Multi-Agent Path Finding for Online Warehouses – An Overview

Adam Harmanec

Abstract

This article gives a short overview of the current state of research in the field of the online warehouse MAPF problem. In this class of problems, agents in a simulated warehouse environment need to be assigned to tasks with one or more endpoints and, without collisions, traverse along it's path. New tasks can be added at any time and they should be completed as effectively and as soon as possible. Different variants of the problem as well as different conditions and objectives can be defined.

Introduction

The Multi-Agent Path Finding (MAPF) problem has been studied for a relatively long time. Traditionally, it has been viewed as a single-shot problem where each agent moves from it's location to a different one while avoiding collisions with other agents. This does not very accurately capture the requirements of many real-world applications such as automated robotic warehouses.

This article provides an overview of a specific part of the research conducted in the field of "Lifelong" or "Online" MAPF. In this problem, new tasks can be added to the system at any time and the limited number of agents should complete them as soon as possible. Other related problems, such as the so-called "Dynamic" MAPF, are also being actively studied, but these are beyond the scope of this article.

Research Summary

The first major attempt to apply AI techniques and algorithms in autonomous warehouses was made by Kiva Systems (Wurman, D'Andrea, and Mountz 2008). Unfortunately, their publications did not provide very detailed information on the specific implementation.

The initial formalization and scientific study of the "Online Warehouse" version of MAPF was performed by (Čáp, Vokřínek, and Kleiner 2015). They proposed a definition for *well-formed infrastructures* and COBRA, a novel on-line multi-robot trajectory planning algorithm that is complete in such settings.

The next progress in this field was made by (Ma et al. 2017). The article formalized a definition for the Multi-Agent Pickup and Delivery (MAPD) problem along with a definition for *well-formed MAPD instances*. Additionally, Ma et al. presented two decoupled MAPD algorithms, Token Passing (TP) and the improved Token Passing with Task Swaps (TPTS), that solve all *well-formed MAPD instances* as well as a centralized MAPD algorithm (CENTRAL) without that guarantee.

The TP algorithm was later improved by (Ma et al. 2018) with the introduction of the Safe Interval Path Planning with Reservation Table (SIPPwRT) search algorithm. The resulting MAPD algorithm TP-SIPPwRT takes kinematic constraints into account and computes continuous agent movements while still being complete for all *well-formed MAPD instances*.

The state of the art algorithms were further refined by (Grenouilleau, van Hoeve, and Hooker 2019) who introduced a *multi-label A** (MLA*) algorithm and a new centralized heuristic for assigning available agents to tasks both of which significantly improve the solution quality and computation time.

A different approach was suggested by (Liu et al. 2019b) for large-scale multi-agent systems while still working with a notion of a *well-formed infrastructure*. They suggested partitioning the environment into sectors and employing a high-level centralized planning algorithm using a traffic heat-map and a low-level decentralized path planning algorithm using a road topology map. Additionally, a strategy for resolving communication failures was proposed.

Previously, task allocation was done without taking agent collisions into account. (Henkel, Abbenseth, and Toussaint 2019) jointly solved the problem of multi-agent task allocation and path planning. The proposed baseline Task Conflict-Based Search (TCBS) was proved to be optimal for the Combined Task Allocation and Path Finding (CTAPF) problem.

(Kou et al. 2019) focused on another objective to minimize the idle time of stations in a sortation center bringing the theoretical methods closer to real-world applications. They developed two offline algorithms that use the proposed ITO and PITO flow networks that can be extended to solve the online TAPF problem.

Having only one or two target locations can be limiting so (Semiz and Polat 2020) studied a variant of the MAPF prob-

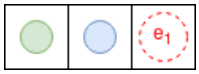

Figure 1: The figure shows an unsolvable problem instance with a green agent with target endpoint e_1 and another blue agent that is free.

lem with multiple delivery locations (MAPF-MD). Semiz and Polat introduced the Multiple Delivery Conflict-Based Search algorithm (MD-DCBS) that solves MAPF-MD instances.

Problem Definition

Generally, the system environment is a continuous 2-D coordinate space or a discrete undirected connected graph. In the environment, there is a fixed number of endpoints, sometimes divided into task and agent endpoints.

