Decentralised Multi-Agent Path Planning based on Self-Organised Time Division Multiple Access

By

Runze Yuan

MSc Robotics Dissertation

Department of Engineering Mathematics
UNIVERSITY OF BRISTOL
&
Department of Engineering Design and Mathematics
UNIVERSITY OF THE WEST OF ENGLAND

A MSc dissertation submitted to the University of Bristol and the University of the West of England in accordance with the requirements of the degree of MASTER OF SCIENCE IN ROBOTICS in the Faculty of Engineering.

August 24, 2023

Declaration of own work

I declare that the work in this MSc dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Name and Date

Acknowledgement

I would like to thank ...

Abstract

Abstract should give a short summary of the motivation, the approach and important insights and results.

Number of words in the dissertation: .... words.

Contents

List of Tables

1 Introduction

1.1 Brief Introduction

The main idea of this project is to attempt to utilise the principles of a decentralised channel sharing communication protocol (*Self-organised Time Division Multiple Access*, STDMA[1]) to achieve collision-free movement among multiple decentralised agents on a 2D plane.

1.1.1 Principles of STDMA

In STDMA, a 1D channel is represented with repeating frames that are consisted of discrete time slots.

STDMA enables agents to achieve **S**elf-organised **T**ime-**D**ivided **M**ulti-**A**ccess within the channel. That is, each agent independently finds a slot which is uniquely its own and uses this slot to broadcast its message.

The key idea of STDMA is to determine empty slots and apply for them.

This idea also works for collision-free moving on 2D plane —— to determine free space and apply for usage.

For detailed explanation about STDMA, please see **PLACEHOLDER**.

1.1.2 The 2D Plane

The 2D plane used in this project is represented with discrete pixels / grids, just like the STDMA protocol represents continuous time with discrete slots.

For such a representation of a 2D plane, there is a term called **grid world**[2].

For detailed assumptions, please see **PLACEHOLDER**.

1.1.3 Agents

To meet the requirement of decentralisation, the agents are assumed to be identical.

The agents can:

- Broadcast and receive messages in the given channel, but cannot do both at the same time (i.e., cannot be listening while speaking or speaking while listening).
- Move only one step / one grid in the map in one time step.

For detailed assumptions, please see **PLACEHOLDER**.

1.2 Aims

- Based on the idea of STDMA, design an algorithm for decentralised agents to achieve colission-free movement and space sharing on 2D plane.
- Evaluate the advantages and drawbacks of the designed algorithm, and therefore get a better understanding on the problem that the alrogithm aims to solve.

1.3 Objectives

1. Implement original STDMA communication protocol with ROS2, use nodes in ROS2 as agents in the channel, achieve self-organised channel sharing and communicating among agents.
2. Design the specific algorithmic content for agents to achieve collision-free movement.
3. Implement the designed algorithm with ROS2, use ROS2 nodes as agents moving on the 2D plane.
4. Build proper test scene and performance evaluator that could extract metrics (makespan, average finish time, etc.) from simulations.
5. Examine the advantages, disadvantages and limitations of the designed algorithm, summarize the results from observations.

1.4 Motivation

The motivation of this project is to answer a question which is inspired by [3], which is:

> **What would happen if use STDMA for dencentralised 2D resource sharing and path planning?**

And this quesiton could be seperated to the following two parts:

1.4.1 Why STDMA?

The reason for using STDMA is its **characteristics**[4]:

1. **Deterministic**: Agents arrange their data transmission based on a determined timetable.
2. **Decentralised**: Agents listen to the channel first, then independently seek and allocate free slots for themselves to use.

These characteristics are useful for multiple agents to achieve collision-free (use free slots only), slf-organised (find slots on its own) moving and resource sharing.

There are also **reasons that making this challenging**:

1. **Dimensional Difference**: STDMA is designed for sharing discrete time slots (1D), and cannot be directly applied for resource sharing on a 2D plane. Modification is needed for 2D application.
2. **Moving by Grids**: In the communication scenario, agents don't have destiniations in the timetable, i.e., don't need to move to specific slot in the timetable. But that's different for 2D space moving, where agents have their destinations and need to move grid by grid to reach their goals. Agents could easily be trapped in situations of inefficiency or even deadlock situations [5], [6].

These characteristics and challenges are making this topic interesting and worthy for investigation.

Figure 1.1: Kiva system operating in Amazon warehouse.[2]

1.4.2 Why decentralised 2D resource sharing and path planning?

There are many algorithms aim to solve this problem, and their shared aspect is a **focus on the movement of multiple decentralised agents within a grid world**.

To easily get a better understanding of this problem, please refer to this website[1]. This website presents the problem scenarios and their solutions through a simple and engaging animation (estimated time required: 1∼2 min). For detailed explanation, please see **PLACEHOLDER**.

For such situations, **there are myriad corresponding real-life problems and applications**.

- **Warehouse Automation**: Automated pick-pack-and-ship system in warehouse (like Fig 1.1.) [7]. Delivering items in sorting station[8], [9].
- **Intersection Management**: Coordinate autonomous vehicle movement through intersections [10].
- **Robot Fleet**: Automating fleets of autonomous robots like forklift fleets [11], [12].
- **Agents in video games and CGIs**: Flock simulating and animating[13], [14].
- **Swarm Robots**: Controlling self-organised robot swarms[15].

[1]https://primalgrid.netlify.app/primal
[2]https://spectrum.ieee.org/interview-brad-porter-vp-of-robotics-at-amazon

In general, the application of this algorithm is very broad. Furthermore, in the context of Industry 4.0 and flexible manufacturing, it has even better prospects for application[16]–[18], as centralized control is gradually becoming inadequate to meet new production needs.

2 Literature Review

This section will provide the details of the STDMA[1] protocol and a review of the related *Distributed Model Predictive Control* (DMPC) algorithms.

2.1 STDMA Explained

As previously mentioned in Section 1.1, STDMA stands for ***Self-organised Time-Divided Multiple Access***, and it allows multiple agents to share the same channel for communication without centralised control. The main assumtion of this protocol is that all agents have synchronised clocks. In practice, this is achieved through GPS[19].

The core idea of STDMA could be summarised as follows: Represent continuous time with repeating frames that are consisted of discrete time slots. While agents are always listening to messages from the channel, they look for free slots to occupy, and therefore use the occupied slots to broadcast their own data.

Agents using STDMA have **four phases**, which are arranged in chronological order as follows:

1. **Initialisation**: Agents in this phase have not yet joined the network. The device listens to an entire frame and determines current allocation of each slot.

2. **Network Entry**: Randomly choose an unallocated slot to broadcast their existance and reserve one slot for the next phase. If the message sent didn't collide with others (i.e., only one agent which is myself choose to use this slot for entering), then the entering is successful. If the entering failed, reverse to the previous phase.

3. **First Frame**: Use the slot reserved in the previous phase to reserve more slots for themselves. The number of reserved slots depends on the size of the data packet that the agent needs to send in each frame.

4. **Continuous Operation**: Use the previously reserved slots to work normally. If some slots are released or more slots are needed, reapply for slots.

Although the description above omitted some details (such as slot choosing strategy, calculation of required number of slots, etc.), it is clear that **the core idea of STDMA is the strategy of finding and reserving unallocated slots**.

This protocol also has several limitations, such as: **(1)** Collision in entering: In the Network Entry phase, multiple agents may accidentally choose the same unallocated slot for entering and broadcast their existance. **(2)** Capacity: When slots are not enough, conflicts would inevitably occur. There are some studies [20], [21] that proposed improvements for its limitations, but improving STDMA is not the focus of this project.

For detailed implementation, please see **PLACEHOLDER**.

2.2 Distributed Model Predictive Control

Distributed Model Predictive Control or DMPC is a control strategy based on constraints and optimization[22]. DMPC not only inherits the capability of MPC to handle constraints but also possesses a decentralised characteristic, making it apt for addressing multi-agent swarm motion control problems, such as: formation control[23], path tracking[24], path planning[25], and obstacle avoidance[26]. The commonality of such problems is that agents are subject to static constraints in a dynamic environment.

The DMPC algorithm applied in this domain can be summarised as: Agents communicate with other agents and solve a finite horizon optimization problem in their local MPC controller, using the solution for action control. The optimization criteria are typically specified based on the agent's task.

In [27], the optimization problem is represented as 0–1 integer linear programming, and then a method based on the alternating direction method of multipliers (ADMM) is developed to coordinate AGVs and alleviate computational burden. [28] parameterises the trajectory of vehicles using polynomial splines, reducing the number of optimisation variables and achieving path tracking for vehicle platooning. [29] utilises a hierarchical DMPC (HDMPC) path prediction method with a trigger mechanism to control the behaviour of drones post-obstacle avoidance, achieving a more efficient multi-agent multi-different-target search in unknown areas compared to conventional methods. [30]addresses the issues of vehicle platooning control and obstacle avoidance by modelling the motion of an agent swarm based on animal behaviours. It then transforms the local MPC problem of the agent into a closed-form convex optimization. [31] utilises the grey wolf optimizer (GWO) to enhance the solution process of the DMPC controller, improving convergence speed and achieving 3D spatial path planning for multiple drones. [32] introduces a drone path planning algorithm based on predicting the motion of surrounding drones without the need for communication. However, this comes at the cost of increased computational load.

