Multi-Agent Pickup and Delivery in Dynamic Environments

Benedetta Flammini[1], Davide Azzalini[1] and Francesco Amigoni[1]

[1]*Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria*

Abstract

In a Multi-Agent Pickup and Delivery (MAPD) problem, a group of agents has to accomplish subsequent pickup and delivery tasks while avoiding collisions. Tasks are provided at runtime, making MAPD a combination of Multi-Agent Path Finding (MAPF) and online task assignment. In this paper, we consider a new formulation of MAPD, in which a team of agents has to solve a MAPD problem without interfering or communicating with other agents not belonging to the team that are operating in the same environment. We address the problem from the point of view of the team of agents, and we propose that they build a model of the behavior of the agents external to the team and exploit this information in planning their paths. We consider different levels of knowledge that the team of agents can have about the other agents, and propose different solutions accordingly. Experimental results show that the inclusion of information about the behavior of external agents in the team's planning phase reduces the number of potential collisions and decreases tasks' completion time for the team.

Keywords

Multi-Agent Path Finding, Multi-Agent Pickup and Delivery, Multirobot Systems

1. Introduction

The MAPD problem [1] consists in finding collision-free paths for a team of agents executing incoming tasks by moving from their starting locations to the delivery locations, passing through the pickup locations. MAPD problems have a dynamic nature since new tasks can be added at any time, and agents have to assign and complete them in an online manner. The MAPD problem is currently very popular within both academia and industry due to its several real-world applications, especially in logistics [2, 3, 4, 5]. We consider a new formulation of MAPD, in which a team of agents has to solve a MAPD problem in an environment in which other moving agents are present. The team of agents has to accomplish its MAPD tasks without interfering or communicating with agents not belonging to the team. Possible applications of this setting can be found in logistics: for example, it can happen in warehouses that some items fall on the ground, or that some locations need cleaning. So, while the agents of the team perform their usual pickup and delivery tasks, other agents (which can also be humans) clean or remove items from the ground.

Since we conservatively assume that the team of agents cannot communicate with external agents and does not know their plans, collisions between a team agent and an external agent may happen. Every time there is a potential collision, the team agent has to reactively avoid the

10th Italian Workshop on Artificial Artificial Intelligence and Robotics (AIRO 2023).

benedetta.flammini@polimi.it (B. Flammini); davide.azzalini@polimi.it (D. Azzalini); francesco.amigoni@polimi.it (F. Amigoni)

collision and replan its path. Frequent replans can slow down the completion of tasks, implying that team agents will need more time to solve their MAPD problem. To overcome this issue, we propose that agents in the team model the behavior of the external agents and use this information to plan better paths that attempt to avoid collisions.

To model the behavior of the external agents, we consider different levels of knowledge that can be available to the team, which trigger different solutions. The levels of knowledge considered are: (1) the team of agents knows the current positions of the external agents in real-time and the tasks they are assigned to; (2) the team of agents knows a set of paths followed by the external agents in the past; (3) the team of agents knows anything about external agents.

Our work is related to the problem of multi-agent moving obstacles' avoidance [6, 7, 8], but differs for the important aspect that in the latter it is not always possible to assume that obstacles follow a behavior that is worth to model over an extended period.

2. Background

The MAPD problem [1] involves agents in an environment represented by an undirected connected graph $G = (V, E)$ (as it is common in this field), where vertices in V represent the locations of the environment, and edges in E the connections between them. Time is assumed to be discrete, and at each time step each agent performs an action $a : V \rightarrow V$. Two types of actions are allowed: remain in the current vertex or move to an adjacent one. All actions are assumed to have unitary cost.

