Multi-Agent Pickup and Delivery with Batteries

Extended Abstract

Marcello Bavaro
Politecnico di Milano
Milan, Italy
marcello.bavaro@polimi.it

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

ABSTRACT

In Multi-Agent Pickup and Delivery (MAPD), a group of moving agents plan coordinated paths to execute pickup and delivery tasks appearing online in a known environment. The typical application of MAPD is in warehouses, where the agents are mobile robots powered by batteries. Current research on MAPD does not fully take into account the need for the agents to recharge their batteries when planning paths. In this paper, we study a variant of the MAPD problem, called MAPD-b, which considers battery consumption and charging stations, and we propose an algorithm to solve it.

KEYWORDS

Multi-Agent Pickup and Delivery; Path Planning; Batteries

ACM Reference Format:
Marcello Bavaro and Francesco Amigoni. 2025. Multi-Agent Pickup and Delivery with Batteries: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

Multi-Agent Pickup and Delivery (MAPD) refers to the problem of planning coordinated conflict-free paths for a set of agents that perform pickup and delivery tasks appearing online, within a common known environment [3]. MAPD models various real-world situations and its main application is when agents are robots in a warehouse [3]. In this setting, robots are often powered by batteries with limited capacity. While battery capacity usually allows a robot to work for a working day (e.g., 8 hours), during peak periods robots may have to work for a longer time, making it relevant to include battery recharging during planning [4, 9].

Notwithstanding their importance, to the best of our knowledge, batteries are not usually considered in the literature about coordinated path planning for MAPD. Beyond MAPD, [5–7] propose solutions for routing battery-powered electric vehicles in an offline setting, in which all tasks are known a priori. In such a scenario, it is possible to efficiently adopt solving techniques like integer linear programming or similar ones, while in our problem the tasks are not known a priori and appear online. In [2], the solution is limited to the scenario in which each agent is associated with a fixed charging station and can charge only after fulfilling a set of tasks. Moreover, battery consumption only depends on the length of the paths without taking into account wait actions.

In this paper, we introduce a variant of the MAPD problem, called *MAPD-b*, that accounts for agents' batteries consumption and the presence of charging stations where agents can recharge. We assume to have a model of the consumption for the actions of the agents and we present an algorithm that exploits it to plan paths for the agents to complete the tasks and to recharge.

2 PROBLEM STATEMENT

A *MAPD with batteries* (*MAPD-b*) problem is a variation of the plain MAPD problem [3]. There are n agents $A = \{a_1, a_2, \ldots, a_n\}$ that move in an environment modeled as an undirected connected graph $G = (V, E)$. Here, we consider environments G that are 2-dimensional grids with square cells. Cells can be free or obstacles. Free cells are the vertices V. We consider 4-connected grids: two free cells v and v' are adjacent, namely $(v, v') \in E$, if their squares share an edge. (Our results can be easily generalized to generic graphs G.) A task set $\mathcal{T}$ stores tasks that are dynamically added to the system. A task $\tau_j = (s_j, g_j) \in \mathcal{T}$ is composed of a pickup location s_j and a delivery location g_j. There are m *charging stations* $C = \{c_1, c_2, \ldots, c_m\}$ which are located in free cells (vertices in V). We assume that $n = m$, namely the number of agents is equal to the number of charging stations.

Time is discretized into time steps. At each time step, an agent a_i can perform two actions: *move* to an adjacent location v' connected to its current location v by an edge $e = (v, v')$ or *wait* in its current location. Both actions last one time step or, equivalently, cost one unit of time. Two types of conflicts must be avoided when planning paths: *vertex conflicts*, when two agents occupy the same vertex at the same time step, and *swapping conflicts*, when two agents cross the same edge in opposite directions at the same time step.

