Multi-Agent Pickup and Delivery with Task Deadlines

Xiaohu Wu,[1] Yihao Liu,[1] Xueyan Tang,[1] Wentong Cai,[1]
Funing Bai,[2] Gilbert Khonstantine,[1] Guopeng Zhao[2]
[1]Singtel Cognitive and Artificial Intelligence Lab for Enterprises, Nanyang Technological University, Singapore
[2]NCS Group, Singapore
xiaohu.wu@ntu.edu.sg, yihao002@e.ntu.edu.sg, {asxytang,aswtcai}@ntu.edu.sg,
eliza.bai@ncs.com.sg, gkhonsta001@e.ntu.edu.sg, leo.zhao@ncs.com.sg

Abstract

We study the multi-agent pickup and delivery problem with task deadlines, where a team of agents execute tasks with individual deadlines to maximize the number of tasks completed by their deadlines. We take an integrated approach that assigns and plans one task at a time taking into account the agent states resulting from all the previous task assignments and path planning. We define metrics to effectively determine which agent ought to execute a given task and which task is most worth assignment next. We leverage the bounding technique to greatly improve the computational efficiency.

Introduction

Multi-Agent Path Finding (MAPF) is a classical problem that aims to find collision-free paths for a group of agents to move from their current locations to their respective target locations with some metric optimized (Stern et al. 2019). Deadlines have been considered in the MAPF problem where there is a common deadline for all agents and the objective is to maximize the number of agents that can reach their target locations by the deadline. This problem is NP-hard. Optimal solutions can be derived via search-based approaches or integer linear programming (Ma et al. 2018).

Multi-Agent Pickup and Delivery (MAPD) is an extension to the MAPF problem where a set of delivery tasks are to be assigned to the agents for execution. A MAPD solution needs to determine the tasks as well as their order to be executed by each agent and plan collision-free paths for the agents to complete their assigned tasks. Heuristic approaches have been proposed to optimize the makespan metric for MAPD (Liu et al. 2019; Li et al. 2020; Farinelli, Contini, and Zorzi 2020). Deadline requirements, however, have not been considered in the MAPD problem.

In reality, there are many scenarios where tasks have individual deadlines. Each task has to be completed (i.e., the agent executing the task arrives at the delivery location) by a specific deadline, in order to satisfy distinct customers with the delivered services/items in a timely manner. In this paper, we study MAPD with task deadlines. We adopt an integrated approach that conducts task assignment and path planning together. In each task assignment, a favorable agent is chosen to execute the next task that is currently the most urgent according to the paths already planned for the tasks previously assigned. We define a metric called the flexibility of a task as the task deadline minus the earliest possible completion time among all the agents to execute the task. This metric allows us to effectively determine which task is most worth assignment next. Based on this metric, we propose a priority-based framework for joint task assignment and path planning.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Problem Definition

Consider an undirected connected graph $\mathcal{G} = (V, E)$ where the nodes in V correspond to locations and each edge in E corresponds to a connection between two locations along which agents can move. There are a set of M agents $\mathcal{A} = \{a_1, \ldots, a_M\}$, and a set of N tasks $\mathcal{T} = \{t_1, \ldots, t_N\}$. All tasks are available at timestep 0. Each task t_j has a pickup location $s_j \in V$, a delivery location $g_j \in V$ and a deadline d_j. To execute a task t_j, an agent has to move from its current location via the pickup location s_j to the delivery location g_j. Each agent has a unit carrying capacity and can execute only one task at a time. Each agent a_i has a unique parking location $p_i \in V$ where it initially stays at timestep 0 and it can exclusively access at any time. After an agent completes all its tasks, it returns to its parking location. We would like to assign tasks to agents and plan paths for agents to execute them. Our objective is to maximize the number of tasks completed by their deadlines. At each timestep, an agent can execute either a move action to move to an adjacent location or a wait action to stay at its current location. Collisions may occur among agents at a location or along an edge. To avoid collisions, the following constraints are imposed in path planning: (i) two agents cannot occupy the same location at the same timestep, and (ii) two agents cannot traverse the same edge in opposite directions at the same timestep. We refer to our problem as Multi-Agent Pickup and Delivery with Task Deadlines (MAPD-TD).

Algorithms for MAPD-TD

We propose a priority-based framework to perform task assignment and path planning in an integrated manner. Each task assignment decision is made based on the paths already planned for the tasks previously assigned. Once a task gets

assigned, the path for executing the task is planned immediately. We start by defining metrics to decide which task and which agent to choose for an assignment.

To decide which task to assign next, we define a metric called the *flexibility*. Let $c_{i,j}$ denote the timestep at which an unassigned task t_j can be completed by an agent a_i after finishing all its assigned tasks so far. The flexibility f_j of a task t_j is given by the task deadline minus the earliest possible completion time among all the agents to execute this task:

$$f_j = d_j - \min_{a_i \in \mathcal{A}} c_{i,j}. \tag{1}$$

The flexibility metric measures the urgency of the task. A lower flexibility value indicates that there is less time buffer and the task is more urgent to execute. Among all the unassigned tasks with non-negative flexibility values, we choose the task t_{j^*} with the lowest flexibility value to assign next:

$$j^* = \operatorname*{argmin}_{f_j \geq 0} f_j. \tag{2}$$

Agents differ in when they become available and where they become available for executing new tasks. Thus, they can have different costs to complete the task t_{j^*}. Suppose an agent a_i takes τ_i timesteps to finish all its assigned tasks according to the planned path. That is, a_i arrives at the delivery location of the last assigned task at timestep τ_i. To improve the resource efficiency, among all the agents that can complete the task t_{j^*} by its deadline d_{j^*}, we choose the agent a_{i^*} that has the lowest cost to execute t_{j^*}:

$$i^* = \operatorname*{argmin}_{c_{i,j^*} \leq d_i} (c_{i,j^*} - \tau_i). \tag{3}$$