A task set $\mathcal{T}$ contains all the tasks that have not been assigned and, due to the dynamic nature of the problem, new tasks can be added at any time. Each task $\tau_j \in \mathcal{T}$ is composed of a pickup location $s_j \in V$ and a delivery location $d_j \in V$. Each agent can have a single task assigned at a time, and a task can be assigned to only one agent. To solve an assigned task, an agent i has to plan and perform a sequence of actions $\pi_i = (a_1, \ldots, a_n)$ that brings it from its current location to the pickup location and then to the delivery location of the task. Agents' paths must not collide, that is: two different agents cannot be in the same location at the same time (*vertex conflict*) and they cannot traverse the same edge in opposite directions at the same time (*swapping conflict*). The aim of a MAPD problem is to plan paths that complete all the tasks in the shortest time.

Token Passing (TP) [9] is a decentralized MAPD algorithm in which each agent assigns itself a task in $\mathcal{T}$ and plans its collision-free paths exploiting the global information contained in the *token*, a synchronized block of memory shared among the agents that includes the task set $\mathcal{T}$, current tasks' assignments, and current agents' paths.

3. Problem Formulation

Given an environment represented as a graph $G = (V, E)$, we define the *MAPD problem in a dynamic environment* from the perspective of a team $\mathcal{A} = \{a_1, ..., a_k\}$ of k agents that has to perform a MAPD problem while in the same environment there are other external moving agents $\mathcal{H} = \{h_1, ..., h_m\}$. Agents $\mathcal{H}$ are not necessarily coordinated in performing their tasks.

We assume the team and the other agents are neither collaborative nor adversarial. We also make a *no-interference assumption*, namely, we assume external agents' decisions to move to be immutable, irrespective of the presence of the team agents, and hence the team agents are in charge of implementing proactive and/or reactive behaviors to avoid collisions. To allow the team agents to implement reactive behaviors for collision avoidance, we assume that each team agent $a_i \in \mathcal{A}$ can detect external agents within a field of view $FOV(a_i) = \big\{l \in V \mid \exists \pi = (\mathit{loc}(a_i), \ldots, l) \text{ with } |\pi| \leq 2\big\}$, which covers all locations l of the environment G that are reachable from the current location $\mathit{loc}(a_i)$ of a_i with paths π of length 2 or less. We assume that the team agents know the location and the next move of external agents within their FOVs.

4. Anticipating Other Agents' Behavior using Prediction Trees

We now suppose the team of agents knows the current positions of external agents, the locations each one of them must reach, and the fact that external agents are coordinated and rational (i.e., they move along the shortest conflict-free paths). The idea is to simulate all possible paths external agents may take to complete their tasks so the team can avoid future potential conflicts.

4.1. Proposed Approaches

We store all the possible shortest paths that external agents can follow to complete their tasks in a *Path Prediction Tree (PPT).* Although we assume external agents plan their paths in a coordinated way, we do not make any assumption on the algorithm they use to coordinate themselves or to plan their paths, nor we make assumptions about whether hierarchies or priorities exist among them (e.g., team agents know that external agents address a MAPD problem but do not know the algorithms they employ). As a consequence, when we simulate external agents' paths in the PPT, we need to consider all possible priority orders that external agents could have followed to make their conflict-free plans. This means that, when simulating the future behavior of each external agent, we have to consider higher-priority agents' paths as obstacles. Once all the combinations of possible shortest paths for each possible priority order are found, the PPT is complete. The tree is not immutable: at the end of each timestep, after all team and external agents have executed their actions, all the predicted paths in the PPT are checked to see if they are still valid given the new locations of external agents and are otherwise pruned. Moreover, when an external agent completes a task and is assigned a new one, the PPT needs to be updated accordingly. The PPT can be created and updated by any agent of the team and is stored in the token so that all team agents can access it. As, especially in grid environments, the computation of all possible combinations of shortest paths is prone to combinatorial explosion and becomes computationally infeasible even for medium-sized environments, we also develop a windowed version of the approach based on *bounded-horizon planning* [10, 11, 12] for the team agents and on simulating only the next w steps of external agents' plans. The low-level algorithm used for planning within PPTs is a modified version of Dijkstra, adapted to navigate a space-time state space, find all the shortest paths connecting two locations, and exploit bounded-horizon planning.

