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 14, 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

In STDMA, agents share a single channel by autonomously determining the serial speaking order. The method for determining the speaking order involves agents autonomously allocating the right to use free times 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 the toggled value to the clock topic

each time it's flipped. This creates a square wave clock signal with a 50% duty cycle in the clock topic. Other standard 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 as one slot and consider several slots as one frame.

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

Note that it's not required for each agent to have the same frame starting point, i.e., agents are allowed to have different frame starting point offsets.

3.1.3   State Machine for Channel Allocation

Agents using STDMA have four phases[1], with each phase corresponding to a status of an agent joining the network. Therefore, the process of an agent joining the network can be managed using a state machine. As the state transitions progressively, the agent gradually joins the network and obtains its slot.

Before explaining the operation of the state machine, it's necessary to briefly describe how the usage of slots in the channel is represented and determined: Agents publish their ID to the channel (which is a ROS2 topic) in the middle of the slot they occupy. Agents keep listening to the channel (subscribe to the channel topic) to update their slot allocation records. If only one agent sends a message in a slot, that slot is occupied by that agent. If no agents send a message in a slot (i.e., it's unused) or if multiple agents send messages simultaneously (i.e., a collision), then the slot is considered unoccupied. At the end of each slot, agents count the number of messages received during that slot and make the aforementioned determination.

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.

An agent's initial state is this state.

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

- → Entering: There are one or more unoccupied slots in the frame.
- ↻ Listening: No free slot left in the frame. This indicates that the channel's capacity has reached its limit and the agent can only stay 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 an unused random slot. In the middle of this slot, the agent publishes its ID.

- → Checking: Agent transit to the 'Checking' state immediately after the occupation attempt message send.

3. **Checking**

In this state, the agent has just published a message in a certain slot and needs to determine its ownership of the slot based on the number of messages received within it. By 'Checking' its ownership of that slot, the agent transits its own state.

Note, both agents that haven't entered the network and those that have will enter this state after releasing a message.

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

- → 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 indicates that the broadcast is not successful or receiving is not successful (e.g. hardware damaged or scheduled broadcasting prevented, etc.). In both situation, we don't want the agent to enter network, since its communicating function is not always fully functional.

- Multiple agents sent messages within one slot (i.e. collision):

  Collision Scenario 1: Multiple agents trying to join the network happen to choose the same slot. Such collisions are inevitable within the framework of this algorithm, but since they only occur among agents that haven't entered the network, they don't affect other communications.

  Collision Scenario 2: A collision occurs between an agent that has joined the network and one that hasn't. Theoretically, this kind of collision shouldn't happen, as agents not yet in the network should only attempt to secure unused slots. If an agent misses a clock pulse, this situation might arise. If an agent misses the clock pulse, it will result in it falling out of sync with all other agents, potentially leading to accidents. This situation should be avoided as much as possible.

4. **In**

In this state, the agent is 'In' the network and should remain within the network, until it releases its slot by ceasing to send messages regularly.

Transition Condition:

- $\rightarrow$ Checking: After message publishing, check its ownership of that slot.

ADD PIC HERE

3.1.4 Summary

The agent's self-organised collision-free channel sharing is implemented. The channel's time is represented as repetitive frames containing a specified number of slots. Agents join the network by attempting to secure a slot within a frame.

3.2 Path Planning and Sharing

3.2.1 Agents' Mobility

Before explaining the agent's planning strategy, it's necessary to describe the mobility of the agents in this paper:

- In each time step (i.e., an STDMA slot), the agent can choose to remain in its current position or move to an adjacent grid that's not diagonal.

- Agents could execute their plan with complete accuracy. This is to simplify the model.

    Assumption 3: The agent's movement is not affected by external disturbances, and its prediction of its own motion is completely accurate.

3.2.2 Sharing

Agents communicate via the channel shared through the STDMA protocol, sharing their movement plans for the upcoming steps.

In the simulation of this paper, for the sake of implementation convenience, the shared plan is published to a separate ROS2 topic (not the topic used for STDMA). This channel's time slot allocation follows the order determined by STDMA.

3.2.3 Timing for Planning

4 Results

The Results section should provide

- An overview of all obtained results

- An in detail discussion/explanation of all results

- A scientific interpretation of the results

4.1 Common attributes to pay attention to are:

- When comparing plots, keep the scale of axes consistent. To do otherwise is misleading for the reader.

- If you are going to compare separate plots, consider if they can be better evaluated when combined into a single plot.

- When plotting data, particularly the *mean*, ensure that you also plot error bars (or other method) of indicating the distribution.

- If a figure or plot is included, ensure it is referenced explicitly in the body text discussion.

- When a large table of data is included, consider whether it would be better communicated as a box-plot or something similar.

- All axes should be labelled and include units of measurement where applicable.

- All captions and figures should have captions with enough information to be understood at a glance. Do not use captions to provide information that is better placed in the body text.

- Remember to identify result outliers and anomalous data and to attempt an explanation or justification.

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.