The high-level idea of our priority-based framework is as follows. In each task assignment, we first compute the completion time of executing each unassigned task by each agent and then derives the flexibility of each task by (1). After that, we choose the task t_{j^*} satisfying (2) and assigns t_{j^*} to the agent a_{i^*} satisfying (3). In planning a path for an agent to execute a task, we make use of the multi-label A* algorithm (Grenouilleau, Hoeve, and Hooker 2019) to plan an optimal path for the agent to move from its current location via the task pickup location to the task delivery location. The A* search is conducted in the space of location-timestamp pairs taking into account the node and edge access constraints imposed by the paths already planned for the previously assigned tasks.

After the assignment and path planning of every task, the states of the agents change. Thus, a key challenge of implementation is to compute the flexibility of the unassigned tasks at every assignment based on the current states of the agents. We leverage the bounding technique in our framework to greatly improve the computational efficiency. A state-of-the-art method to avoid collisions in path planning is to reserve for every agent a dummy path from the agent's current location to its parking location whenever the agent finishes one task (Liu et al. 2019). This may involve plenty of extra vain computation of paths that the agents will never use. We improve this method by identifying the conditions under which planning such dummy paths is necessary and selectively reserving dummy paths for agents.

small warehouse						
ϕ	N \ M	10	20	30	40	50
0.0	$10 \times M$	0.9360	0.9230	0.8573	0.8212	0.7978
0.1	$10 \times M$	0.9660	0.9635	0.9560	0.9335	0.8980
0.25	$10 \times M$	0.9950	0.9920	0.9943	0.9890	0.9854
large warehouse						
ϕ	N \ M	60	90	120	150	180
0.0	$10 \times M$	0.8778	0.8194	0.7723	0.7198	0.6848
0.1	$10 \times M$	0.9707	0.9250	0.8587	0.8119	0.7604
0.25	$10 \times M$	0.9977	0.9930	0.9820	0.9638	0.8991

Table 1: Average success rate for our algorithm

small warehouse						
ϕ	N \ M	10	20	30	40	50
0.0	$10 \times M$	0.8670	0.8090	0.7760	0.7378	0.7214
0.1	$10 \times M$	0.9380	0.8920	0.8580	0.8198	0.7902
0.25	$10 \times M$	0.9890	0.9810	0.9630	0.9402	0.9098
large warehouse						
ϕ	N \ M	60	90	120	150	180
0.0	$10 \times M$	0.7867	0.7376	0.7047	0.6650	0.6247
0.1	$10 \times M$	0.8755	0.8137	0.7697	0.7253	0.6881
0.25	$10 \times M$	0.9792	0.9381	0.8917	0.8358	0.7908

Table 2: Average success rate for the baseline algorithm

Experimental Results

We simulate two warehouse environments of different sizes (Liu et al. 2019). To generate task deadlines, we run a simple algorithm to construct hypothetical task streams and find the hypothetical path for each task stream without considering any conflicts with other hypothetical paths. We specify a parameter ϕ and set the deadline of each task to $(1 + \phi)$ times its completion time in the hypothetical path.

We set the number of agents $M = 10, 20, 30, 40, 50$ for the small warehouse and set $M = 60, 90, 120, 150, 180$ for the large warehouse. We set the number of tasks $N = 10 \times M$ and $\phi = 0, 0.1, 0.25$. For each setting of (M, N, ϕ), we randomly generate 10 problem instances and present the average performance over these instances.

Table 1 shows the success rate of our algorithm (the ratio of the number of tasks completed by their deadlines to the total number of tasks). Table 2 shows the success rate of a baseline algorithm that, for each task to assign, simply chooses the agent giving rise to the task flexibility (1) (i.e., completing the task earliest). It can be seen that our algorithm choosing the agent with the lowest execution cost as given by (3) can improve the success rate by up to 10%.

Acknowledgments

This research was conducted at Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU), which is a collaboration between Singapore Telecommunications Limited (Singtel) and Nanyang Technological University (NTU) that is funded by the Singapore Government through the Industry Alignment Fund – Industry Collaboration Projects Grant.

References

Farinelli, A.; Contini, A.; and Zorzi, D. 2020. Decentralized Task Assignment for Multi-item Pickup and Delivery in Logistic Scenarios. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 1843–1845.

Grenouilleau, F.; Hoeve, W.-J. v.; and Hooker, J. N. 2019. A Multi-Label A* Algorithm for Multi-Agent Pathfinding. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS)*, 181–185.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. K. S.; and Koenig, S. 2020. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 1898–1900.

Liu, M.; Ma, H.; Li, J.; and Koenig, S. 2019. Task and Path Planning for Multi-Agent Pickup and Delivery. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 1152–1160.

Ma, H.; Wagner, G.; Felner, A.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2018. Multi-Agent Path Finding with Deadlines. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 417–423.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Bartak, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the 12th International Symposium on Combinatorial Search (SoCS)*, 151–158.