}$
17: **while** $\mathcal{T}' \neq \phi$ **do**
18: $t \leftarrow$ current timestep
19: $\tau_{j^*} \leftarrow \underset{\tau_j \in \mathcal{T}'}{\operatorname{argmin}} \left(\alpha \cdot (d_j^{\mathrm{p}} - t) + (1-\alpha) \cdot h(loc(a_i), v_j^{\mathrm{p}})\right)$
20: Remove τ from $\mathcal{T}$
21: **if** no agent is assigned to τ_{j^*} **then**
22: Assign a_i to τ_{j^*}
23: Update a_i's path in $token$ with $\mathcal{P}(a_i, v_{j^*}^{\mathrm{p}}, v_{j^*}^{\mathrm{d}}, token, t)$
24: **else**
25: $a_i' \leftarrow$ agent that is assigned to τ_{j^*}
26: **if** a_i reaches $v_{j^*}^{\mathrm{p}}$ before a_i' **then**
27: Unassign a_i' from τ_{j^*} and assign a_i to τ_{j^*}
28: Remove a_i''s path from $token$
29: Break
30: **if** no task is assigned to a_i **then**
31: Remove deadlock or stay
32: UPDATEPICKUPDEADLINE($\mathcal{T}$ whose dummy path is overwritten, $token$)

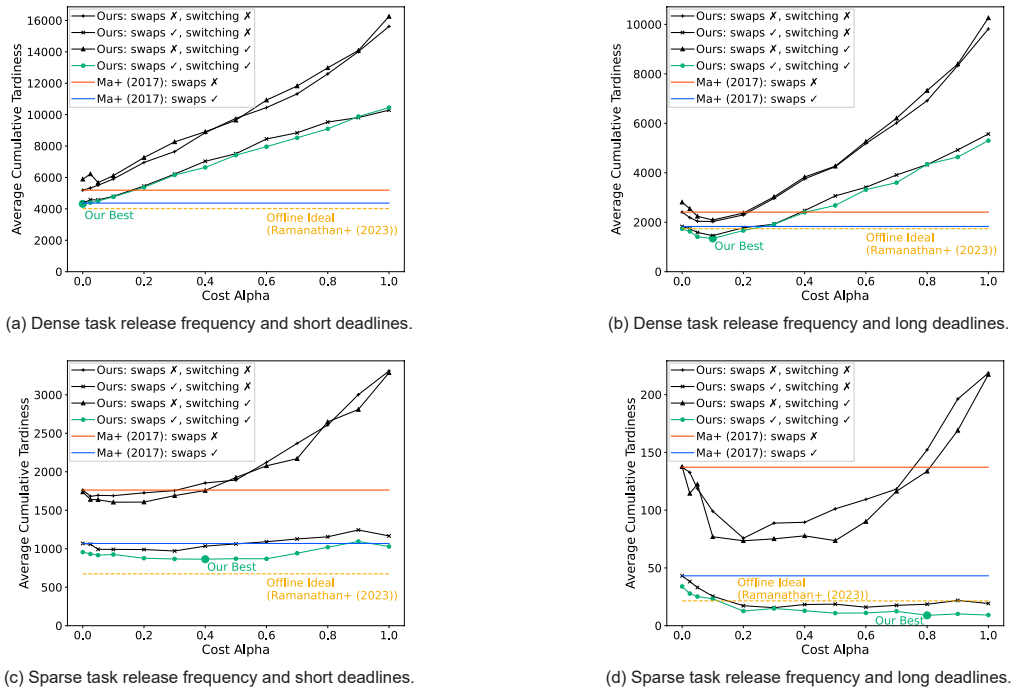
V. NUMERICAL EXPERIMENTS

This section outlines the numerical experiments conducted to compare the existing method (TP) with the proposed algorithms (D-TP and D-TPTS). Our primary focus is to evaluate the effectiveness of the proposed algorithms in reducing tardiness in an online setting.

A. Evaluation of Task Tardiness

The numerical experiments were carried out in a grid environment, representing an automated warehouse, as illustrated in Fig. 1. We generated 151 tasks by randomly selecting pickup and delivery locations from the task endpoints, ensuring no duplication. Each agent moved to the delivery location via the pickup location for the assigned task. We varied parameters such as task-release times and deadline duration after task release to examine their significant impact during the experiments (see Table II). The experiment involved 15 agents, with their initial positions randomly selected from the non-task endpoints.