4.2. Experimental Results

We validate our approach on two types of grid environments: warehouses with narrow corridors and more open environments inspired by city layouts, with 3 external agents and 2 team agents. The metrics used for evaluation are the number of potential conflicts generated (i.e., team agents' replans) and the service time, i.e., the average number of time steps needed to finish executing a task after it has been added to the task set $\mathcal{T}$. Although the non-windowed version allows to avoid potential conflicts completely, it is computationally unfeasible for environments bigger than 80×80 cells. When adopting the windowed version, a trade-off needs to be made between the computing time and number of potential conflicts admissible. For example, with a window size $w = 6$ we observe an average decrease of around 50% in the number of potential conflicts without hurting the service time w.r.t. ignoring the presence of other agents.

5. Learning Behavioral Models using Past Observations

This section deals with the case in which the team of agents knows a set of paths performed by the external agents in the past. The idea is to build a model of the external agents using past observations of their movements.

5.1. Proposed Approaches

The first approach [13] is an extension to TP, called TP with collision avoidance and replanning + model (TP-CA-M), in which the team agents implement a proactive behavior at planning time for minimizing the probability of incurring in potential conflicts, in addition to the reactive one. The algorithm computes an estimate for the probability of occupancy p_i for each vertex $v_i \in V$ as the ratio of time steps in which v_i has been observed in the past to be occupied by some external agent, over the total number of time steps constituting the observation period. Then, the solving algorithm redefines the underlying graph as $\hat{G} = (V, \hat{E})$ maintaining the same vertices and edges as in G, but assigning a weight to the edges. The weight w_{ij} of an edge $(v_i, v_j) \in \hat{E}$ represents the cost of moving from v_i to v_j, and is defined as a linear combination of the distance between these vertices and the estimated probability of occupancy for the destination vertex v_j.

In the second approach, the focus is to estimate the position of each external agent i maintaining a belief b_i^t, that is a probability distribution over all possible vertices $v \in V$ expressing the likelihood for agent i to occupy a given vertex at time t. The beliefs $\{b_i^t\}$ can then be aggregated into a probability $p_o(v_k)$ of occupancy for vertex v_k (at time t) via the complement rule. To propagate the belief through time, the movement of an external agent is modeled with a n-th order Markov chain, where the states correspond to the vertices $v \in V$. We can interpret a belief b_i^t as the probability distribution over the states of the Markov chain at time t, and therefore we can obtain the corresponding belief state after τ time steps as $b_i^{t+\tau} = b_i^t P^\tau$, where P is the transition matrix of the Markov chain learned from the set of paths performed by the external agents in the past using maximum likelihood estimation. This allows the model to make predictions on the future positions of the agents when it comes to planning, and to update the belief state in light of the observations collected via field of view at each time step.

The idea of a weighted graph is maintained, however, the weight attributed to an edge is not stationary anymore: $w_{ij}(\tau) = (1-\alpha)\frac{1}{diam(G)} + \alpha p_c(i,j,\tau)$. Since the belief state of an agent is propagated over time, it now depends on the future time step τ at which the transition takes place. Term $p_c(i,j,\tau)$ combines the probability of a vertex conflict and of a swapping conflict.

5.2. Experimental Results

Two types of grid environments are used for testing, both presenting the features of a warehouse with narrow corridors, using an observation window of 600 time steps, 4 external agents, and 3 team agents. Best results are obtained for $\alpha = 0.2$: a significant reduction of potential conflicts (i.e., team agents' replans) w.r.t. a purely reactive approach is observed both for TP-CA-M (reduction of 37%) and, especially, Markovian models (79% for the second order). The service time is similar to the purely reactive approach, with a reduction of 2.5% for TP-CA-M and 4% for the second-order Markovian model, meaning that planning for paths with a lower number of potential conflicts results in a small reduction in the speed of completion of the tasks.