A (possibly fixed) number of agents is placed into the environment. Each agent has a location in a given time step, which is either an XY coordinate or a vertex. The body of the agent occupies a point, a closed disk or a single vertex.

In each time step, an agent can stay in place or move at some speed in any direction. In a graph environment, the agent can move to another vertex that is connected by an edge to it's current vertex. While moving, the agents must avoid collisions. Two agents collide if their bodies intersect or if they occupy the same vertex at the same time. Other types of collisions can be defined (Stern et al. 2019).

A task is an ordered set of locations that need to be visited by an agent. The task is completed once an agent visits all of the locations and it is then removed. An agent is called free if it isn't currently executing any task. Tasks can be assigned to free agents automatically or the task allocation can be considered part of the problem. All unassigned tasks are placed into a task set. The task set may start with any number of tasks and new tasks can be added at any time.

The objective is to complete the tasks as quickly as possible. The performance of an algorithm is usually measured by the system's throughput, i.e. by the average number of time steps needed to complete a task after it was added to the task set. The problem is solved if all tasks are completed in a finite number of steps.

Well-Formed Infrastructure

Because not all instances are solvable (Figure 1) and some are only solvable by offline solvers (Švancara et al. 2019), multiple articles define a variant of a *well-formed infrastructure* that can always be solved by a complete online algorithm.

The exact definition differs between articles but the main points are the following:

1. There is a finite number of tasks.
2. For any two endpoints, there exists a valid path that traverses no other endpoints.

   Sometimes an extra condition is added:

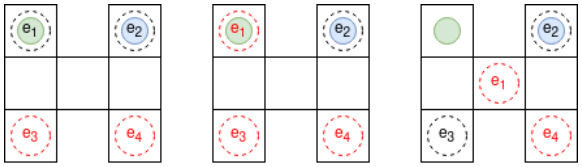

Figure 2: The figure shows three problem instances with a blue and a green agent and four endpoints e_1 to e_4. Task endpoints are red and non-task endpoints are **black**.

3. There is at least the same number of non-task endpoints as the number of agents. (In other words, there is always a place where an agent can wait so that it doesn't interfere with the paths between task endpoints that are required by the previous point)

Figure 2 shows three problem instances. The first environment is well-formed. The second one is well-formed only if the third condition is omitted, because there are two agents but only one non-task endpoint. The last environment is not well-formed because, for example, all paths from e_2 to e_3 traverse e_1.

Algorithms Overview

In this section, we take a more in-depth look at some of the algorithms and methods mentioned in the section **Research Summary**.

The Continuous Best-Response Approach

COBRA (2015) is a decentralized multi-agent path planning algorithm. It operates on a continuous 2-D space and only allows relocation tasks with one endpoint (i.e. an agent has to move from it's current location to a different one). In the general formulation, it is assumed that each agent is able to compute an optimal collision-free path to the target destination.

Distributed token-passing is used as a synchronization mechanism. The token represents a synchronized shared block of memory that stores the paths of all agents. Only one agent can hold the token at any point in time and only that agent can read or change it's content.

The basic algorithm outline is as follows. At the beginning, all agents acquire the token and set their path to stay at their current positions indefinitely. Then, when assigned a task, an agent acquires the token, computes a trajectory to the task's endpoint, releases the token and starts following it.

In practice, the continuous 2-D space representation isn't very practical for computing paths so it is discretized and represented as a graph. In such a representation, the agent path can be found using the Dijkstra's shortest-path algorithm.

Čáp, Vokřínek, and Kleiner (2015) proved that COBRA is complete in *well-formed infrastructures*. That is true for both the continuous and discretized space representation.

The proposed method has a relatively low quadratic complexity in the number of robots for the handling of a single

task. Unlike some reactive techniques (van den Berg et al. 2011), it is complete in most man-made environments.

It's disadvantage is that it cannot be easily extended to complete tasks with more than one endpoints (e.g. for when an agent needs to pickup and deliver a package in a warehouse), because it does not take into account that an agent might be occupying an endpoint that is part of a task assigned to another agent.

