Multi-Agent Pickup and Delivery with Task Probability Distribution

Extended Abstract

Andrea Di Pietro
Politecnico di Milano
Milano, Italy
andrea4.dipietro@mail.polimi.it

Nicola Basilico
Università degli Studi di Milano
Milano, Italy
nicola.basilico@unimi.it

Francesco Amigoni
Politecnico di Milano
Milano, Italy
francesco.amigoni@polimi.it

ABSTRACT

Multi-Agent Pickup and Delivery (MAPD) consists in completing a set of tasks by having agents move to the pickup location and then to the delivery location of each task. In MAPD, new tasks are dynamically added to the system throughout its lifetime and existing algorithms usually assume either complete ignorance or full knowledge about the position and the time at which future tasks will appear until they are actually added to the system. This paper introduces a novel MAPD problem in which a spatial and temporal probability distribution of future tasks is known and defines algorithms that take advantage of this knowledge to reduce the average time required to execute tasks. In particular, we build on an existing MAPD algorithm, Token Passing (TP), proposing different ways to exploit a given task probability distribution. Experiments show that these methods can have a positive impact on the time required to complete the tasks.

KEYWORDS

Multi-Agent Pickup and Delivery; Task Probability Distribution

ACM Reference Format:
Andrea Di Pietro, Nicola Basilico, and Francesco Amigoni. 2023. Multi-Agent Pickup and Delivery with Task Probability Distribution: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 INTRODUCTION

In MAPD [5], agents have to plan collision-free paths on a graph in order to execute a set of tasks, each consisting of reaching a pickup vertex and then a delivery one. MAPD generalizes the well-known Multi-Agent Path Finding (MAPF) problem [11], where each agent has to complete a single and pre-assigned task whose pickup location is the agent's start vertex. Warehouse management [13], aircraft operations [7], ride-sharing services [2], and video games [6] are examples of important real-world applications where this problem might appear.

To solve MAPD, the literature features both online and offline methods. In the online approach, tasks are assumed to become known only upon their arrival. Some algorithms decompose the problem into a sequence of MAPF instances [1] where each free agent can be assigned to a new task and paths are replanned at each time [12] for all agents or for a group of them (e.g., only for free agents [5]). Offline approaches, on the other hand, assume that tasks are known in advance and solve the assignment and path-planning problems in a single phase [3, 8].

The above two families of approaches are antithetical with respect to how they are informed about future tasks. The former assumes that the system is completely unaware of any information about tasks until their arrival. The latter assumes to have full knowledge at planning time. Following the recognized need for intermediate approaches [4, 9], in this work we propose online solution methods for MAPD that are informed by a prior probability distribution of future task arrivals. Such knowledge might be available in those settings where historical data can be used to predict where and when new tasks will appear and could be exploited to improve task assignment and path planning [9].

The methods we propose are refinements of TP [5], arguably one of the most common algorithms for MAPD. TP is a decentralized online algorithm in which agents assign themselves to tasks and plan paths one after another by passing a shared *token* that contains the current task set, task assignments, and paths. It leverages the same idea of Cooperative A* [10], where each agent plans its optimal path under the constraints imposed by the agents that planned before.

2 PROBLEM AND METHODS

We introduce *MAPD with task probability distribution* (*MAPD-P*). As in MAPD, we consider a graph $G = (V, E)$, where k agents, each at an initial vertex, can wait at the current vertex or move to an adjacent one. Each action (wait or move) costs one discrete time step. A set T contains the unassigned tasks, which are dynamically added at runtime. Each task is defined as $\tau_j = (s_j, g_j)$, where s_j and g_j are the pickup and delivery vertices, respectively. A function $P(t, s_j, g_j)$ gives the probability that task $\tau_j = (s_j, g_j)$ will arrive at time t. From P, we derive $P_1(t, s)$ as the probability that a task arrives at time t with pickup vertex s and any delivery vertex.

