UC Irvine
UC Irvine Electronic Theses and Dissertations

Title
Motion Planning for Multi-Agent Systems with Obstacles based on Buffered Voronoi Cell

Permalink
https://escholarship.org/uc/item/7sb01199

Author
MA, MENG

Publication Date
2024

Copyright Information
This work is made available under the terms of a Creative Commons Attribution License, available at https://creativecommons.org/licenses/by/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA,
IRVINE

Motion Planning for Multi-Agent Systems with Obstacles based on Buffered Voronoi Cell

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Electrical and Computer Engineering

by

Meng Ma

Thesis Committee:
Assistant Professor Quoc-Viet Dang, Chair
Assistant Professor Yanning Shen
Professor Terry Sanger

2024

© 2024 Meng Ma

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to Professor Quoc-Viet Dang, my master thesis advisor. His support and encouragement have played an indelible role throughout my master's journey and the completion of my thesis. Without his guidance and perseverance, this thesis would not have come into existence.

I would also like to thanks the two professors in my thesis committee. I am grateful for their help and guidance in my courses, which has broadened my knowledge and was invaluable in completing my thesis. I deeply appreciate their taking time out of their busy schedules to guide my work.

Additionally, I must thank Professor Sun Lei in Nankai University, who guided me into the field of robotics and research. His influence has been fundamental in shaping my academic path.

In the end, I would like to thank University of California Irvine for providing me with an excellent platform to complete my master, which is crucial to my academic career.

ABSTRACT OF THE THESIS

Motion Planning for Multi-Agent Systems with Obstacles based on Buffered Voronoi Cell

by

Meng Ma

Master of Science in Electrical and Computer Engineering

University of California, Irvine, 2024

Assistant Professor Quoc-Viet Dang, Chair

This paper presents a decentralized motion planning algorithm for multi-robot systems in the environment with static obstacles. In the proposed algorithm, the environment is divided into several obstacle-free convex spaces to make sure tools like convex programming to be used and robots won't be stuck in local minimum. In each time step, the robot finds the largest obstacle-free convex area containing its current position and builds a buffered Voronoi cell, within which the robot can move freely without collision, based on the position information of the robots in its obstacle-free convex area. A prediction and recovery mechanism is also employed to solve the deadlock caused between robots and obstacles. The simulation results show that the proposed algorithm can solve the problem effectively for robots with single integrator dynamics and which do not have non-holonomic constraints in 2D dimension. The idea of the algorithm can also be extended into 3D dimensions for Unmanned Aerial Vehicles (UAV).

1. Introduction

With the development of automation technology, an increasing number of robots are becoming visible in various aspects of human life. Robot navigation has emerged as a significant area of focus within the field of robotics. In this domain, motion planning for multi-robot systems is one of the basic problems. Motion planning in multi-robot systems necessitates not only the safe and collision-free trajectory for each robot, but also requires the fast and accurate decision-making mechanism based on the possible limited information of the system. Besides, deadlock may also occur when several robots are crowded into a narrow space or need to pass the same area at the same time.

Two primary categories of algorithms are utilized to address motion planning in multi-robot systems: centralized algorithms and decentralized algorithms. Centralized algorithm [1, 2] has a central planner that has the global knowledge of all environments and robots, such as positions, velocities, and objectives of all robots. It uses this information to generate the control signals for all robots to optimize one or multiple performance parameters. Centralized algorithms are often favored in scenarios where heightened performance standards or precise control accuracy are imperative. However, as the number of robots increases and the environment becomes complex, the computational complexity of planning for all robots centrally can be challenging. Contrary to the centralized algorithm, in decentralized algorithms [3, 4, 5], which are also known as distributed algorithms, each robot has its own planner and only makes decisions according to the information it has. Due to its scalability and low computational complexity compared with centralized algorithms, decentralized

algorithms are widely used in robot swarms or unmanned aerial vehicles which have a large number of robots. Nevertheless, many decentralized algorithms fail to account for obstacles or deadlock problems, or they only provide simple solutions.