Each agent a_i has a *battery* whose charge level at time step t is denoted as b_i^t while its maximum charge level is denoted by B_i. At the beginning, all the batteries are fully charged ($b_i^{t=0} = B_i$). Actions performed by an agent consume an amount of battery C_m for the move actions and C_w for the wait actions. We assume that the consumptions are identical for all the agents (this assumption can be easily removed). An agent can recharge its battery at any charging station. We assume (but this can be easily changed) that, for each time step waiting in a charging station, an agent a_i recharges its battery of an amount $0.1 \cdot B_i$ until the battery is fully charged.

Solving a MAPD-b problem consists of planning conflict-free paths for the agents to complete all the tasks. To complete a task τ_j an agent must visit its pickup (s_j) and delivery (g_j) locations, in this order. The quality of a solution is evaluated in terms of *makespan*, which is the time step at which the last task is completed, or *service*

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

time, which is the average time between the time step at which a task τ appears in $\mathcal{T}$ and the time step at which τ is completed. An optimal solution minimizes either makespan or service time. The MAPD problem is NP-hard to solve optimally [8] and so is also the MAPD-b problem.

Note that, in our setting, agents deplete their batteries and, at some point, they need to recharge in order to be able to complete the tasks. Note also that, when some agents run out of their batteries, the remaining agents could still complete all the tasks, but likely with a larger makespan and service time.

3 PROPOSED ALGORITHM

The algorithm we propose in this paper to address the MAPD-b problem is a variation of the Token Passing (TP) algorithm. TP is a popular decentralized algorithm used to solve MAPD problems [3]. Its main idea is that single agents plan paths in turns, taking into account the paths already planned by other agents to avoid conflicts. TP is based on a block of memory shared among the agents and called *token*. It contains the current paths of all agents and the current set of tasks $\mathcal{T}$. Agents not executing any task request the token, which is assigned to each one of them in turns. The token holder analyzes the available tasks, chooses the one with the closest pickup location, uses a single-agent path planner (e.g., A*) to compute a minimum cost path to complete the task that is conflict-free (given the paths in the token), stores it in the token, and releases the token.

Our algorithm is called *A-TP* because it is far-sighted as the ant (A) of the fable by Aesop[1]. In A-TP, agents behave slightly differently than in plain TP. When, at time step t, an agent a_i plans a path to complete a task $\tau_j = (s_j, g_j)$, a path is returned if and only if the battery level $b_i^{t+\Delta t}$ of a_i at time step $t + \Delta t$ when it will be in g_j will be enough to follow a path that reaches an available charging station $c_k \in C$. A charging station c_k is *available* at t when no agent is at c_k or is heading to c_k (i.e., there is no path heading to c_k in the token) at t. The battery consumption of a path is computed according to the consumptions of move and wait actions (C_m and C_w, respectively) composing the path. Practically, when an agent a_i plans a path to complete a task, it reserves a subsequent path to a charging station. In this way, an agent always has a reserved path to a charging station, which is stored in the token, so other agents can see it and plan accordingly. After an agent a_i completes its current task, it decides whether to recharge or execute another task. If a path to complete a task (and then reach a charging station, as described above) exists given the current battery level, then a_i performs the task. Otherwise, a_i reserves a path to reach the closest available charging station starting from the next time step $t + 1$ (to have the chance of performing any new task that could appear at $t + 1$). If this is not possible, a_i immediately moves towards a charging station following the previously reserved path.

When using the A-TP algorithm in MAPD-b, no agent will run out-of-battery if (a) the initial battery levels of all agents are enough to reach a charging station from their start locations, (b) at least one agent has a maximum battery level sufficient to complete the most expensive task, and (c) the consumption model (C_m and C_w) is accurate.

[1] https://en.wikipedia.org/wiki/The_Ant_and_the_Grasshopper

4 EXPERIMENTAL ACTIVITY

The A-TP algorithm is implemented in Python 3.10 and tested in two simulated warehouses. The first warehouse is represented by a grid of 15×15 cells with 10 agents, 10 charging stations, 24 delivery locations, and 24 pickup locations. The second warehouse is bigger and composed of 40×40 cells with 26 agents, 26 charging stations, 40 delivery locations, and 216 pickup locations. The density of agents in the environments and the values for C_m and C_w (see later) are realistically set according to [1].