A solution for MAPD-P is a set of cost-minimizing collision-free paths (agents do not share the same vertex nor the same edge at the same time) that complete all tasks in a finite time. The solution's cost can be given by the *makespan*, that is, the minimum number of time steps after which all the tasks are completed, or by the *service time* defined as the average number of time steps elapsed from the arrival of a task and its completion. It can be easily shown that MAPD-P reduces to MAPF and, as a consequence, is NP-hard.

In the following, we outline three techniques to exploit the availability of P in the online resolution of MAPD-P. The techniques, called *TP-m2*, *TP-m1*, and *TP with preemption*, can be combined together and are implemented as refinements of the TP algorithm.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

They will be triggered for the specific conditions they cover, allowing a rollback to the standard TP's routines in any other case.

TP-m2. In general, it might happen that a free agent cannot be assigned to a new task because, at that time, there are no unassigned tasks in T. In TP, these agents simply wait at their current locations (if they are not delivery locations of other tasks) or at designated locations to not interfere with other agents. In any case, the waiting location has no relationship with possible future tasks.

Our first refinement, TP-m2, exploits P to take advantage of the waiting time of agents and make them move towards locations that are likely to become pickup vertices of future tasks. More precisely, given any (approximate) distance function h between two locations, if at time w a free agent a_i (call $loc(a_i)$ its current location) does not find a task in T, it will compute a path to the location s^* maximizing

$$p(w, s, a_i) = \frac{\sum_{t=w+1}^{w+1+h(loc(a_i),s)} P_1(t, s)}{1 + h(loc(a_i), s)}. \quad (1)$$

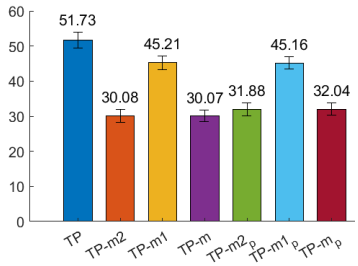
This function provides a score proportional to the probability that a task will appear at s by the time the agent would reach it. The value is divided by h to favor closer locations.

TP-m1. In TP, the agent with the token assigns itself to the closest task in T (if any) and there is no subsequent check on whether other agents become free closer to the pickup location of the task. When the closest task is far from the agent, it could be more convenient to speculatively leave the task to another agent and wait for future tasks with closer pickup locations. In TP-m1 we use P to make this kind of decision.

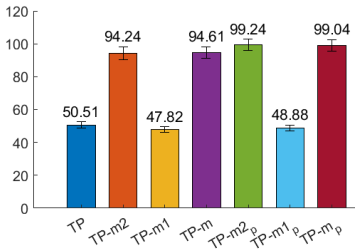
Instead of directly sending agent a_i to the pickup location s_j of the closest task $\tau_j \in T$, we verify whether s^* (computed by maximizing Equation 1) is a more attractive location. This location will replace s_j if (i) it is closer than s_j and (ii) $p(t, s^*, a_i) > (1 + h(loc(a_i), s_j))^{-1}$. Condition (ii), states that the attractiveness of s^* must be at least as large as the one of s_j (where a task resides with certainty). Notice that, due to condition (i), this rule is not excessively speculative since only locations that are closer than s_j will be considered.

TP-m[1,2] with preemeption. Planning for destinations chosen by any of the above methods can result in wasted efforts when speculated task arrivals do not realize. Suppose that an agent a_i is traveling to a speculatively chosen location s^* where no task will be present upon arrival. If in the meanwhile another task appears close to s^*, other free agents will be assigned to it. However, it is likely that a_i would be a better candidate since it is already approaching that task's neighborhood and, more importantly, it will be free once arrived there.