This paper presents a decentralized motion planning algorithm with deadlock prediction and recovery mechanism for multi-robot systems with static obstacles. The algorithm can generate collision-free paths and control inputs real time for robots and has an extended version of the deadlock handling mechanism proposed by Abdullhak et al. in [6], which has been proven to drastically reduce the occurrence of deadlock. To solve the collision avoidance, the algorithm uses the definition of buffered Voronoi cell (BVC) proposed by Zhou et al. in [7], and extends it into environments with static obstacles. In each time step, the robot needs to: (1) calculate a convex area, which is the largest rectangular area within the environment in this paper, containing its positions; (2) calculate the buffered Voronoi cell in the convex area based on the position information of other robots; (3) calculate the shortest path from its current position to the goal, choose the subgoal in the convex area and the waypoint in the BVC; (4) check if the waypoint may cause a deadlock; if yes, choose another waypoint based on deadlock recovery mechanism.

Decentralized motion planning algorithms in multi-robot systems have been widely studied. In [8], Fiorini et al. introduced a definition named “Velocity Obstacle”, which gives a first-order approximation of the robot’s velocities that would cause a collision in a period of time in future. Collision can be avoided by selecting velocity that is not inside Velocity Obstacle. The concept was then extended into “Reciprocal Velocity Obstacle (RVO)” by Jur van den Berg et al. in [9], which simply assumes that other agents make similar decisions to

avoid the obstacles. Optimal Reciprocal Collision Avoidance (ORCA) is another velocity-based algorithm proposed by Jur van den Berg et al. In ORCA, each robot is inferred a half-plane in velocity space that is allowed to be selected by other robots to guarantee collision avoidance. It was then expanded to robots with non-holonomic constraints by J. Alonso-Mora in [11] and Javier Alonso-Mora et al. in [12]. Besides, there are also Acceleration-velocity obstacles (AVO) which takes into account acceleration constraints [13]; Hybrid Reciprocal Velocity Obstacle (HRVO) which counters the oscillation issue in RVO [14]; and Generalized Velocity Obstacles (GVO) which extends the velocity obstacles to handle nonholonomic kinematic constraints [15]. However, all of these VO-variant algorithms need both position and velocity information.

In [7], Zhou et al. presents the definition of buffered Voronoi cells and introduces the BVC algorithm that only needs the position information for robots. They proved that the BVC algorithm has the same computational complexity and similar empirical performance as the ORCA algorithm. Sensor and localization uncertainty was then added into BVC by Wang et al. in [16] and Zhu et al. in [17], taking the sensor and odometry error into account. The probabilistic buffered Voronoi cell was then used in environments with obstacles in [18], but they still use the simple heuristic right-hand rule to deal with deadlock problems. In [6], a novel deadlock prediction and recovery method was introduced by Abdullhak et al., but do not consider obstacles. The algorithm presented in this paper not only accounts for environmental obstacles but also incorporates the mechanism for deadlock recovery.

The rest of the paper is organized as follows. Section 2 is the problem formulation and some preliminary definitions about BVC. In section 3, a detailed explanation of the algorithm

is presented. Section 4 provides the simulation and the result analysis of the algorithm. And section 5 is the discussion.

2. Problem formulation and Preliminaries

In this section, I will provide a formulation of the motion planning problem for multi-robot systems in an environment with static obstacles and some preliminary definitions.

Consider a system with n robots in a 2D plane R^2 denoted by $R_1, R_2, \dots, R_n$, the start position for the robots is $S_1, S_2, \dots, S_n$, the goal position is $G_1, G_2, \dots, G_n$, and the obstacles in the environment is $O_1, O_2, \dots, O_m$ for m obstacles. For robot R_i, I denote the current center position at time t as $p_{i,t}$. Each robot has the same size with a safety radius r_s.

The motion planning problem for multiple robots can be formulated as: in the system above, finding the input controls u_i for R_i so that R_i can move form S_i to G_i without any collision with other robots and any obstacle.

A. Assumption

In this thesis, we assume: 1) All the robots have a same size with a safety radius r_s; 2) The robot knows the positions of all robots and the obstacles without communication loss or error; 3) Robots are able to execute the control inputs calculated by the algorithm without delay or error; 4） All obstacles are rectangular, and can be determined by four vertices.