From the aforementioned papers, it's evident that research in this domain predominantly focuses on the representation of the problem and the methods to solve optimisation challenges. The primary distinction between such algorithms and the subject of this study is: the mode of communication between agents is often expressed as a constraint incorporated into the algorithm design[33], while the actual implementation of communication is frequently omitted. Common communication topology assumptions include: serial[27], agents communicating with others freely within a certain range[29]–[31], [34], directed graphs[35], [36]. On top of this, bandwidth limitations are occasionally overlooked[34]. Moreover, common target scenarios (such as formation control and path tracking) differ significantly from the target scenario of this project.

Despite the aforementioned differences, the essence of the algorithm implemented in this paper aligns with that of DMPC: agents communicate amongst themselves and, based on the information from other agents, individually solve an optimisation problem within a finite horizon. A portion of the generated solution is then taken as their own control sequence.

2.3 Multi Agent Path Finding

The classical definition of the *Multi Agent Path Finding* (MAPF) problem is: **Multiple agents moving step by step in an undirected graph in discrete time**[37], which is precisely the target scenario of this project.

The MAPF algorithm can be divided into **several categories: classical, heuristic, rule-based, and machine learning combined**. Among these methods, those that are rule-based are the closest to the approach of this project.

- Classical: The most popular classic MAPF algorithm is the *Artificial Potential Field* (APF) algorithm, which put force fields around obstacles and goals to generate a force to guide the agents. APF methods does not guarantee collision-free and is prone to fall into local minima. There are still some research today dedicated to improving the performance of the APF algorithm, with a primary focus on optimizing the potential field function[38]–[40].

- Heuristic algorithms mainly include A* and its variants (like D* [41]). Heuristic algorithms mainly operate on the principle of searching for the optimal solution. Many researchers combine A* with other algorithms to achieve better performance and goes beyond the original A* [42], [43].

- ML Combined: The machine learning algorithm mainly used in this field is reinforced learning, and usually combined with other non-RL techniques [44]–[47].

- Rule-based: The rule-based methods are based on the idea that each agent follows a set of identical local decision-making principles (which means it is decentralized) and usually improve algorithm performance by modifying the rules. Most of these algorithms are bio-inspired, including *Particle Swarm Optimisation* (PSO)[48], [49], *Genetic Algorithm* (GA)[50], [51], *Ant Colony Optimisation* (ACO)[52], [53], *Pegion Inspired Optimisation* (PIO)[54], [55] and *Grey Wolf Optimisation* (GWO)[56]–[58]. Apart from these bio-inspired algorithms, there is also a category of algorithms called push-and-swap[59], [60]. They address problems through two kinds of actions: push (move towards the goal) and swap (exchanging positions between agents).

Generally speaking, researches in this area are trade-offs between completeness, computational complexity, and optimality. There is currently no perfect algorithm for this question. Besides, direct communication between agents isn't always a key component in these methods (e.g. [41], [46]).

3 Algorithm

This section provides detailed information on algorithm and its implementation.

Environment

- **Hardware:** ROG Zephyrus M16 Laptop
  - CPU: 11th Gen Intel(R) Core(TM) 17-11800H @ 2.30GHz
  - GPU: NVIDIA GeForce RTX 3060 Laptop GPU (unrelated to the experiment, information provided just for content completeness)
- **Software:**
  - OS: WSL2 (Ubuntu 22.04 LTS) in Windows 11 23H2
  - Implementation Platform: ROS2 Humble, all codes written in python

3.1 Communication with STDMA

This is the **first part of the algorithm: self-organised communication between agents**.

In STDMA, agents share a single channel by autonomously determining the serial speaking order. The approach to determining the speaking order involves agents independently assigning the right to use available time slots within the channel.

3.1.1 Synchronised Clock

Assumption 1: Agents have synchronised clocks.

In practice, the synchronised clock is typically implemented with GPS. In the implemented simulation of this paper, it's achieved using a ROS2 publisher and a topic. A dedicated ROS2 node periodically toggles its member boolean value and publishes this boolean value to the clock topic each time it's toggled. This creates a square wave clock signal with a 50% duty cycle in the clock topic. Agents obtain the synchronised clock signal by subscribing to the clock topic.

3.1.2 Discretisation of Channel Time

Agents consider a complete clock signal cycle (a high and a low) as one slot and consider a specific number of slots as a frame. Time frames continuously cycle, providing a continuous and reusable slot resource for agents to use.

> **Assumption 2:** The number of slots in a frame is predefined within the agents, and this parameter value is the same for all agents.

Please note that it is not necessary for each agent to have the same starting point for their frame. In other words, agents are permitted to have different frame starting point offsets.

The exact middle of each slot is the timing for message transmission. This divides a slot into two parts: before sending the message and after sending the message.

3.1.3 State Machine for Channel Allocation

Agents employing STDMA go through four phases[1], with each phase representing a stage in an agent's integration into the network. Consequently, a state machine can effectively manage the process of an agent joining the network. Through progressive state transitions, the agent incrementally integrates into the network and secures its slot.

ADD PIC HERE

The implementation of this state machine in this paper is as follows:

1. **Listening**

   In this state, the agent's objective is to determine the current channel slot allocation by 'Listening' to the channel. Agents in this state have not yet joined the network.

   Slot allocation determination: In a slot:

   - If only one message was sent: the slot is occupied by the sender.

- If multiple messages were sent or no message was sent: the slot is considered free.

The initial state of an agent is this state.

Transition Condition: After listening to a complete frame, the agent attempts to exit this state.

- $\rightarrow$ Entering: There are one or more unoccupied slots within the frame.
- $\circlearrowleft$ Listening: There are no free slots remaining in the frame. This suggests that the channel's capacity has been fully utilized, and the agent can only stay 'Listening' and remain idle.

2. **Entering**

   In this state, the agent attempts to occupy a free slot in order to try 'Entering' the network.

   Transition Condition: The agent tries to occupy a random free slot that was identified during the 'Listening' state. Within this slot, the agent **transmits its unique ID** in an effort to occupy the slot.

   - $\rightarrow$ Checking: The agent transitions to the 'Checking' state immediately after sending the occupation attempt message.

3. **Checking**

   In this state, the agent has just broadcast a message in a specific slot and must ascertain its ownership of that slot, based on the quantity and content of messages received within it. By 'Checking' its ownership of that slot, the agent transits its state.

   Note that both agents that have not yet entered the network and those that have will transition to this state after sending a message.

   Transition Condition: Transition the state based on the number of messages received in the slot where a message has been sent.

   - $\rightarrow$ In: If only one message is received and it's from itself, it indicates that the agent has ownership of that slot, meaning it is 'In' the network.

- → Listening: In any other situation: The agent does not have ownership of that slot and should try to secure a slot again.

  Explanation: All other situations include:

  - Didn't receive its own message: This suggests that either the broadcast was unsuccessful or the reception was unsuccessful (e.g., due to hardware damage or scheduled broadcasting being prevented, etc.). In both situations, we do not want the agent to enter the network, as its communication function may not be consistently reliable.
  - Multiple agents sent messages within one slot (i.e. collision):
    Collision Scenario 1: Multiple agents attempting to join the network coincidentally select the same slot. While such collisions are unavoidable within the context of this algorithm, they only occur among agents that have not yet entered the network and thus do not impact other normal communications.
    Collision Scenario 2: A collision occurs between an agent that has already joined the network and one that has not. In theory, this type of collision should not occur, as agents not yet in the network should only attempt to occupy unoccupied slots. However, this situation may arise if an agent misses a clock pulse. Missing a clock pulse can cause an agent to fall out of synchronization with all other agents, potentially leading to conflicts and accidents. Efforts should be made to prevent this situation as much as possible.

4. **In**

   In this state, the agent is 'In' the network and should stay in the network until it releases its slot by stopping its regular message transmission.

   Upon reaching their destinations, agents will shut down and release the slot they occupied in the channel.

   Transition Condition:

   - → Checking: After publishing a message, verify ownership of that slot.

ADD PIC HERE

3.1.4 Summary

This achieves agent self-organization and collision-free channel sharing. The channel's time is represented as repeating frames containing a specified number of slots. An agent attempts to join the network by trying to occupy one slot within a frame, the joining process is managed and implemented by a state machine.

3.2 Path Planning and Sharing

3.2.1 Planning Framework and Definition

Map

The map is a grid world with obstacles, and:

Assumption 3: All obstacles in the map are stationary.

Agent Capabilities and Constraints

Capabilities:

- Agents can choose to either remain stationary or move to an adjacent non-diagonal grid cell at each time step (corresponding to a slot in the STDMA section).
- Agents can execute their own plans with complete accuracy.
- When plan is exhausted or have no plan, agent executes a default behavior: remains stationary.

Assumption 4: The agent's movement is not affected by external disturbances, and its prediction of its own motion is completely accurate.

Constraints

- Agents should not collide with each other.
- Agents should not use positions on the map that are occupied by obstacles.

Collision Definition

In this paper, the following two scenarios are considered as collisions:

- Two or more agents are located at the same 2D position on the map at the same time.
- Two agents swapping their positions. That is, at time t, agent A is at position a, and agent B is at position b. At time $t+1$, agent A is at position b, while B is at position a.