Token Passing and Token Passing with Task Swaps

TP and TPTS (2017) build up on some of the same concepts as COBRA. It was however designed for the MAPD problem so each task consists of two endpoints (pickup and delivery) and an agent has to traverse them in a given order. It uses a discrete graph representation of the environment. The algorithm also assumes that the environment is *well-formed* including extra the 3rd point of the definition.

As the name suggests, both TP and TPTS use distributed token-passing in a similar way to COBRA. Both of the algorithms start by initializing the token with trivial trajectories where the agents stay in their location for the next time step.

In TP, all free agents sequentially acquire the token and search for tasks such that no paths of other agents end in any one of the task's endpoints. If there are such tasks, the agent chooses the closest one and starts executing it. If there is no such task and the agent is in a task endpoint, it moves to another free endpoint. Otherwise it remains in it's location.

TPTS is very similar to TP but it allows a free agent to swap tasks with another agent that is moving to the first endpoint of a task if the free agent can reach it faster. The other agent is then given the token and tries to assign itself to a new task.

Both TP and TPTS solve all *well-formed MAPD instances* (Ma et al. 2017).

The discussed algorithms are decentralized and while remaining complete and relatively effective, they require lower compound computation times than some centralized methods. TPTS facilitates more communication between agents and so is more effective with the cost of being more computationally expensive.

In some instances, TP produces a better solution than TPTS. Figure 3 shows such an example. The left figure shows the environment before the start of the simulation. The figure in the center shows the paths chosen by TP if the green agent is granted the token first. The resulting service time is two. The right figure shows the paths chosen by TPTS if the green agent is granted the token first. The resulting service time is three (the average of one and five).

Token Passing - Safe Interval Path Planning with Reservation Table

TP-SIPPwRT (2018) transforms TP into a continuous environment while taking kinematic constraints into account. In standard TP, the shortest path is usually computed using A* that searches in both space and time. That approach presumes that agents move at constant speeds, ignores rotations and acceleration. SIPPwRT, an improved version of SIPP

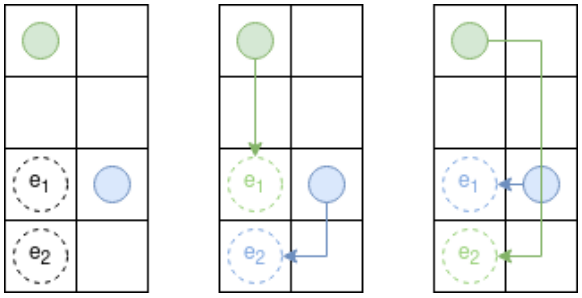

Figure 3: The figure shows a problem instance with a green and a blue agent and two endpoints e_1 and e_2.

(Phillips and Likhachev 2011), computes paths for continuous forward movement and point turns. Additionally, agents are assumed to have a disc-shaped body with an arbitrary radius.

SIPP discretizes the environment into cells and for each cell it keeps track of (safe) time intervals when it is empty and time intervals when it is occupied. To find a path, a simple A* search in this representation is performed and the visited cells are updated so that they track the traversal of that path. This search is quite effective because a lot of the A* branches can be pruned. It is always preferable for an agent to arrive at a cell at an earlier time so all search paths that arrive at that cell later can be ignored.

SIPPwRT improves upon SIPP by employing a reservation table, a more efficient data structure that speeds up the operations needed by the TP-SIPPwRT algorithm. SIPPwRT returns time-optimal collision-free paths (Ma et al. 2018) and so if we use it instead of A* in TP, the resulting TP-SIPPwRT will still be complete for all *well-formed MAPD instances*.

When the original TP with A* is updated to account for forward movement and point turns, it is less effective than the novel TP-SIPPwRT. It could also be used to generalize other algorithms that work with an idealistic discrete representation and it could also be easily extended to work with tasks with any number of endpoints.

Even though TP-SIPPwRT takes a big step towards modeling real-world agents, it still ignores some kinematic constraints such as acceleration and deceleration.