B. Kinematic Constraints

We simply assume that the robot used in this thesis is an omnidirectional mobile robot, which does not have non-holonomic constraints. For omnidirectional mobile robots, there are three degrees of freedom: X-axis, Y-axis and a rotation axis Z-axis. But we only need to consider X-axis and Y axis because of the absence of non-holonomic constraints.

The kinematic constraints can be represented as follows.

For robot R_i at time t, there are:

$$p_{i,t+1} = Ap_{i,t} + Bu_{i,t},$$
$$\|u_{i,t}\| \leq u_{max},$$

where A, B are kinematic matrices of the robot and u_{max} is the maximum line speed.

In this thesis, the state vector p is $[x, y]^T$, which is the position of the robot, the control vector is $[v_x, v_y]^T$, which is the velocity of the robot. The State Transition Matrix A can be a 2×2 identity matrix:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The control input matrix B is:

$$\begin{bmatrix} \Delta_t & 0 \\ 0 & \Delta_t \end{bmatrix}$$

where Δ_t is the time step of a control loop.

C. Collision-free Constraints

For each robot R_i, I formulate the collision free constraints as follows:

$$\left\|p_{i,t} - p_{j,t}\right\| > 2r_s, \forall i, j \in \{1, 2, \dots, n\}, i \neq j$$

$$\left\|p_{i,t} - O_j^*\right\| > r_s, \forall j \in \{1,2, \dots, m\}$$

where O_j^* is the nearest point of obstacle O_j to $p_{i,t}$, and $\|\cdot\|$ denotes the Euclidean distance. The first constraint is the collision-free constraint between robots and the second constraint is the constraint between robots and obstacles.

D. Voronoi Diagram

Voronoi Diagram is a method of segmenting a plane into parts based on several points. During segmenting, all the points in the plane that is closest to one given point were segmented in an area, which is called the Voronoi Cell of this point. Suppose we have a set of points: $x_1, x_2, \dots, x_n$, the Voronoi cell of the point x_i is defined as:

$$V_i = \{x \in R^2 | \ \|x - x_i\| \leq \left\|x - x_j\right\|, \forall j \neq i\},$$

where $\|\cdot\|$ denotes the Euclidean distance. The Voronoi diagram is defined as $V = V_1 \cup V_2 \cup \dots \cup V_n$. The point set X is defined as the *Seed* for this Voronoi diagram. In this proposed motion planning algorithm for a multi-robot system, we use robots' positions as seeds to build a Voronoi diagram.

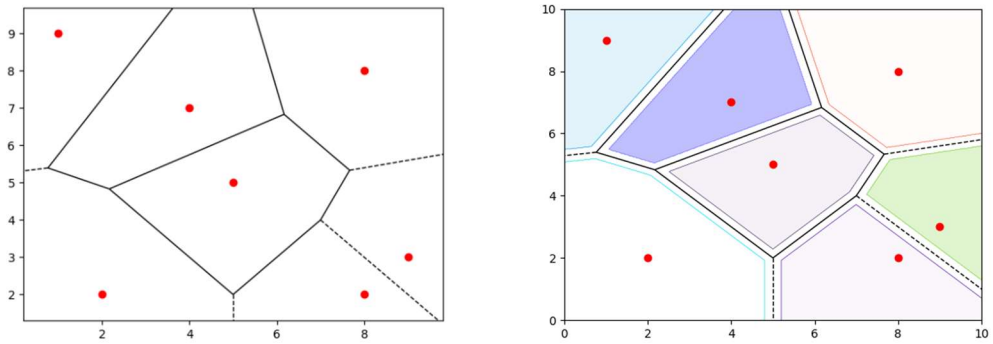

Figure 2. 1: Voronoi diagram and buffered Voronoi cell for 7 robots

E. Buffered Voronoi Cell

To make sure each robot can move without collision in its own Voronoi cell, the actual moving area for robot's center is smaller than the Voronoi cell. Suppose we have n robots $R_1, R_2, \dots, R_n$, with Voronoi cell $V_1, V_2, \dots, V_n$. For robot R_i, the buffered Voronoi cell is defined as the area formed by moving each edge of Voronoi cell V_i to robot R_i's position p_i by a distance of safety radius r_s. Thus, the collision-free constraint between robots must be satisfied if each robot's position is within its buffered Voronoi cell.