ADD PIC HERE

Plan Definition and Constraints

Definition

A plan is a sequence of 3D coordinate points. Each coordinate point comprises a 2D location and a specific time, essentially indicating both the spatial and temporal aspects.

Constraints

1. The points in the plan must align with the agents' capabilities and constraints (collision-free, move no more than one grid at a time).
2. The first point in the plan must be a non-diagonal adjacent point to the agent's initial position at the start of planning, and it must also satisfy the agent's other constraints.

Plan Broadcasting Method and Time Window for Planning

Broadcasting

Through the channel facilitated by STDMA, agents broadcast their individual plans. This means agents broadcasting their plans in the middle of each slot allocated to them within the channel.

Note that during plan formulation, only a sequence of 2D coordinates is generated, and time is implicit in the planning process. The time dimension of the plan is added by the recipient upon receiving the plan. As a result, only a set of 2D coordinate sequences needs to be transmitted each time broadcasting.

Time Window for Planning

As previously mentioned, each agent publishes their message (their plan and ID) in the middle of the slot allocated to it. Based on this rule, for every agent, there exists a periodically repeating time window where all variables (map, others' plans) related to planning are constant, and the agent's own plan can be immediately published as soon as it is formulated.

This time window is the first half of the agent's own slot. Within this interval, no other agents will broadcast new plans, and as soon as this interval ends, the agent can immediately publish its own plan. Given that this time window is a segment within a slot, it will occur once in each frame.

ADD PIC HERE

Model Prediction

In this paper, the model predictions have been significantly simplified by assuming that agents can execute their own plans with absolute accuracy.

Agents follow the rules below to predict their own and other agents' movements:

1. A plan that has been published will definitely be executed.
2. If an agent exhausts its plan before its planning window arrives, the agent will execute a default behaviour: remaining stationary. This will continue until the agent publishes a new plan.

The default behaviour decouples frame length from plan length. This is because, even if the agent cannot generate a plan sufficient for a whole frame, it can still execute the default behaviour, waiting for the next planning window to arrive.

And this default behaviour presents an issue: it doesn't adhere to the non-collision constraint. An agent, while idling in place, might occupy a position that another agent intends to use in the future, leading to collisions. To address this concern, the length of the plans that all agents can publish is set to a uniform value.

In this way, since agents plan sequentially and with a uniform plan length, the end of each plan will always coincide with the leading edge of the system's overall horizon. This allows them to execute the default behaviour (idling in place) at this point without conflicting with other plans, since no other plans exist within this time frame. When an agent formulates its plan, it takes into consideration the idling of other agents after their plans have concluded, ensuring it avoids collisions with agents that are stationary.

ADD PIC HERE

For this reason, the pathfinding algorithm always starts from scratch. Since each plan must be of the same length, the plan length is constrained by the worst-case scenario. In the worst case, an agent must abandon all previous work and start planning anew. Due to this, a warm start for the pathfinding algorithm does not improve algorithm performance or the planning horizon.

Note, when the length of a published plan exceeds the frame length, it isn't contradictory to the idea that a plan once published will definitely be executed and not inheriting the results from the previous plan (cold start planning). Take the sequential plans of a single agent as an example: after the first plan is made and published, other agents, in their subsequent planning, will draft their paths based on this released plan to ensure they don't occupy positions specified in the first plan. During the second planning, even if it doesn't directly inherit the unused portions of the first plan, since the planning at this point can't possibly be better than the previous one, the resulting solution remains the same. This is akin to implicitly inheriting the parts of the plan that were published but not utilised.

In the algorithm's implementation, this functionality is achieved by imposing a globally uniform cap on the plan length (referred to as the 'plan length limit') and setting an upper boundary for the foreseeable future that can be predicted during a single computation, known as the 'planning horizon', based on an agent's computational capacity.

The process of an agent entering the map

Since agents' awareness of each other relies solely on planning and consensus on default behaviour, an agent is not known to others before it enters the network and successfully generates a plan, which can lead to potential collisions. Therefore, when an agent starts up, it does not occupy a position on the map; instead, it follows the following process to enter the map:

- The agent attempts to join the network.
- The agent tries to generate a plan with its starting point as the origin. If successful, it executes according to the plan (thus acquiring a place on the map). If unsuccessful, it repeatedly

attempts to generate this plan.

- The agent operates continuously on the map.

Due to the subsequent experiments setting the starting points at the edge of the map, this process can be regarded as the agent's attempt to enter a specific area's network and subsequently its entry into that area.

The process of an agent exiting the map

Once an agent detects its arrival at the goal, it exits the map: it no longer holds a position within the map and relinquishes its slot in the channel (by ceasing to transmit information to the STDMA channel).

Due to the subsequent experiments setting the goals at the edge of the map, this process can be regarded as the agent exiting a specific area's network and leaving that area.

3.2.2 Path Planning Function and Implementation

At the start of each planning window, the path planning function is called to generate the upcoming movement plan for the agent.

Inputs of the function

- map, agent's current position, goal, plans of other agents

  Foundational information required for the planning process.

- first ever plan

  This is a boolean value that indicates whether the current plan is the agent's first plan upon entering the map, with an initial value of False.

  This parameter is prepared for the aforementioned process of an agent entering the map. When this value is true, the first point of the plan must be the agent's starting point. When this value is false, the first point of the plan is selected from points adjacent to the agent's current position. Once the agent successfully generates its first plan, this boolean value is set to True.

- planning horizon

  The upper limit of the time range explored within a single plan session. If a possible plan reaches this length, it will no longer be extended, and other plans that have not reached this length will be extended instead. If all possibilities within the horizon have been explored or a plan that reaches the goal has been generated, planning is stopped, and the plan that allows the agent to approach the goal most closely among the generated potential plans is returned.

- plan length limit

  The purpose of this parameter is to limit the maximum length of the returned plan. Only the portions of the generated plan that do not exceed this length are returned.

Algorithm 1: Path Planning Function

```
Function path_plan( map, current position (2D coordinate), goal (2D coordinate), plans
  of other agents, first ever plan, planning horizon, plan length limit):
    current position ← set (current position, time = 0, cost = time + Manhattan distance
      (heuristic cost) of current position and goal, path = empty list)
    frontier ← [current position]
    visited ← set()
    path list ← []
    while frontier do
        current point ← pop the point with the lowest cost in frontier
        current path ← the path corresponding to the current point
        current time ← the time corresponding to the current point
        if current path length = planning horizon then
            path list append current path
        if current path length > planning horizon then
            continue
        if current point = goal and current plan length ≤ plan length limit then
            return current path
        if first plan then
            neighbour ← start of the agent
        else
            neighbour ← non-diagonal neighbour points of current point
        foreach neighbour do
            if (neighbour, current time+1) is not in visited and neighbour is valid then
                frontier append set(neighbour, time = current time + 1, cost = time +
                  Manhattan distance of neighbour and goal, path = current path +
                  neighobur)
                add (neighbour, current time+1) to visited
    if path list then
        path ← the plan with the last point having the smallest Manhattan distance to the
          goal
        return The portion of the plan that does not exceed the plan length limit
    else
        return None
```

Output of the function:

If the agent is currently inside the map, generate a plan of the specified length starting from a position adjacent to the current position.

If the agent is not inside the map, generate a plan of specified length starting from the specified starting point.

If there is no plan meets the requirements, return None.

Actions after reaching the goal:

The agent shuts itself down, stops sending messages to the channel (meaning it gives up its slot in the channel), and disappears from the map.

3.3 Summary

3.3.1 Assumptions:

1. Agents have synchronised clocks.
2. All obstacles in the map are stationary.
3. The agent's movement is not affected by external disturbances, and its prediction of its own motion is completely accurate.
4. There always exists at least one continuous 2D path in the space for the agent to use
5. The number of slots in a frame is predefined within the agents, and this parameter value is the same for all agents.
6. The number of predicting horizon upper limit is predefined within the agents, and this parameter value is the same for all agents.

3.3.2 Implemented Functions:

After the agent is activated, it autonomously attempts to join the current communication network. After joining the network, it tries to enter the map from the specified starting point. Once inside the map, based on the movement plans of others and map information, it continuously searches for a collision-free continuous 2D coordinate sequence that can bring it closer to its goal, and broadcasts its own movement plan. When it reaches the goal, it releases its position in the communication network, exits the map and ceases activity.

4 Results

4.1 Experiment Design

Map

All experiments in this paper were conducted on a map simulating a warehouse scenario. This map was selected from a widely-used benchmark set[61][1] for testing, due to its appropriate size and obstacle distribution.