We leverage this rationale by giving a_i a preemption right to future tasks that might arrive inside a *preemption zone* defined by a space-time neighborhood of s^*. Specifically, when agent a_i decides to move to a location s^* where no task is currently available, the agent is assigned a preemption zone composed of the set of pickup locations s that satisfy the following conditions: (a) no other agent is traveling to s; (b) no current task has s as its pickup location; (c) s does not belong to any other preemption zone; (d) $h(s^*, s)$ is less than a threshold $\bar{h}$. If not redeemed before, preemption rights are cleared after $\bar{t}$ time steps have elapsed since the agent arrived at s^*.



(a) Service time



(b) Cost per task

Figure 1: Results with 80 task at frequency 0.05, 10 agents, warehouse with corridors.

3 EXPERIMENTAL RESULTS

Figure 1 reports results obtained in a warehouse environment with corridors defined on a grid of approx. 25×50 cells[1]. Tasks follow a Poisson distribution for the arrival time and a uniform distribution for locations. A* is used for planning paths for single agents and h is implemented as the Manhattan distance. When enabling preemption, $\bar{h}$ and $\bar{t}$ are set to 3. Results are averaged over 20 repeatable random runs. Besides service time, we consider also the *cost per task*, i.e., the number of agents' movements divided by the total number of tasks. Makespan is not reported here since no remarkable differences are observed among the different algorithms. In the figure, "TP-m" corresponds to having both *m1* and *m2* enabled, the subscript "p" stands for preemption. In these experiments, P encodes a full and exact knowledge about tasks arrivals. Under this idealistic scenario, we assess the maximum gains and costs achievable by our methods.

TP-m2 and TP-m, with and without preemption, have a lower service time than TP-m1 and TP-m1$_p$. Hence, exploiting agents' idleness seems to offer more potential margins of gain. This trend is observed only with low task frequencies: increasing the arrival rate keeps the agents always busy reducing exploitable idle time. Looking at the cost per task, we note that the large benefits of the TP-m2 come at greater consumption of resources. This is reasonable since in TP idle agents are just waiting at some locations instead of planning and following speculative paths.

[1]Code available at https://github.com/andrea4dipietro/MAPD-P/tree/master/MAPD-P.

ACKNOWLEDGMENTS

This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

REFERENCES

[1] J. Li, A. Tinka, S. Kiesel, J. Durham, S. Kumar, and S. Koenig. 2021. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 11272–11281.

[2] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye. 2019. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *Proceedings of the World Wide Web Conference (WWW)*. 983–994.

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

[4] H. Ma. 2020. *Target Assignment and Path Planning for Navigation Tasks with Teams of Agents*. Ph.D. Dissertation. University of Southern California, Department of Computer Science, Los Angeles, CA.

[5] H. Ma, J. Li, S. Kumar, and S. Koenig. 2017. Lifelong multi-agent path finding for online pickup and delivery tasks. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 837–845.

[6] H. Ma, J. Yang, L. Cohen, S. Kumar, and S. Koenig. 2017. Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments. In *Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 270–272.

[7] R. Morris, C. Pasareanu, K. Luckow, W. Malik, H. Ma, T. K. S. Kumar, and S. Koenig. 2016. Planning, Scheduling and Monitoring for Airport Surface Operations. *AI Magazine* 29(1) (2016), 9–20.

[8] V. Nguyen, P. Obermeier, T. Son, T. Schaub, and W. Yeoh. 2017. Generalized Target Assignment and Path Finding Using Answer Set Programming. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1216–1223.

[9] O. Salzman and R. Stern. 2020. Research challenges and opportunities in multi-agent path finding and multiagent pickup and delivery problems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 662–667.

[10] D. Silver. 2005. Cooperative pathfinding. In *First Artificial Intelligence and Interactive Digital Entertainment Conference*. 117–122.

[11] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, S. Kumar, E. Boyarski, and R. Bartak. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*. 151–158.

[12] J. Svancara, M. Vlk, R. Stern, D. Atzmon, and R. Bartak. 2019. Online multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 7732–7739.

[13] P. Wurman, R. D'Andrea, and M. Mountz. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AAAI-16 Workshop on Planning for Hybrid Systems* 29 (2008), 608–614.