F. Decentralized Motion planning based on Buffered Voronoi Cell

The buffered Voronoi cell guarantees that when robots move in their buffered Voronoi cell, there will be no collision between robots. At each time step, the robot just repeats the following steps: 1) generate the buffered Voronoi cell; 2) choose a waypoint in its buffered Voronoi cell; 3) calculate the control input for the robot to move to the waypoint; until each robot reaches their goal. In the previous work [7], Zhou et al. have demonstrated the feasibility of using BVC for multi-robot path planning in an environment without obstacles.

In their work, they use Receding Horizon Path Planning and regard the motion control of robots as a quadratic problem. At each time step, the robot calculates a QP problem and follows the control to move within its BVC.

Like other decentralized motion planning algorithms for multi-robot systems, deadlock is a problem that cannot be ignored. Deadlock happens when robots block the way of other robots due to the fact that they do not know the control decision made by others and at least one robot cannot reach its goal. In [7], Zhou et al. solve deadlock by a simple heuristic method, the right-hand side rule: when robots encounter deadlock, they all choose their right-hand side to move. But this kind of method becomes inefficient when the number of robots is large. In [6], Abdullhak et al. propose a new deadlock prediction and recovery mechanism, with which the success rate for robots to reach their goals is ideal even with a large number of robots.

However, both of these algorithms do not consider obstacles and cannot be used in an environment with obstacles. There are several reasons: (1) Adding obstacles in the environment introduces non-convex polygon in Voronoi diagram, the QP receding horizon path planning cannot apply because the problem is not a convex problem; (2) The non-convex optimization is both computationally expensive and unpracticable, robots may get stuck in local minimum; (3) Neither the right-hand rule by [7] and the deadlock recovery by [6] cannot be directly used in environment with obstacles because they do not consider deadlock caused between robots and obstacles, which sometimes may cause collision.

3. Method

In this section, I will first propose the overview of the algorithm, then divide the algorithm into several stages to explain it in detail.

A. Overview of Algorithm

The basic idea for the proposed algorithm is dividing the non-convex environment filled with obstacles into multiple convex areas for robots to move in their BVC, which guarantees collision-free constraints during the entire movement. The deadlock prediction and recovery mechanism in [6] by Abdullhak et al. is extended and used to reduce the loss on algorithm's performance caused by deadlock.

The proposed algorithm can be divided into several stages: (1) Find the convex area, calculate the buffered Voronoi cell for the robot in this convex area; (2) Use shortest path algorithm to find the nearest point in this convex area, set it as the subgoal g_s; (3) Choose the nearest point in the BVC as the waypoint g_t to subgoal; (4) If the robot encounters deadlock, go to deadlock recovery; (5) If the robot reaches the subgoal, choose a new subgoal.

The overview is given in Algorithm 1.

Algorithm 1 Collision avoidance algorithm with obstacles

Input: Robot's position p_t, goal position g, other robot's postion p_{it}, obstacles' position O

Initialize
while (!Robot reaches the goal g) **do**
    Generate buffered Voronoi cell V_t at time t in C_t
    Calculate the closest point g_t^* in BVC to g_s
    if Robot reach subgoal g_s **then**
        if $AreaChange = False$ **then**
            Set $AreaChange = True$
            Use shortest path algorithm find new subgoal g_s
            Find the new largest rectangular area C_t
        else
            Set $AreaChange = False$
        end if
    else if g_s in V_t **then**
        set g_t as g_s
    else if $DeadloclRecover = True$ **then**
        Perform Deadlock Recovery
        Set $g_t = g_r$
    else if $DeadlockDetected = True$ at g_t^* **then**
        Initialize Deadlock Recovery
        Choose RecoverPoint g_r
        Set $g_t = g_r$
    else
        Set g_t as g_t^*
    end if
    Calculate control signal u_t towards g_t
    Move robot, update position
end while
Return

B. Get the Convex Area

To ensure that robots can move without collision within its BVC and the problem does not deviate from convex optimization, the buffered Voronoi diagram must be generated in an environment having no obstacles. In the proposed algorithm, I choose the largest horizontal rectangular area in the map that contains the coordinates of the robot as the convex area because all obstacles can be represented by rectangles. However, it is still a good choice to

choose the largest convex area that contains the point. In [19], Deits et al. presented IRIS (Iterative Regional Inflation by Semidefinite programming), which is an algorithm that can quickly compute the large polygon of collision-free space in a computation time which is linear in the number of obstacles.