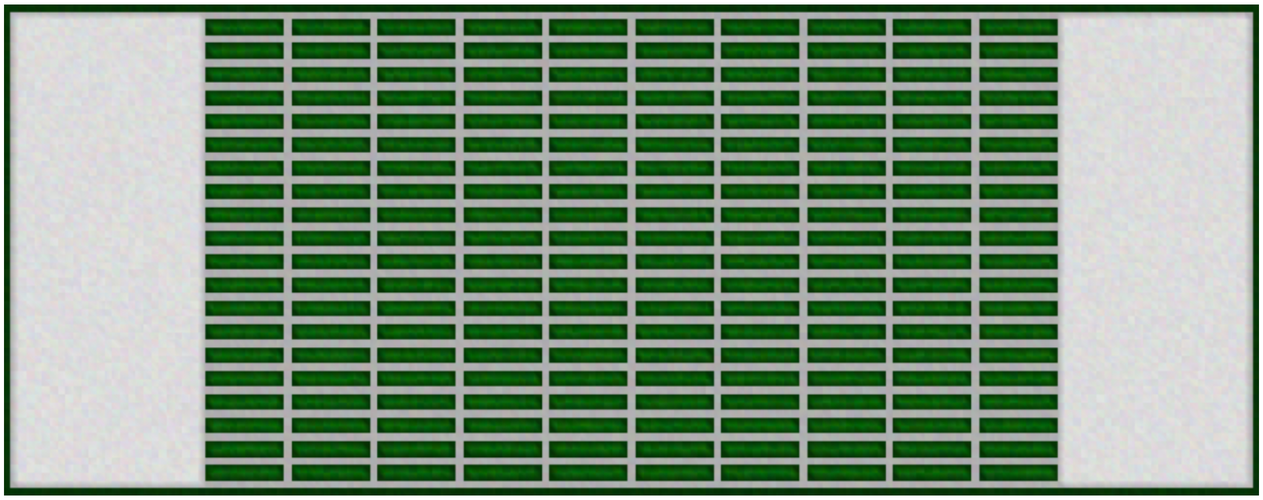


Figure 4.1: The map used in the experiments. The outer border is not included in the map.

The size of the map is 161×63 (the outermost green frame is not part of the map), with green representing obstacles and white representing passable areas. All narrow corridors (channels at the border and between obstacles) are one grid / pixel wide.

[1]https://movingai.com/benchmarks/mapf.html

Agent Initialisation and Behaviour

The starting points and their corresponding goal points for agents are pre-generated and stored in a launch file. During each experiment, all agents are started simultaneously, and then sequentially read the start-goal point pairs from this file and assign them to the agents as they are launched. (e.g. if 50 agents are launched in a experiment run, the first 50 start-goal pairs are read from the launch file and assigned to the agents in order. The same approach is applied for different numbers of agents.) The starting points are located in the outermost circle of the map (the outermost green obstacle border in Fig 4.1 is not part of the map). The sequence of starting points is stored in the launch file after being randomly shuffled. The goal points are on the opposite side of the map, and the mapping relationship between the starting points and the goal points is as follows:

$$x_{goal} = width(map) - x_{start} \tag{4.1}$$

$$y_{goal} = height(map) - y_{start} \tag{4.2}$$

Where x refers to the horizontal coordinate, and y refers to the vertical coordinate. The effect of this mapping is shown in the following figure: **ADD PIC HERE**

After successfully joining the network, the agent attempts to generate a plan starting from the given start. If no plan meeting this requirement is available, the agent is not present on the map (since the start is on the map boundary, it can be considered as waiting outside the map). If the agent reaches the goal, it will shut down and disappear from the map (since the goal is also on the map boundary, it can be considered as exiting the map).

Parameters and Metrics

There are four adjustable parameters in the algorithm:

1. plan length limit: The upper limit of the length of the plan returned in a single planning session. Plans that exceed this length will be discarded.

2. planning horizon: The upper limit of the future horizon that can be predicted in a single planning window, which is also the maximum length of the plan that can be generated in a single planning window.

3. frame length: The length of the frame in the communication channel.

4. agent number: The number of agents.

There are five performance metrics:

1. total path efficiency ratio: sum of actual path lengths / sum of optimal path lengths. The optimal path is determined using A* algorithm under the assumption that there are no other agents present in the map.

2. average path efficiency ratio: average of individual agents' (actual path length / optimal path length).

3. final agent arrival time: the time at which the last agent reached its goal.

4. average agent arrival time: the average arrival time for individual agents.

5. average network join time: the average time spent by agents to join the network.

4.2 Experiment Results

4.2.1 Quantitative Results

Analyze the impact of parameter changes on system performance using quantitative results.

plan length limit and planning horizon

These two parameters mainly impact the generated plans. The plan length limit sets the global maximum length for the returning plans, while the planning horizon determines the upper length boundary for plans that can be generated.

A longer planning horizon is more favorable, as it allows agents to make effective actions within this range. Beyond this scope, agents can only execute default behaviour (remaining stationary) until next planning window arrive (see Section 3.2.1).

On the other hand, the plan length limit might have diverse effects when set to longer or shorter values:

- When set to longer values: Sharing longer plans among agents, might enhance collaboration efficiency and overall effectiveness by enabling agents to coordinate over extended horizon.

- When set to shorter values: Sharing shorter plans among agents, could lead to more agile agent movements, potentially increasing efficiency by allowing for greater flexibility.

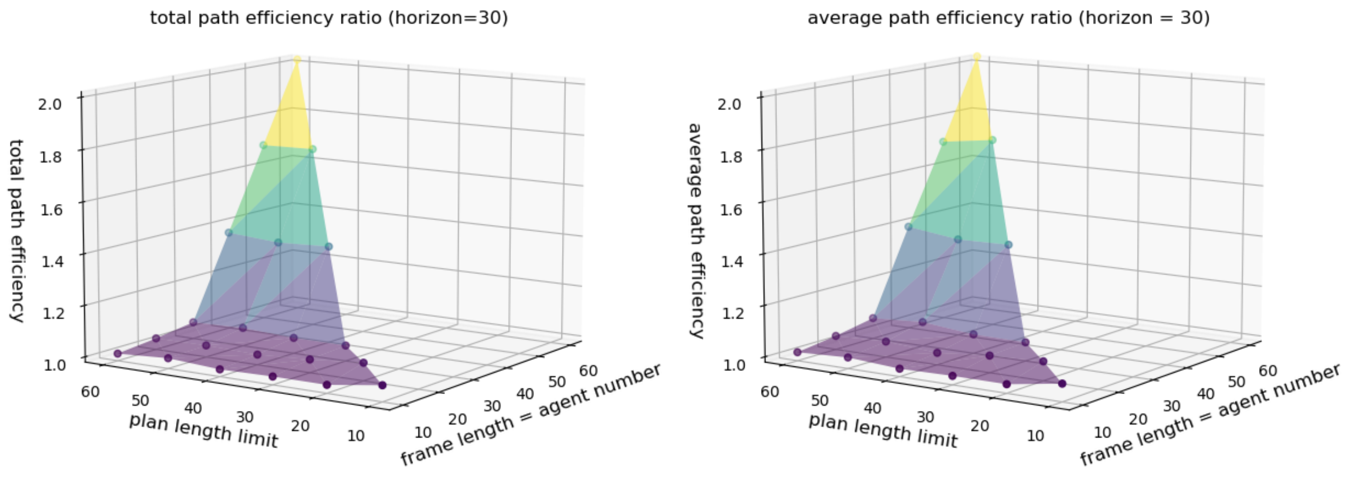
Additionally, the plan length limit should be at least equal to the frame length. This is because when the plan length is shorter than one frame, the agent will remain stationary, significantly reducing efficiency.

To validate these hypotheses, two groups of experiments are conducted as follows:

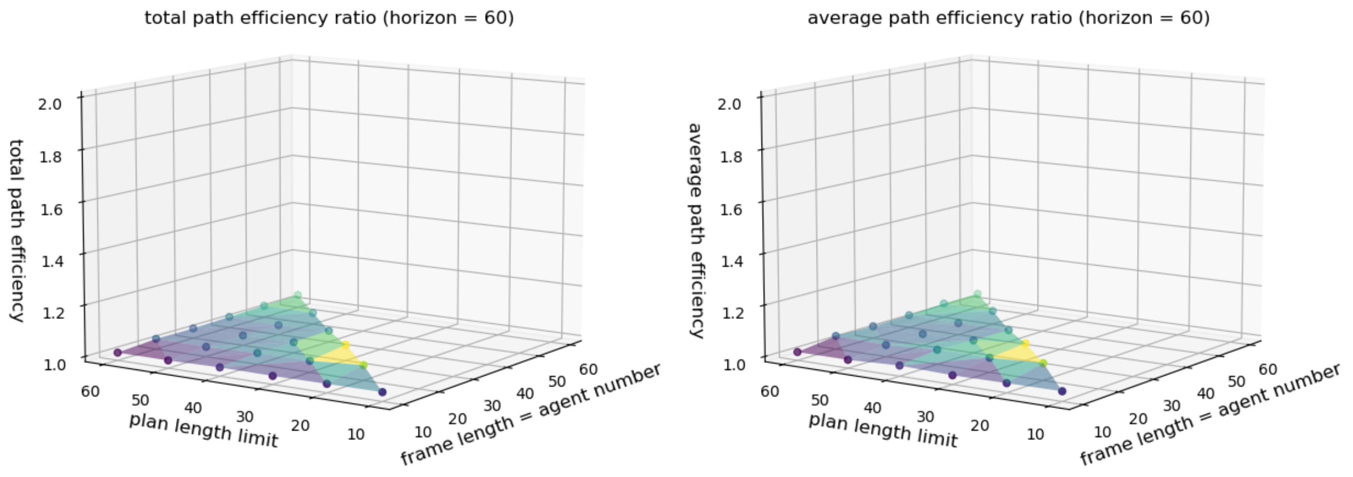
- Group 1.1: agent number = frame length, planning horizon = 30, plan length limit and frame length explore a parameter space ranging from 10 to 60 in increments of 10. Notably, the plan length limit is always maintained to be greater than or equal to the frame length.