We also examined an offline setting where tasks and their release times are predetermined, in contrast to the online setting where tasks can be added at any time. In an offline setting, tasks can be allocated with foresight to accommodate future tasks and preemptively moved in



(a) Dense task release frequency and short deadlines.

(b) Dense task release frequency and long deadlines.

(c) Sparse task release frequency and short deadlines.

(d) Sparse task release frequency and long deadlines.

Fig. 2. Comparisons of cumulative tardines.

TABLE II

SETTINGS OF TASK RELEASE FREQUENCY AND DEADLINES.

Task-release times	Dense [0, 300]	Sparse [0, 500]
Deadline duration after release	Short [20, 80]	Long [60, 120]

*Values are randomly selected from a uniform distribution in [min, max].

anticipation of release times. Generally, offline methods tend to yield solutions closer to the optimal solution compared to online methods. In this context, we referred to the method by Ramanathan et al. [15], who regarded deadlines in offline tasks as ideal benchmarks. They employed a method that sorted tasks based on deadlines and assigned tasks to agents with lower execution costs.

Fig. 2 illustrates the results of the proposed method alongside ideal values in the offline setting. The horizontal axis represents the weight α used during task assignment, where smaller values prioritize execution cost and larger values prioritize deadlines. When $\alpha = 0$ and task switching is not employed, the method is equivalent to TP and TPTS [9]. The vertical axis indicates the total tardiness for each task, averaged across the results of 30 experiments. Our analysis demonstrates that the cumulative tardiness varies depending on the value of α and the presence of task exchanges, regardless of the release frequency and deadlines.

B. Discussion

We begin by examining the variations in tardiness based on task-release frequency and deadline length. In Fig. 2, both the proposed and conventional methods exhibit notable trends in tardiness. Tardiness increases when tasks are released frequently and when deadlines are short. Densely released tasks leave agents with limited spare time, leading to a buildup of unexecuted tasks and an increase in cumulative tardiness. Similarly, shorter deadlines elevate the likelihood of tasks missing their deadlines, further contributing to cumulative tardiness.

Next, we explore the weight α in the proposed method. We compare the disparities in cumulative tardiness caused by varying α when neither task swapping nor task switching is implemented. Minimum tardiness is observed for dense task releases and short deadlines (Fig. 2(a)) at $\alpha = 0.0$, for dense releases and long deadlines (Fig. 2(b)) at $\alpha = 0.1$, for sparse releases and short deadlines (Fig. 2(c)) at $\alpha = 0.025$, and for sparse releases and long deadlines (Fig. 2(d)) at $\alpha = 0.2$. When tasks are released frequently and deadlines are short, prioritizing tasks with lower execution costs is crucial for timely management. However, in scenarios where tasks are not released frequently or deadlines are not short, there is some flexibility to consider deadlines. While TP and TPTS [9] solely considered execution costs ($\alpha = 0$), the proposed method integrates temporal margin for deadlines along with execution costs, resulting in reduced tardiness. Nonetheless, excessively prioritizing the temporal margin of deadlines may escalate the execution costs of each task. This can overwhelm agents and increasing cumulative tardiness. Hence, adjusting the value of α based on the situation is imperative.

In all experiments, the lowest tardiness was achieved when both task swapping and task switching were implemented. For example, in Fig. 2(b), implementing only task swapping reduced cumulative tardiness by 569.4, whereas

implementing both task swapping and task switching further reduced tardiness by an additional 116.7. Task switching facilitates task reassignment when more urgent tasks are added. Additionally, task swapping enables task exchanges between agents, leading to further reduction in cumulative tardiness.

However, implementing only task switching may elevate cumulative tardiness compared to not implementing anything, especially when task releases are frequent (Figs. 2(a) and 2(b)). This aligns with the discussion on the value of α; in scenarios with frequent task releases, minimizing execution costs outweighs considering task urgency. Task reassignment based on urgency through task switching increases execution costs, consequently amplifying tardiness.

Finally, we compare the proposed method (online) with the ideal values (offline). In most cases, the tardiness of the proposed method was equivalent to or worse than the ideal values (Figs. 2(a), (c), and (d)). However, in scenarios with dense tasks and long deadlines, the proposed method outperformed the ideal values (Fig. 2(b)). Ramanathan et al. [15] sorted tasks in advance based on deadlines and assigned them to agents with lower execution costs. This indicated that they prioritized deadlines over execution costs. They noted that their method excelled with extremely short deadlines, exhibiting less tardiness than the proposed method in settings with frequent releases and short deadlines. However, maintaining a balance between execution costs and tardiness becomes crucial when handling numerous tasks and longer deadlines. The effectiveness of the proposed method is evidenced in such scenarios despite operating online.