There are two reasons why I choose the rectangle as the convex area. (1) The map only has rectangle obstacles; it is more straightforward to compute the rectangle than a convex polygon. (2) The robot needs to compute the convex area at each time step to track the new subgoal. A map with rectangular obstacles has only a limited number of obstacle-free rectangles. The algorithm only needs to find all the rectangles in the beginning of the algorithm, so that the robot can traverse these rectangles once, instead of running the whole algorithm at each time step.

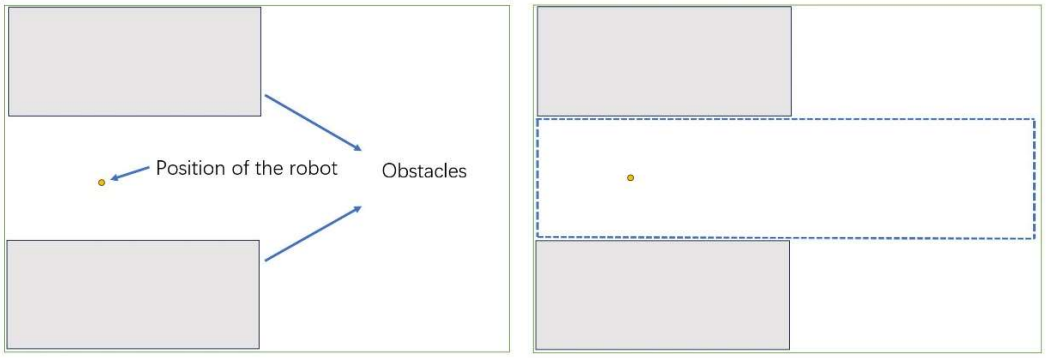

Figure 3. 1: The largest rectangle used in the proposed algorithm

C. The Shortest Path Algorithm

When there are no obstacles, the robot can directly select the point closest to the goal in BVC as the waypoint and directly calculate the control input. However, this strategy cannot be used in an environment with obstacles since robots may be stuck in the local minimum. To

make sure the robot is moving on the right direction, the shortest path algorithm is used to choose the subgoal g_s.

There are many shortest path algorithms, such as Dijkstra's algorithm [21] and A* algorithm [20]. In this algorithm, A * is used to generate the shortest path.

At each time step, the robot finds a shortest path from its current position p_t to its final goal g, then choose the last point in the rectangle C as the subgoal g_s in this time step. The current position p_t and the subgoal g_s is in one collision-free rectangle C, so that robot can move from p_t to g_s using normal nearest waypoint g_t^* in its BVC.