- Group 1.2: agent number = frame length, planning horizon = 60, plan length limit and frame length explore a parameter space ranging from 10 to 60 in increments of 10. Notably, the plan length limit is always maintained to be greater than or equal to the frame length.

The random factor in the experiment is the time spent for agents to join the network, as agents randomly choose free slots in the channel when joining. Due to the considerable time required for the experiment (one slot takes 1 second), each point in the parameter space is tested only once. Although this approach doesn't eliminate all randomness, the trends indicated in the experimental results are still quite clear.



(a) Experiment Group 1.1: path efficiency versus plan length limit (planning horizon = 30)



(b) Experiment Group 1.2: path efficiency versus plan length limit (planning horizon = 60)

Figure 4.2: Performance versus Planning Horizon and Plan Length Limit.

In Figure 4.2, the relationship between two metrics (total / average path efficiency) associated with path quality and the planning horizon, as well as the plan length limit, is illustrated. The results can be divided into two parts: the portion where the frame length exceeds the planning horizon, and the portion where the frame length does not exceed the planning horizon.

- Frame length exceeds planning horizon:

  The slope in Figure 4.2a represents this segment of results. Within this portion, the frame lengths are all greater than the planning horizon, indicating that agents remain stationary during the final part of each frame.

  In this section, the relationship between frame length and planning horizon predominantly influences path efficiency. It's evident that both the total path efficiency and the average path efficiency nearly equal the ratio of frame length to planning horizon (as seen at frame length = 60, the ratio is near 2.0, for instance).

- Frame length does not exceed planning horizon:

  In Figure 4.2, all the remaining flat sections represent the results of this segment. From the graph, it's evident that when the frame length does not exceed the planning horizon, path efficiency remains almost unchanged regardless of variations in the plan length limit. The z-axis values in all these flat sections of the graph are below 1.05.

  This indicates that either the plan length limit has negligible impact on path efficiency, or the map is too open for the agents involved in the experiment.

  The explanation for the map being too open: As mentioned in 4.1, although all the corridors have a width of only one agent's width (one grid / pixel), due to the specific distribution of obstacles, when a path is blocked, an agent can simply choose to turn at different intersections to avoid collision, this allows them to avoid the occupied corridors without sacrificing path efficiency. The setting of agent starting and ending points, as described in 4.1, might contribute to this situation. All agents need to cross the map, and there are many optimal routes for crossing this map, which could exacerbate the open nature of the map.

Overall, these experimental results indicate that, within the context of this test scenario, the plan length limit is not a critical factor. As long as the frame length does not exceed the planning horizon, path efficiency can be maintained at a level close to optimal.

frame length and agent number

These two parameters are related to the system capacity. Frame length refers to the quantity of slots within a frame, while agent number represents the number of agents initiated simultaneously at the beginning of simmulation run.

The advantages and disadvantages exist for both longer and shorter frame lengths.

- Longer frame lengths: there is more channel space available for agents, allowing a larger number of agents to join the channel. However, if collisions occur during the joining process, according to the protocol, agents need to re-listen to the entire frame to reconfirm the allocation and make another attempt to join. This time duration for re-listening increases with the longer frame length. Consequently, longer frame lengths do not necessarily lead to better outcomes.

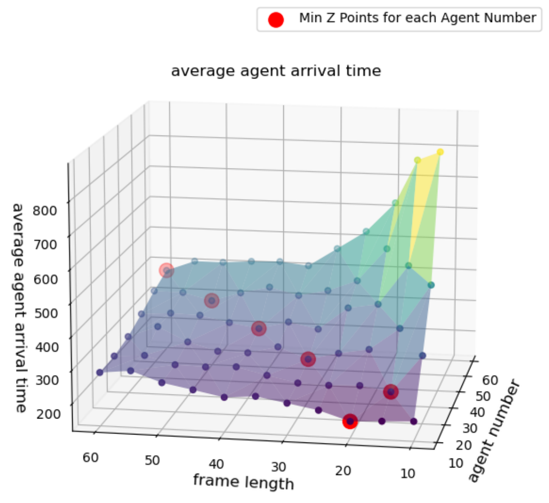
- Shorter frame lengths: When the frame length is shorter, there is limited channel space, which can result in a significant number of agents lingering in the channel joining phase. This can lead to reduced efficiency. Although this scenario may not suffer from the prolonged re-listening disadvantage seen with longer frame lengths, the inherent channel capacity is already low due to the shorter frame length.

From this perspective, it is apparent that the magnitude of the frame length is relative to the agent number.

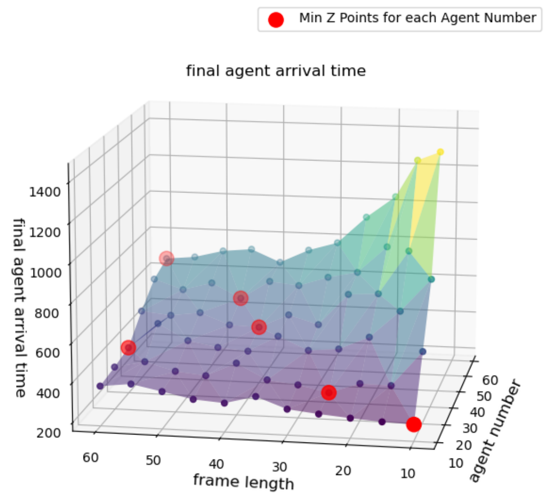
To investigate the impact of varying frame lengths on system performance, the following experiments are conducted:

- Group 2: The parameter space will be traversed twice with the following settings: planning horizon = 60, plan length limit = 60 (these two are independent parameters), agent number ranging from 10 to 60 with a step size of 10, and frame length ranging from 10 to 60 with a step size of 5. Certain points in the parameter space (specifically, those with agent numbers significantly surpassing frame lengths) are omitted due to excessive time consumption and clearly inadequate performance.

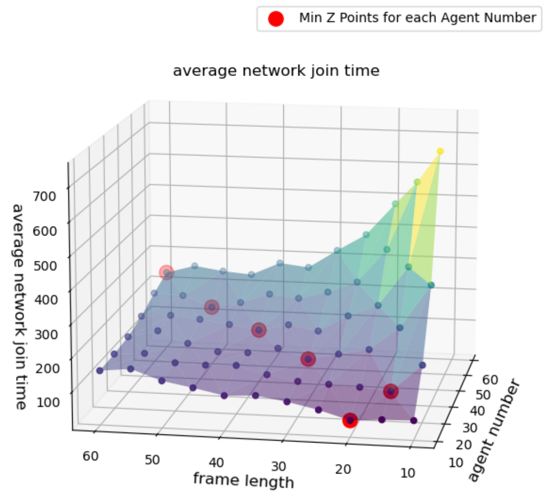
Similar to the previous group of experiments, the same random factor exists in this group as well: the randomness associated with agents randomly selecting free slots to attempt joining the channel upon entry. Due to the considerable time required for the experiment (with each slot taking 1 second), the parameter space is traversed only twice, and the average values of performance metrics are taken as results. Although this approach doesn't entirely eliminate randomness from the outcomes, the performance trends with parameter variations are still quite evident.



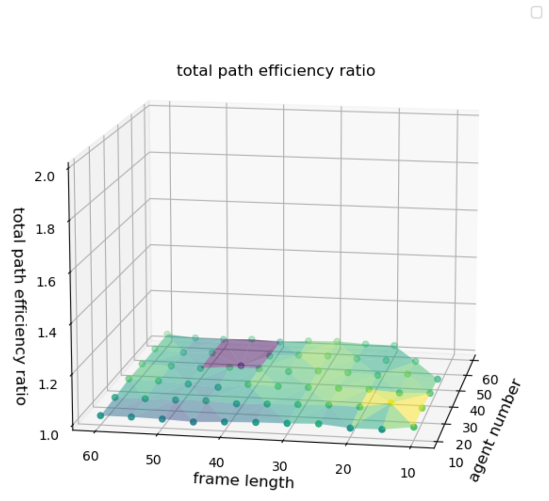
(a) Average agent arrival time results



(b) Final agent arrival time results



(c) Average network join time results



(d) Total path efficiency ratio result

Figure 4.3: Experiment Group 2: Performance versus agent number and frame length.

There are only two factors that can influence the most important parameter, which is the arrival time. These factors are: (1) the duration it takes for an agent to join the network, and (2) the efficiency of the agent's path.

In these two factors, the path efficiency of the agent has almost no impact on the arrival time in this experimental scenario, as its variation across the entire parameter space is minimal. As can be seen from Figure 4.3d, the efficiency of the agent's path is consistently high (all the points on the z-axis in the figure are below 1.05, which is in line with the results from the first group of experiments). The similarity in shape between Figure 4.3a, Figure 4.3b, and Figure 4.3c further supports this observation. While the final arrival time may deviate slightly due to higher randomness (as opposed to the average arrival time which is generated by averaging), the overall trend remains consistent.

Consequently, the performance parameters in this scenario are primarily determined by the time it takes for an agent to join the network. The variation in performance across the parameter space can be explained from this perspective.