For each run, 2000 or 6000 tasks are generated by randomly choosing pickup and delivery locations. The initial battery level of each agent a_i, corresponding to the maximum level B_i, is randomly generated from a uniform distribution $U(80, 100)$. In the tests, the paths are planned using A* with a heuristic h equal to the Manhattan distance. We test for various C_m and C_w consumption configurations, ranging from low to high consumption levels. Each configuration (number of tasks, C_m, C_w) is tested averaging over 20 runs. A run is considered completed if all the tasks are executed within 30,000 time steps.

Due to space constraints, we discuss only a very small sample of the results we obtained. For example, on the 15×15 warehouse with $C_m = 1$ and $C_w = 0.1$ (for which agents need to recharge their batteries on average 50 times for each run), A-TP successfully completes all 20 runs. In the bigger environment, with $C_m = 0.01$, $C_w = 0.01$, and 6000 tasks (for which each agent needs to recharge at least once) A-TP completes all the tasks. In all experiments we performed, no agent ran out-of-battery, as expected. The drawback is the doubling of the number of calls to the A* path planner made by A-TP. Basically, the number of calls to A* for planning recharging paths is almost equal to the number of calls to A* for planning paths to complete tasks. This, however, does not prevent the application of A-TP to real-world scenarios since it needs, on average, 50 ms per time step to plan on our hardware (a computer with an Intel Core i5-12400F and 16 GB of DDR4 RAM).

5 CONCLUSION

In this paper, we presented a variant of the MAPD problem, called MAPD-b, which accounts for the depletion of the batteries of agents and allows for the presence of charging stations. We also introduced a solving algorithm for MAPD-b, called A-TP, that guarantees that no agent runs out-of-battery. In future work, we will investigate variants of A-TP that require fewer charging stations than agents ($m < n$). Moreover, we will start to move towards implementing our algorithm on real robots.

ACKNOWLEDGMENTS

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

REFERENCES

[1] E. Guizzo. 2008. KIVA SYSTEMS: three engineers, hundreds of robots, one warehouse. IEEE Spectrum. https://spectrum.ieee.org/three-engineers-hundreds-of-robots-one-warehouse
[2] F. Kudo and K. Cai. 2024. Anytime Multi-Task Multi-Agent Pickup and Delivery Under Energy Constraint. *IEEE RA-L* 9, 11 (2024), 10145–10152.
[3] H. Ma, J. Li, T. Kumar, and S. Koenig. 2017. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proc. AAMAS*. 837–845.

[4] D. McNulty, A. Hennessy, M. Li, E. Armstrong, and K. Ryan. 2022. A review of Li-ion batteries for autonomous mobile robots: Perspectives and outlook for the future. *J Power Sources* 545 (2022), 231943.
[5] H. Qin, X. Su, T. Ren, and Z. Luo. 2021. A review on the electric vehicle routing problems: Variants and algorithms. *Frontiers of Engineering Management* 8 (2021), 370–389.
[6] M. Soysal, M. Çimen, and S. Belbağ. 2020. Pickup and delivery with electric vehicles under stochastic battery depletion. *Computers & Industrial Engineering* 146 (2020), 106512.
[7] J. Yang and H. Sun. 2015. Battery swap station location-routing problem with capacitated electric vehicles. *Comput Oper Res* 55 (2015), 217–232.
[8] J. Yu and S. LaValle. 2013. Structure and intractability of optimal multi-robot path planning on graphs. In *Proc. AAAI*. 1443–1449.
[9] B. Zou, X. Xu, and R. De Koster. 2018. Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. *Eur J Oper Res* 267, 2 (2018), 733–753.