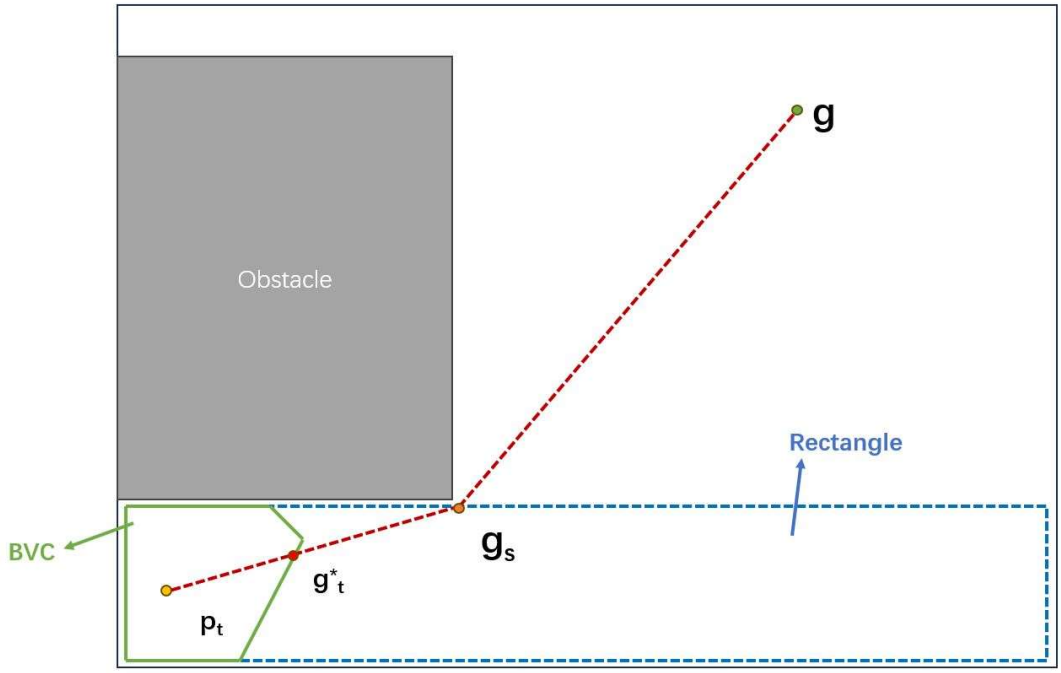

Figure 3. 2: The choice of subgoal g_s and waypoint g_t^*

D. The Area Change Procedure

When the robot reaches the subgoal g_s, it goes into another procedure of the algorithm: *Area Change Procedure*. In order for the robot to leave the current rectangle and enter the

next area, a new subgoal needs to be selected. In the proposed algorithm, the new subgoal g_s' is set to the first point on the shortest path that is not within the current collision-free rectangle C. And to make sure the collision-free constraints of the robot when moving from the current position p_t to the new subgoalg_s', robot will choose another collision-free rectangle C' that contains both p_t and g_s' and build a new buffered Voronoi cell.

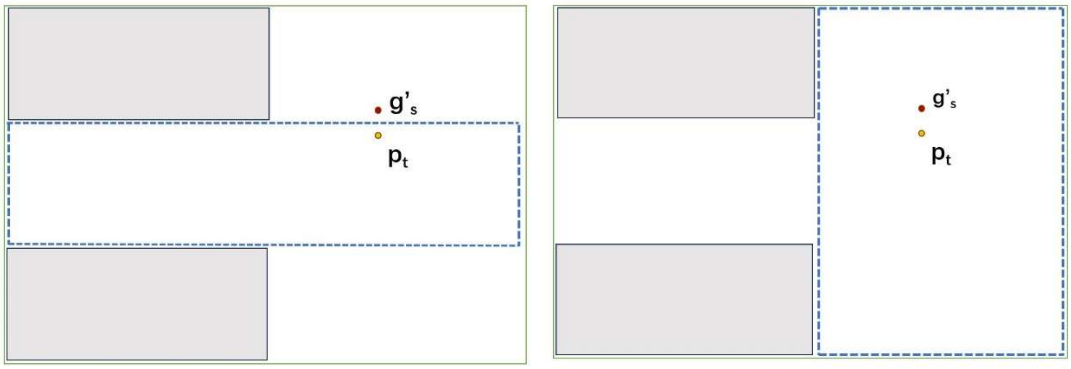

Figure 3. 3: The process of change the rectangle

Then the robot will move to the new subgoal g_s'. When reaching g_s', the robot goes back to normal state and continues moving in the new rectangle and BVC.

E. Deadlock Prediction.

Deadlock constitutes a significant challenge in decentralized algorithms, occurring when one or more robots are unable to progress towards their designated goals solely through the use of their independent decentralized planning mechanisms.