As can be seen from Figure 4.3a, 4.3b, 4.3c, for each agent number, the points of optimal performance tend to concentrate where the agent number and frame length values are close or equal. Using the line where agent number = frame length as a reference, the results can be divided into two parts for discussion:

- Agent Number < Frame Length:

  This part corresponds to the upper-right part of each figure in Fig 4.3. When the agent number increases relative to the frame length, there is a significant negative impact on performance.

  This is because, in this part, the channel capacity is relatively limited, and a portion of agents must remain idle until available channel space emerges, allowing them to start their operations.

- Agent Number > Frame Length

  This part corresponds to the bottom-left part of each figure in Fig 4.3. As the frame length increases relative to the agent number, the performance gradually declines.

  In this part, the channel has enough capacity for all the agents to operate in the network at same time. However, according to the protocol, if multiple agents collides in the channel (accidentally trying to occupy the same free slot when entering), they must wait and listen through an entire frame before attempting again. This mechanism means that an excessively long frame length has a negative impact on the agent join time.

This explains why, for each agent number, the points with the optimal performance are mostly when the frame length equals the agent number or slightly higher than the agent number. In these cases, there is enough room in the channel to avoid having agents solely in idle mode, while the frame length is also not excessively long, preventing a prolonged process of retrying to join the network.

4.2.2 Deadlock Situation

A deadlock situation arises only under exceptionally strict conditions.

Condition

When the following conditions are simultaneously met, a deadlock between agents occurs:

1. Two agents are positioned within a sufficiently long corridor that's only one agent wide, each agent starts from different side of the corridor and needs to reach the opposite end of this corridor.

2. The length of plans published by agents = frame length, i.e., planning horizon $\geq$ frame length and plan length limit = frame length.

3. When one agent begins its planning, the other agent's plan should already be halfway executed, with only half a frame length remaining.

    - This yet-to-be-executed half should contain the current or adjacent position of the planning agent, which means this unexecuted part interferes the new plan of the planning agent.

    - The content of this half frame length plan should be solely directed towards its goal (advancing).

Explanation

Because the characteristic of STDMA, the two agents make plan in turns, as their planning time window is their slot in the repeating frame. Additionally, agents publish plans of the same length throughout the entire system (see Section 3.2.1).

In this scenario, an agent's plan could be divided into two parts:

- Retreating: Due to collision-free constraints, when the other agent is moving forward in its plan, the agent must retreat.
- Advancing: When planning and in the later part of the planning horizon, the other agent's plan has exhausted (and at this point, the planning agent doesn't assume that another agent will idle after its plan ends, because a sufficient plan has already been published for it to use until its next planning window.). The planning agent could fill the rest part of its new plan with actions of proceeding toward its goal.

When the lengths of the advancing and retreating segments are equal, a dynamic deadlock scenario emerges: both agent spend half of their plans for advancing and the other half for retreating.

Solution

Due to the stringent requirements of this deadlock scenario, breaking any one of the three conditions will suffice.

For example:

- Use odd-numbered frame length: This breaks the tie between advancing and retreating, because these two parts cannot be of equal length.
- Avoid using long narrow corridors: Add additional area to the corridor, don't force agents to push each other.
- Allow agents to publish longer plans: If agents could publish plans that longer than frame length, this deadlock scenario could not happen. If one agent reserves an extensive span of space in advance (simply by broadcasting its plan) for advancing, the other agent will be

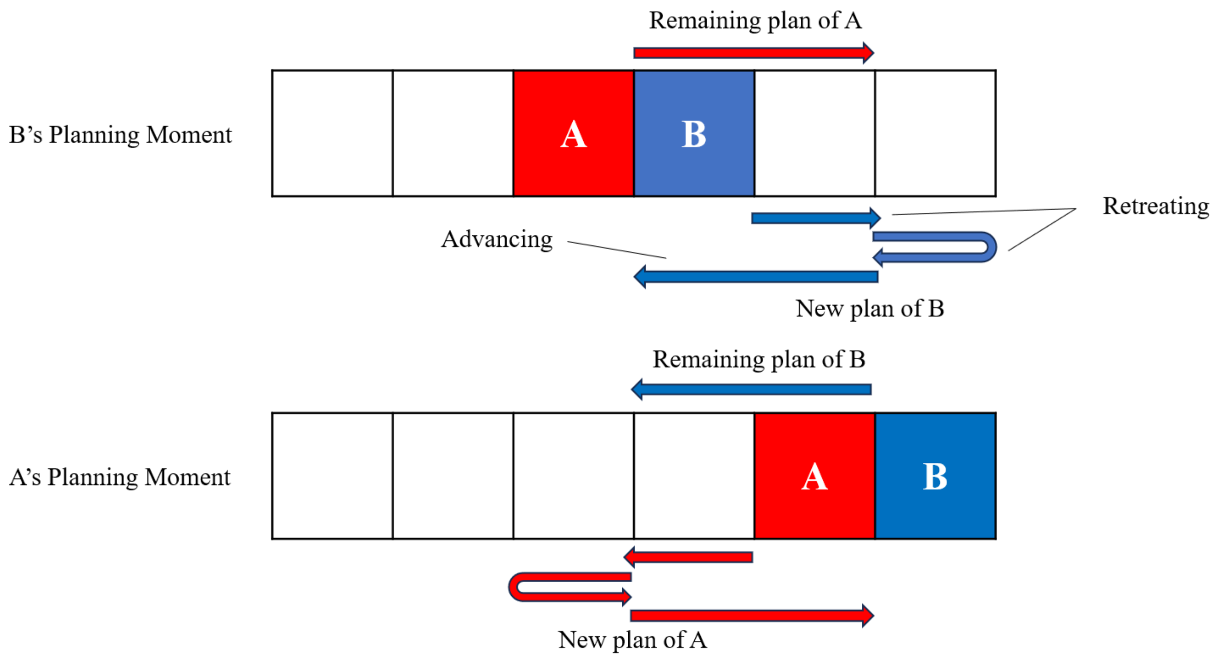


Figure 4.4: Example of a deadlock situation (frame length = 4, agent publishes plan with a length of 4).

compelled to retreat in the corresponding part of its plan. This results in a highly imbalanced advance-retreat cycle, facilitating a swift exit from the corridor area.

This situation didn't show up in the quantitative result generating. This is mostly because the obstacles on our test map were relatively small compared to the planning horizon applied, so one agent could book all space in the corridor for advancing in one plan. Also, the conditions for this situation is quite strict.

4.2.3 Failure Situation

In the deadlock scenario, if there is a third agent, it could result in a failure: agent in the middle cannot find any possible plan.

Condition

1. Three agents are positioned within a sufficiently long corridor that's only one agent wide.

2. All agents are trying to get from one side of the corridor to the other, and one agent's starting and ending points are the complete opposite of the other two agents'.