6. Learning Behavioral Models using Real-Time Observations

It can happen that team agents do not have information regarding the past behavior of external agents. For this reason, we discuss the case in which team agents have to build the behavioral model of the external agents relying solely on the information they observe when external agents enter their field of view. The models presented in Section 5 are preserved, with the difference that in the probabilistic matrix and the transition matrix of the first and second approach, respectively, the probability is initially uniformly distributed and is updated according to the observations made at runtime. In terms of computing effort, learning other agents' behaviors in real-time when using the probability occupation matrix or a first-order Markov chain just requires the update of a matrix, which is feasible to perform online at runtime. For higher-order Markov chains, it could become computationally unfeasible to do it online.

The results collected under this scenario, without any initial window of observation, show that the performance of a trained probabilistic/Markovian model and an untrained one, that simply hypothesizes a uniform distribution for the probabilistic/transition matrix, are not that different. This highlights the difficulty of our models in learning a distinctive behavior for the external agents: a possible reason may involve the pattern of motion itself, which may be close to a random movement for an agent.

7. Conclusions and Future Directions

In this paper, we proposed and formalized a new MAPD variant in which a team of agents has to carry out its MAPD tasks in an environment with other external moving agents. Three different solving approaches have been proposed according to three possible levels of knowledge that team agents have of external agents. Interesting directions of future work include (i) the management of deadlocks, e.g., those conflicts that cannot be solved without violating the no-interference assumption; and (ii) a more in-depth investigation of the real-time approach.

Acknowledgments

Benedetta Flammini is financially supported by the ABB – Politecnico di Milano Joint Research Center. This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

References

[1] H. Ma, J. Li, T. Kumar, S. Koenig, Lifelong multi-agent path finding for online pickup and delivery tasks, in: Proc. AAMAS, 2017, pp. 837–845.

[2] P. R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Mag. 29 (2008) 9–20.

[3] R. Morris, C. S. Pasareanu, K. Luckow, W. Malik, H. Ma, T. S. Kumar, S. Koenig, Planning, scheduling and monitoring for airport surface operations, in: Proc. AAAI Workshop on Planning for Hybrid Systems, 2016, pp. 608–614.

[4] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, Cobots: Robust symbiotic autonomous mobile service robots, in: Proc. IJCAI, 2015, pp. 4423–4429.

[5] D. Silver, Cooperative pathfinding, in: Proc. AIIDE, 2005, pp. 117–122.

[6] W. Budiharto, A. Santoso, D. Purwanto, A. Jazidie, Multiple moving obstacles avoidance of service robot using stereo vision, TELKOMNIKA 9 (2011) 433–444.

[7] J. Kim, Y. Do, Moving obstacle avoidance of a mobile robot using a single camera, Procedia Eng. 41 (2012) 911–916.

[8] C. Fulgenzi, A. Spalanzani, C. Laugier, Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid, in: Proc. ICRA, 2007, pp. 1610–1616.

[9] M. Liu, H. Ma, J. Li, S. Koenig, Task and path planning for multi-agent pickup and delivery, in: Proc. AAMAS, 2019, pp. 1152–1160.

[10] D. Silver, Cooperative pathfinding, in: Proc. AIIDE, 2005, pp. 117–122.

[11] Z. Bnaya, A. Felner, Conflict-oriented windowed hierarchical cooperative A*, in: Proc. ICRA, 2014, pp. 3743–3748.

[12] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, S. Koenig, Lifelong multi-agent path finding in large-scale warehouses, in: Proc. AAAI, 2021, pp. 11272–11281.

[13] B. Flammini, D. Azzalini, F. Amigoni, Multi-agent pickup and delivery in presence of another team of robots (extended abstract), in: Proc. AAMAS, 2023, pp. 2562–2564.