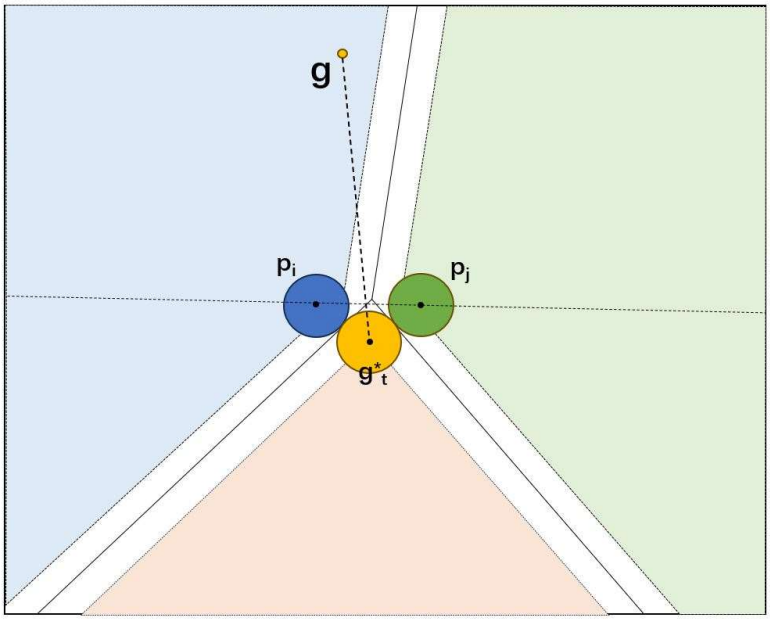

Figure 3. 4: Deadlock without obstacle

When the robots move in the environment without obstacles, the deadlock caused by two robots can be easily solved by right-hand rules: the robots both choose its right-hand side or left-hand side to move, then the deadlock can be broken. Thus, an unsolvable deadlock can only happen when the number of robots is larger than 2.

Suppose at time step t, the robot chooses the closest point g_t^* as the waypoint to the goal g, R_i and R_j are two robots in neighbor robot group N, which satisfies

$$N = \{R_i | \|g_t^* - p_i\| < kr_s, k \geq 2, i = 1, 2, \dots, n\}$$

In [6], when the closest waypoint g_t^* and the goal g are on the different sides of the line between all neighbor robot pairs R_i and R_j in N, whose distance is smaller than $4r_s$, a deadlock is detected.

This prediction strategy works well when there are no obstacles. However, it fails to predict the deadlock caused between other robots and obstacles. When robots are moving in an environment with obstacles, deadlock may occur with only two robots.

To deal with the deadlock caused between robots and obstacles, we not only consider neighbor robot pairs, but also consider the point pair between the neighbor's position p_i and the projection g_{pt}^* of g_t^* on the obstacles.

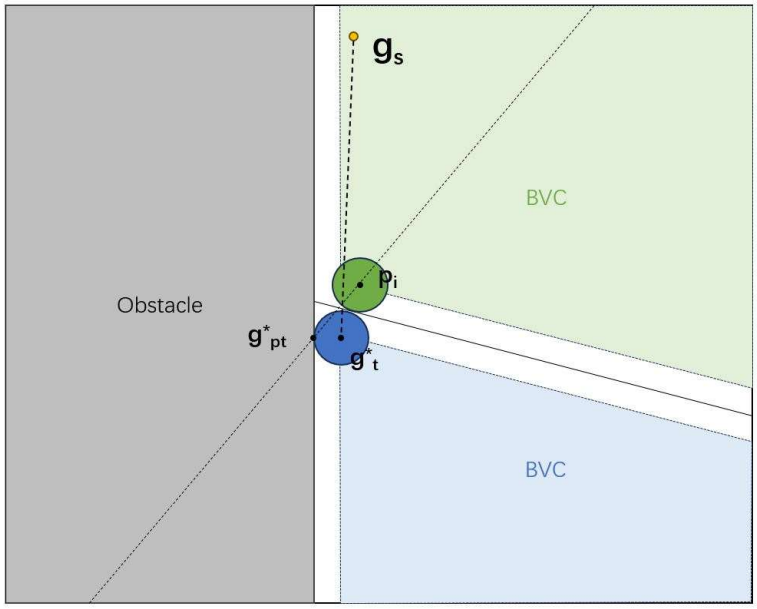

Figure 3. 5: Deadlock with obstacle

Suppose the deadlock happens when robot R is at g_t^*, the neighbor robot R_i who shares the edge between R and R_i is at p_i. The projection of waypoint g_t^* on the obstacle O is g_{pt}^*, which is also the nearest point of obstacle O to g_t^*.