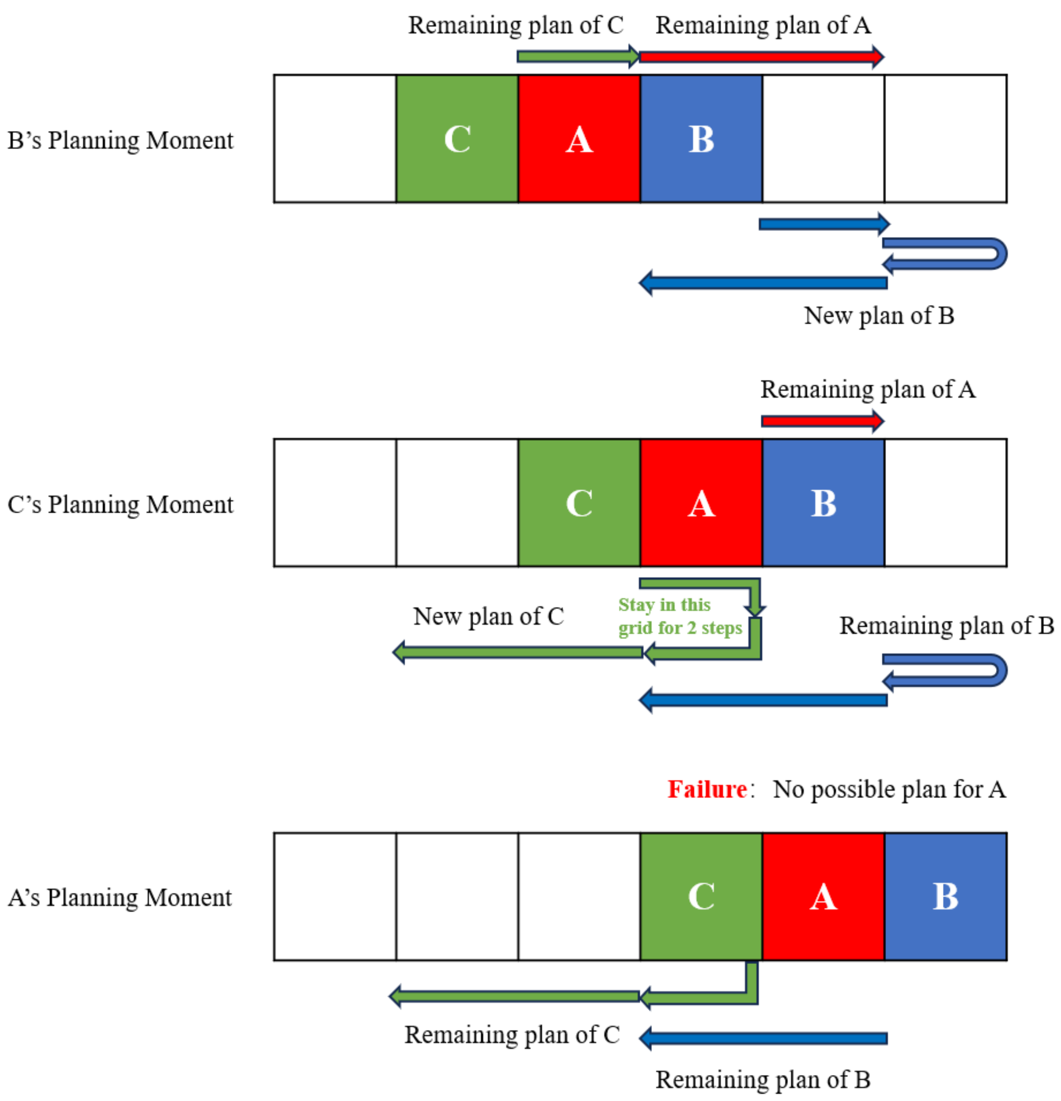


Figure 4.5: Example of failure. Due to the squeeze of agent C, agent A cannot find any possible plan (frame length = 4, agent publishes plan with a length of 4).

Explanation

The primary reason this situation arises is due to the conflict between the two agents on the same side of the corridor. Since agents plan sequentially, the following scenario will always occur among the two agents on the same side:

- When the outermost agent (C in Fig 4.4) is planning, the inner agent's plan exhaust within the planning horizon, leading the outermost agent to believe it can keep moving forward, subsequently advancing to a position adjacent to the agent on the opposite side.
- When the inner agent (A in Fig 4.4, the one in the middle of the three agents) plans next, there isn't enough space for it due to the squeeze from the outer agent.

Solution

- When the plan length that an agent can publish is sufficiently long relative to the length of the corridor, allowing an agent to reserve most of the space in the corridor in advance for its progression can prevent this situation (as seen in previous experiments where this situation didn't arise).
- Modify the consensus among agents. Currently, an agent only assumes other agents will remain stationary after their plans run out if the published plan length of the other agents is less than one frame (see Section 3.2.1). If, during planning, agents always assume that other agents will stay in place once their plans are exhausted, this situation can be avoided. However, it might lead to potential decreases in path efficiency.
- Avoid using overly long narrow corridors.

4.2.4 Local Optimal Trap

When an obstacle is sufficiently large such that an agent cannot surpass it within the planning horizon, the agent can become ensnared in a local optimal trap. The agent will remain stationary at the position inside the obstacle that is closest to the goal, as it cannot reach a position closer to the goal within a single planning horizon.

This occurs because each time the agent plans, it only explores potential plans within a certain horizon and then selects the one where the final position is closest to the goal. If the planning horizon is long enough, this problem could be overcome.

This situation didn't show up in the quantitative result generating. This is mostly because the obstacles on our test map were relatively small compared to the planning horizon applied.

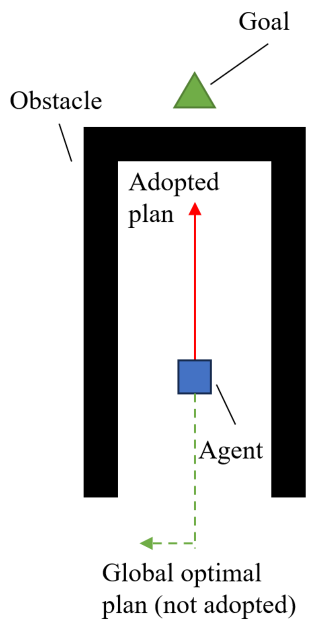


Figure 4.6: An agent trapped in a local optimum due to a short planning horizon, preventing its arrival.

5 Discussion and Conclusion

The conclusion needs to provide

- A short summary (What has been done and what are the main results)
- Limitations of your work, where applicable.
- Discussion of your work in the bigger picture (How does this contribute to the research field?)
- Future work (What could be next steps in this work?). Remember to keep future work realistic. A good approach is to discuss what the next progression of this project would be, and to justify why this would be interesting.

You will find it easier to write your conclusion if you copy-and-paste your *Aims, Objectives*, and any research questions or hypotheses you stated. You can then discuss each of these explicitly in turn, and how you were able to answer them or complete them successfully. When things have not gone as well as you would have hoped, demonstrate your critical thinking and reasoning to analyse the short-comings of your project - to demonstrate that you understand the underlying causes and that you could conduct good futurework from this learning experience.

A Appendix

This is optional. Not every report needs an appendix If you have additional information like code pieces, long tables, etc. that would break the flow of the text in the report, you can put it here.

References

[1] T. Gaugel, J. Mittag, H. Hartenstein, S. Papanastasiou, and E. G. Ström, *In-depth analysis and evaluation of self-organizing tdma*, 2013. DOI: `10.1109/VNC.2013.6737593`.

[2] B. Guillaume, World representation in artificial intelligence, in *Encyclopedia of Computer Graphics and Games*, N. Lee, Ed. Cham: Springer International Publishing, 2020, pp. 1–10, ISBN: 978-3-319-08234-9. DOI: `10.1007/978-3-319-08234-9_308-1`. available from: `https://doi.org/10.1007/978-3-319-08234-9_308-1`.

[3] E. F. Asadi and A. Richards, Scalable distributed model predictive control for constrained systems, *Automatica*, vol. 93 2018, pp. 407–414, 2018.

[4] J.-H. Lim, Understanding stdma via computer simulation: Feasibility to vehicular safety applications, configurations, and time synchronization errors, *EURASIP Journal on Wireless Communications and Networking*, vol. 2016 2016, pp. 1–14, 2016.

[5] R. Stern, Multi-agent path finding–an overview, *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures* 2019, pp. 96–115, 2019.

[6] R. Stern, N. Sturtevant, A. Felner, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, 2019, pp. 151–158.

[7] P. R. Wurman, R. D'Andrea, and M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, *AI magazine*, vol. 29, no. 1 2008, pp. 9–9, 2008.

[8] N. M. Kou, C. Peng, H. Ma, T. S. Kumar, and S. Koenig, "Idle time optimization for target assignment and path finding in sortation centers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 9925–9932.

[9] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, Persistent and robust execution of mapf schedules in warehouses, *IEEE Robotics and Automation Letters*, vol. 4, no. 2 2019, pp. 1125–1131, 2019.

[10] K. Dresner and P. Stone, A multiagent approach to autonomous intersection management, *Journal of artificial intelligence research*, vol. 31 2008, pp. 591–656, 2008.

[11] F. Pecora, H. Andreasson, M. Mansouri, and V. Petkov, "A loosely-coupled approach for multi-robot coordination, motion planning and control," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, 2018, pp. 485–493.

[12] J. Salvado, R. Krug, M. Mansouri, and F. Pecora, "Motion planning and goal assignment for robot fleets using trajectory optimization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 7939–7946. DOI: 10.1109/IROS.2018.8594118.

[13] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, *IEEE Transactions on automatic control*, vol. 51, no. 3 2006, pp. 401–420, 2006.

[14] H. Ma, J. Yang, L. Cohen, T. Kumar, and S. Koenig, "Feasibility study: Moving non-homogeneous teams in congested video game environments," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 13, 2017, pp. 270–272.

[15] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, Trajectory planning for quadrotor swarms, *IEEE Transactions on Robotics*, vol. 34, no. 4 2018, pp. 856–869, 2018.

[16] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, Industry 4.0, *Business & information systems engineering*, vol. 6 2014, pp. 239–242, 2014.

[17] D. A. Rossit, F. Tohmé, and M. Frutos, Industry 4.0: Smart scheduling, *International Journal of Production Research*, vol. 57, no. 12 2019, pp. 3802–3813, 2019.

[18] M. De Ryck, M. Versteyhe, and F. Debrouwere, Automated guided vehicle systems, state-of-the-art control algorithms and techniques, *Journal of Manufacturing Systems*, vol. 54 2020, pp. 152–173, 2020.

[19] Z. Fernández, I. Val, M. Mendicute, and E. Uhlemann, "Analysis and evaluation of self-organizing tdma for industrial applications," in *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2019, pp. 1–8. DOI: 10.1109/WFCS.2019.8758039.

[20] H. Mosavat-Jahromi, Y. Li, Y. Ni, and L. Cai, Distributed and adaptive reservation mac protocol for beaconing in vehicular networks, *IEEE Transactions on Mobile Computing*, vol. 20, no. 10 2020, pp. 2936–2948, 2020.

[21] T. Terauchi, K. Suto, and M. Wakaiki, "Harvest-then-transmit-based tdma protocol with statistical channel state information for wireless powered sensor networks," in *2021 IEEE 93rd vehicular technology conference (VTC2021-Spring)*, IEEE, 2021, pp. 1–5.

[22] R. Scattolini, Architectures for distributed and hierarchical model predictive control–a review, *Journal of process control*, vol. 19, no. 5 2009, pp. 723–731, 2009.