Multi-Label A*

MLA* (2019) is an extension of the classic A* algorithm that finds a shortest path traversing more than one node. In the original TP implementation, the path search was executed in two steps. First, the shortest path to the pickup endpoint was found and then the shortest path from the pickup endpoint to the delivery endpoint was found and appended to it. This can lead to some artificial constraints, e.g. if the pickup endpoint is the delivery endpoint of some other task, the traditional A* algorithm will discard this path even if the agent could visit and leave the pickup location before the other agent reaches it.

The MLA* algorithm works very similarly to A* but each node of the search graph is labeled to keep track of the end-

point to which the algorithm is currently finding the shortest path. Once the pickup endpoint is reached, the label is updated and the search continues until it finds the delivery endpoint.

The updated MLA* algorithm can help TP find better and shorter paths (Grenouilleau, van Hoeve, and Hooker 2019). Additionally, this approach could easily be extended to tasks with more than two endpoints.

Task Conflict-Based Search

TCBS (2019) solves the problem of task allocation and multi-agent path planning simultaneously. In the previous algorithms, the tasks are usually assigned to the closest agent without taking into account the paths of other agents. In some instances, an agent that is further from the task's first endpoint can actually reach it faster than some other agent that is closer but is blocked by a path of a third agent.

TCBS works the same way as CBS (Sharon et al. 2015), but it operates on the level of task assignments rather than path nodes. Each node of the search tree contains tuples consisting of a task and an agent it is assigned to. Additionally, a node can contain some forbidden configurations (e.g. the presence of an agent at a location at some point in time). Such a tree is searched using the traditional A* with specific rules for node expansion, goal testing, calculating the cost and choosing the node to expand.

The proposed algorithm, although it may not be very efficient, solves the combined problem of task allocation and path finding (CTAPF) optimally and can therefore be used to verify the (sub-)optimality of other algorithms.

Idle Time Optimization Flow Network

ITO (2019) is a flow network used to assign stations (task endpoints) to all agents. In this special version of MAPF, each station (task endpoint) has a queue that can host more than one agent. The goal is to minimize the times when the queues are empty. The maximum flow in the ITO network corresponds to the optimal task assignment.

The task allocation ITO network can be combined with a MAPF flow network (Yu and LaValle 2012) to form the Path Finding with ITO (PITO) flow framework. This framework can simultaneously find task assignments as well as the collision-free paths for all agents to their assigned stations while minimizing the stations' idle times.

This approach is inherently one-shot but it can be used in an online scenario by recalculating the solutions at each time step. When compared to the repeated use of the Hungarian method (Kuhn 2010) to assign tasks to agents, PITO is significantly more effective in minimizing the idle times.

Even though this concrete method was developed with a limited use-case in mind, the idea of using flow networks for both the task allocation and path planning seems very promising for other applications as well.

Multiple Delivery Conflict-Based Search

MD-DCBS (2020) solves the problem of online MAPF with multiple delivery locations. It works similarly to CBS (Sharon et al. 2015) but uses the D*-lite (Koenig and Likhachev 2002) algorithm to traverse the CBS tree because it supports replanning, making it very useful for dynamic environments. Additionally, an extra D*-lite instance is used for each successive destination pair for each agent. CBS then runs on the aggregated paths found by these instances.

MD-DCBS is more effective than running CBS multiple times because D*-lite can cache some information. It is optimal and complete in terms of the planned paths but suboptimal in terms of task allocation.

Conclusion and Future Work

The MAPF problem is currently studied very intensively. Recently, more attention has been paid to the online version of the problem, particularly to the variant reminiscent of the requirements of autonomous robotic warehouses. The methods that were proposed were successfully tested in toy and industry computer simulations and sometimes even with physical robots.

Some algorithms are tailored to a specific application like sortation centers (Kou et al. 2019) but, even more so recently, the suggested approaches are encompassing more and more physical and kinematic constraints of real-world agents.

Some of the areas that could be explored in the future include, but are not limited to:

- Decreasing the computational requirements of centralized solutions,
- increasing the complexity of decentralized methods,
- allowing for more complicated agent tasks (e.g. where an agent can simultaneously execute more tasks),
- adding deadlines, prioritites or other constraints to agent tasks
- and more accurately representing the shape and movement of physical agents.

Appendix

The following list contains some interesting articles that study a different version of the online MAPF problem that didn't thematically fit into this article.