VI. Conclusions

This study addresses task deadlines by introducing a modified version of the MAPD problem, termed online MAPD-D. In online MAPD-D, tasks can be added at any time and assigned deadlines. To address MAPD-D, we propose two algorithms, namely D-TP and D-TPTS. D-TP allocates tasks by considering their pickup deadlines along with execution costs; meanwhile, D-TPTS facilitates task exchanges among agents and within a single agent. The conducted numerical experiments demonstrate that both D-TP and D-TPTS effectively reduce task tardiness compared to existing methods.

These experiments are conducted in a 35×21 grid environment; however, our ability to solve MAPD-D in larger environments is limited due to computational constraints. In the future, exploring the development of decentralized algorithms could enable the solution of large-scale MAPD-D. Additionally, algorithms should be devised to handle more realistic scenarios, such as paths being obstructed by uncertain obstacles [19].

Acknowledgments

We thank Kenji Ito, Keisuke Otaki, and Yasuhiro Yogo for their insightful inputs and discussions.

References

[1] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. Kumar, R. Barták, and E. Boyarski, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, 2019, pp. 151–158.

[2] O. Salzman and R. Stern, "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*, 2020, pp. 1711–1715.

[3] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses." *AI Magazine*, vol. 29, no. 1, pp. 9–20, 2008.

[4] W. Honig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and Robust Execution of MAPF Schedules in Warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[5] K. Dresner and P. Stone, "A Multiagent Approach to Autonomous Intersection Management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, 2008.

[6] A. Okoso, K. Otaki, and T. Nishi, "Multi-Agent Path Finding with Priority for Cooperative Automated Valet Parking," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2135–2140.

[7] J. Li, H. Zhang, M. Gong, Z. Liang, W. Liu, Z. Tong, L. Yi, R. Morris, C. Pasareanu, and S. Koenig, "Scheduling and Airport Taxiway Path Planning Under Uncertainty," in *Proceedings of the 2019 Aviation and Aeronautics Forum and Exposition*, 2019, pp. 1–8.

[8] D. Silver, "Cooperative Pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 1, 2005, pp. 117–122.

[9] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, "Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2017, pp. 837–845.

[10] H. Ma, W. Hönig, T. K. S. Kumar, N. Ayanian, and S. Koenig, "Lifelong path planning with kinematic constraints for multi-agent pickup and delivery," in *Proceedings of the AAAI Conference on Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19, 2019, pp. 7651–7658.

[11] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and Path Planning for Multi-Agent Pickup and Delivery," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2019, pp. 1152–1160.

[12] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig, "Lifelong Multi-Agent Path Finding in Large-Scale Warehouses," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 11 272–11 281.

[13] H. A. Aryadi, R. Bezerra, K. Ohno, K. Gunji, S. Kojima, M. Kuwahara, Y. Okada, M. Konyo, and S. Tadokoro, "Multi-Agent Pickup and Delivery in Transformable Production," in *Proceedings of the 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–8.

[14] X. Wu, Y. Liu, X. Tang, W. Cai, F. Bai, G. Khonstantine, and G. Zhao, "Multi-Agent Pickup and Delivery with Task Deadlines," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 12, 2021, pp. 206–208.

[15] S. Ramanathan, Y. Liu, X. Tang, W. Cai, and J. Li, "Minimising Task Tardiness for Multi-Agent Pickup and Delivery," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 2349–2351.

[16] H. Ma, G. Wagner, A. Felner, J. Li, T. K. S. Kumar, and S. Koenig, "Multi-Agent Path Finding with Deadlines," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018, pp. 417–423.

[17] H. Wang and W. Chen, "Multi-Robot Path Planning With Due Times," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4829–4836, 2022.

[18] T. Huang, V. Shivashankar, M. Caldara, J. Durham, J. Li, B. Dilkina, and S. Koenig, "Deadline-Aware Multi-Agent Tour Planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, 2023, pp. 189–197.

[19] B. Shofer, G. Shani, and R. Stern, "Multi Agent Path Finding under Obstacle Uncertainty," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, 2023, pp. 402–410.