[23] S.-S. Liu, M.-F. Ge, and Z.-W. Liu, "Multi-uav formation control based on distributed model predictive control," in *2022 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, IEEE, 2023, pp. 292–297.

[24] Z. Zang, Z. Li, J. Gong, C. Gong, H. Yu, and X. Zhang, "Optimal path tracking controller of multiple unmanned tracked vehicles: A distributed model predictive control approach," in *2022 International Conference on Advanced Robotics and Mechatronics (ICARM)*, IEEE, 2022, pp. 581–586.

[25] S. M. A. Mousavi, B. Moshiri, and Z. Heshmati, On the distributed path planning of multiple autonomous vehicles under uncertainty based on model-predictive control and convex optimization, *IEEE Systems Journal*, vol. 15, no. 3 2020, pp. 3759–3768, 2020.

[26] I. Ravanshadi, E. A. Boroujeni, and M. Pourgholi, Centralized and distributed model predictive control for consensus of non-linear multi-agent systems with time-varying obstacle avoidance, *ISA transactions*, vol. 133 2023, pp. 75–90, 2023.

[27] J. Xin, X. Wu, A. D'Ariano, R. Negenborn, and F. Zhang, Model predictive path planning of agvs: Mixed logical dynamical formulation and distributed coordination, *IEEE Transactions on Intelligent Transportation Systems* 2023, 2023.

[28] J. Matouš, D. Varagnolo, K. Y. Pettersen, and C. Paliotta, Distributed mpc for formation path-following of multi-vehicle systems, *IFAC-PapersOnLine*, vol. 55, no. 13 2022, pp. 85–90, 2022.

[29] L. Zhao, R. Li, J. Han, and J. Zhang, A distributed model predictive control-based method for multidifferent-target search in unknown environments, *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1 2022, pp. 111–125, 2022.

[30] S. M. A. Mousavi, B. Moshiri, and Z. Heshmati, On the distributed path planning of multiple autonomous vehicles under uncertainty based on model-predictive control and convex optimization, *IEEE Systems Journal*, vol. 15, no. 3 2020, pp. 3759–3768, 2020.

[31] C. Yang, Z. Liu, W. Zhang, and W. Yue, "Cooperative online path planning for uavs based on dmpc and improved grey wolf optimizer," in *2022 41st Chinese Control Conference (CCC)*, IEEE, 2022, pp. 5008–5013.

[32] Z. Niu, X. Jia, and W. Yao, Communication-free mpc-based neighbors trajectory prediction for distributed multi-uav motion planning, *IEEE Access*, vol. 10 2022, pp. 13 481–13 489, 2022.

[33] S. Novoth, Q. Zhang, K. Ji, and D. Yu, Distributed formation control for multi-vehicle systems with splitting and merging capability, *IEEE Control Systems Letters*, vol. 5, no. 1 2020, pp. 355–360, 2020.

[34] J. Zheng, M. Ding, L. Sun, and H. Liu, Distributed stochastic algorithm based on enhanced genetic algorithm for path planning of multi-uav cooperative area search, *IEEE Transactions on Intelligent Transportation Systems* 2023, 2023.

[35] Z. Tang, L. Wang, Y. Wang, H. He, and B. Li, Distributed-integrated model predictive control for cooperative operation with multi-vessel systems, *Journal of Marine Science and Technology*, vol. 27, no. 4 2022, pp. 1281–1301, 2022.

[36] Z. Zang, Z. Li, J. Gong, C. Gong, H. Yu, and X. Zhang, "Optimal path tracking controller of multiple unmanned tracked vehicles: A distributed model predictive control approach," in *2022 International Conference on Advanced Robotics and Mechatronics (ICARM)*, IEEE, 2022, pp. 581–586.

[37] R. Stern, Multi-agent path finding–an overview, *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures* 2019, pp. 96–115, 2019.

[38] T. Zhao, H. Li, and S. Dian, Multi-robot path planning based on improved artificial potential field and fuzzy inference system, *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 5 2020, pp. 7621–7637, 2020.

[39] C. He, Y. Wan, Y. Gu, and F. L. Lewis, Integral reinforcement learning-based multi-robot minimum time-energy path planning subject to collision avoidance and unknown environmental disturbances, *IEEE Control Systems Letters*, vol. 5, no. 3 2020, pp. 983–988, 2020.

[40] J. Xue, X. Kong, B. Dong, and M. Xu, Multi-agent path planning based on mpc and ddpg, *arXiv preprint arXiv:2102.13283* 2021, 2021.

[41] H. Li, T. Zhao, and S. Dian, Prioritized planning algorithm for multi-robot collision avoidance based on artificial untraversable vertex, *Applied Intelligence*, vol. 52, no. 1 2022, pp. 429–451, 2022.

[42] H. Li, T. Zhao, and S. Dian, Prioritized planning algorithm for multi-robot collision avoidance based on artificial untraversable vertex, *Applied Intelligence*, vol. 52, no. 1 2022, pp. 429–451, 2022.

[43] Z. Yuan, Z. Yang, L. Lv, and Y. Shi, A bi-level path planning algorithm for multi-agv routing problem, *Electronics*, vol. 9, no. 9 2020, p. 1351, 2020.

[44] H. Shiri, H. Seo, J. Park, and M. Bennis, Attention-based communication and control for multi-uav path planning, *IEEE Wireless Communications Letters*, vol. 11, no. 7 2022, pp. 1409–1413, 2022.

[45] D. Zhu, B. Zhou, and S. X. Yang, A novel algorithm of multi-auvs task assignment and path planning based on biologically inspired neural network map, *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2 2020, pp. 333–342, 2020.

[46] T. Fan, P. Long, W. Liu, and J. Pan, Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios, *The International Journal of Robotics Research*, vol. 39, no. 7 2020, pp. 856–892, 2020.

[47] T. Fan, P. Long, W. Liu, and J. Pan, Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios, *The International Journal of Robotics Research*, vol. 39, no. 7 2020, pp. 856–892, 2020.

[48] E. Mobarez, A. Sarhan, and M. Ashry, “Obstacle avoidance for multi-uav path planning based on particle swarm optimization,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1172, 2021, p. 012 039.

[49] Z. Chen, H. Wu, Y. Chen, L. Cheng, and B. Zhang, Patrol robot path planning in nuclear power plant using an interval multi-objective particle swarm optimization algorithm, *Applied Soft Computing*, vol. 116 2022, p. 108 192, 2022.

[50] J. Song, L. Liu, Y. Liu, J. Xi, and W. Zhai, Path planning for multi-vehicle-assisted multi-uavs in mobile crowdsensing, *Wireless Communications and Mobile Computing*, vol. 2022 2022, pp. 1–21, 2022.

[51] R. A. Saeed, D. Reforgiato Recupero, and P. Remagnino, The boundary node method for multi-robot multi-goal path planning problems, *Expert Systems*, vol. 38, no. 6 2021, e12691, 2021.

[52] G. Yi, Z. Feng, T. Mei, P. Li, W. Jin, and S. Chen, Multi-agvs path planning based on improved ant colony algorithm, *The Journal of Supercomputing*, vol. 75 2019, pp. 5898–5913, 2019.

[53] J. Liu, S. Anavatti, M. Garratt, and H. A. Abbass, Modified continuous ant colony optimisation for multiple unmanned ground vehicle path planning, *Expert Systems with Applications*, vol. 196 2022, p. 116 605, 2022.

[54] D. Zhang and H. Duan, Social-class pigeon-inspired optimization and time stamp segmentation for multi-uav cooperative path planning, *Neurocomputing*, vol. 313 2018, pp. 229–246, 2018.

[55] B. H. Wang, D. B. Wang, and Z. A. Ali, A cauchy mutant pigeon-inspired optimization–based multi-unmanned aerial vehicle path planning method, *Measurement and Control*, vol. 53, no. 1-2 2020, pp. 83–92, 2020.

[56] S. Mirjalili, S. M. Mirjalili, and A. Lewis, Grey wolf optimizer, *Advances in engineering software*, vol. 69 2014, pp. 46–61, 2014.

[57] M. Zhou, Z. Wang, J. Wang, and Z. Dong, A hybrid path planning and formation control strategy of multi-robots in a dynamic environment, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 26, no. 3 2022, pp. 342–354, 2022.

[58] G. Huang, Y. Cai, J. Liu, Y. Qi, and X. Liu, A novel hybrid discrete grey wolf optimizer algorithm for multi-uav path planning, *Journal of Intelligent & Robotic Systems*, vol. 103 2021, pp. 1–18, 2021.

[59] M. Kulich, T. Novák, and L. Přeucil, "Push, stop, and replan: An application of pebble motion on graphs to planning in automated warehouses," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 4456–4463. DOI: 10.1109/ITSC.2019.8916906.

[60] H. Wang and M. Rubenstein, Walk, stop, count, and swap: Decentralized multi-agent path finding with theoretical guarantees, *IEEE Robotics and Automation Letters* [online], vol. 5, no. 2 2020, pp. 1119–1126, 2020. DOI: 10.1109/LRA.2020.2967317.

[61] R. Stern, N. R. Sturtevant, A. Felner, *et al.*, Multi-agent pathfinding: Definitions, variants, and benchmarks, *Symposium on Combinatorial Search (SoCS)* 2019, pp. 151–158, 2019.