- (Sigurdson et al. 2018) proposes a decentralized real-time heuristic that can be used to navigate agents without the need to find a complete path.
- (Wan et al. 2018) introduces a different type of online MAPF called the dynamic MAPF (DMAPF), where the number of agents can change over time.
- (Liu et al. 2019a) provides several methods for solving the offline version of multi-agent pickup and delivery (MAPD) problem.
- (Švancara et al. 2019) formally defines the different versions of the online MAPF problem and proposes several solvers for the intersection type of online MAPF where agents disappear once they reach their goal.

- (Bogatarkan, Patoglu, and Erdem 2019) solves the dynamic MAPF problem using answer set programming.
- (Han and Yu 2019) solves the dynamic MAPF problem using a decentralized algorithm called DDM.
- (Li et al. 2020) proposes a new approach of tackling the online MAPF problem by decomposing it into a sequence of time-bounded sub-problems.

References

Bogatarkan, A.; Patoglu, V.; and Erdem, E. 2019. A declarative method for dynamic multi-agent path finding. In Calvanese, D., and Iocchi, L., eds., *GCAI 2019. Proceedings of the 5th Global Conference on Artificial Intelligence*, volume 65 of *EPiC Series in Computing*, 54–67. EasyChair.

Grenouilleau, F.; van Hoeve, W.-J.; and Hooker, J. N. 2019. A multi-label a* algorithm for multi-agent pathfinding. In *ICAPS*.

Han, S. D., and Yu, J. 2019. Ddm*: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics. *CoRR* abs/1904.02598.

Henkel, C.; Abbenseth, J.; and Toussaint, M. 2019. An optimal algorithm to solve the combined task allocation and path finding problem. *CoRR* abs/1907.10360.

Koenig, S., and Likhachev, M. 2002. D*lite. In *Eighteenth National Conference on Artificial Intelligence*, 476–483. USA: American Association for Artificial Intelligence.

Kou, N. M.; Peng, C.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2019. Idle time optimization for target assignment and path finding in sortation centers. *ArXiv* abs/1912.00253.

Kuhn, H. W. 2010. *The Hungarian Method for the Assignment Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg. 29–47.

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 Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

Liu, M.; Ma, H.; Li, J.; and Koenig, S. 2019a. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, 1152–1160. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Liu, Z.; Wang, H.; Zhou, S.; Shen, Y.; and Liu, Y. 2019b. Coordinating large-scale robot networks with motion and communication uncertainties for logistics applications. *CoRR* abs/1904.01303.

Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017. Lifelong multi-agent path finding for online pickup and delivery tasks. *CoRR* abs/1705.10868.

Ma, H.; Hönig, W.; Kumar, T. K. S.; Ayanian, N.; and Koenig, S. 2018. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. *CoRR* abs/1812.06355.

Phillips, M., and Likhachev, M. 2011. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, 5628–5635.

Semiz, F., and Polat, F. 2020. A job-assignment heuristic for lifelong multi-agent path finding problem with multiple delivery locations. *ArXiv* abs/2003.07108.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219:40–66.

Sigurdson, D.; Bulitko, V.; Yeoh, W.; Hernández, C.; and Koenig, S. 2018. Multi-agent pathfinding with real-time heuristic search. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Barták, R. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. *CoRR* abs/1906.08291.

van den Berg, J.; Guy, S.; Lin, M.; and Manocha, D. 2011. *Reciprocal n-Body Collision Avoidance*, volume 70. 3–19.

Wan, Q.; Gu, C.; Sun, S.; Chen, M.; Huang, H.; and Jia, X. 2018. Lifelong multi-agent path finding in a dynamic environment. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 875–882.

Wurman, P.; D'Andrea, R.; and Mountz, M. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine* 29:9–20.

Yu, J., and LaValle, S. M. 2012. Multi-agent path planning and network flow. *CoRR* abs/1204.5717.

Čáp, M.; Vokřínek, J.; and Kleiner, A. 2015. Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures. In *ICAPS*.

Švancara, J.; Vlk, M.; Stern, R.; Atzmon, D.; and Barták, R. 2019. Online multi-agent pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence* 33:7732–7739.