Since the deadlock must happens when the robot is at the vertex of BVC (by [6]), there must be:

$$\|g_t^* - p_i\| = 2r_s$$

$$\left\|g_t^* - g_{pt}^*\right\| = r_s$$

Proposition 1: For above condition to be satisfied, the distance between p_i and g_{pt}^* must be equal or less than $3r_s$:

$$\left\|p_i - g_{pt}^*\right\| \leq 3r_s$$

Proof: Assume $\left\|p_i - g_{pt}^*\right\| > 3r_s$, for any point x that is not on the line segment between p_i and g_{pt}^*, there are:

$$\|x - p_i\| + \left\|x - g_{pt}^*\right\| > \left\|p_i - g_{pt}^*\right\| > 3r_s$$

If we choose x as a point where $\|x - p_i\| = 2r_s$, then there must be

$$\left\|x - g_{pt}^*\right\| > r_s$$

Which contradicts the above condition.

Thus, we can extend the deadlock prediction condition.

Suppose we have a neighbor robot group N:

$$N = \{R_i | \|g_t^* - p_i\| < kr_s, k \geq 2, i = 1, 2, \ldots, n\}$$

and an obstacle O within r_s of g_t^*. The projection of waypoint g_t^* on the obstacle O is g_{pt}^*. A deadlock between robots and obstacles is detected when: the closest waypoint g_t^* and the goal g are on the different sides of the line between robot's position p_i and the waypoint g_t^*'s projection g_{pt}^* on the obstacle O, for all robots whose distance from p_i to g_{pt}^* is smaller than $3r_s$.

At each time step before moving, the robot detects the deadlock between robots and the deadlock between robot and obstacle. As long as one deadlock is detected, the robot enters the deadlock recovery.

F. Deadlock Recovery and recovery success prediction

When a deadlock is detected at waypoint g_t^*, the robot goes to Deadlock Recovery procedure, abandon waypoint g_t^*, and choose another recovery point in its BVC to bypass the deadlock location g_t^*.

In [6], Abdullhak et al. choose the point that is furthest from the line between the robot's current position p_t and the robot's goal g in robot's BVC at time step t, which is called outermost point o_t. In the proposed algorithm, we still use the same way to choose the recovery point, however, with some little changes.

First, the line between p_t and g needs to be changed into the line between p_t and the subgoal g_s. Second, if the deadlock occurs between the obstacles and the robots, the outermost point o_t should not be on the same side with the obstacle.

As the robot moves toward the recovery point, when all the robot in the neighbor robot group N and the projection g_{pt}^* of waypoint g_t^* are on one side of the line between the robot current position p_t and the subgoal g_s, the deadlock recovery is considered successful, the robot goes back to normal state and choose new a waypoint to move to the subgoal.

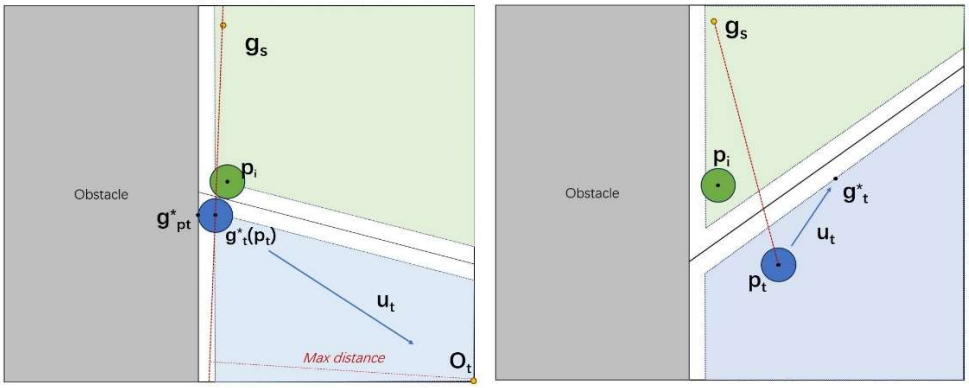

Figure 3. 6: The process of deadlock recovery. The left one shows the deadlock recovery, the right one is the recovery success.

In the deadlock recover procedure, the robot keeps moving to the recovery point o_t until one of these conditions satisfies: (1) The recovery success give the true result; (2) The recovery point o_t outside the new BVC, the algorithm finds a new recovery point then continues recovering; (3) The robot reaches the recovery point, checks the recovery success result: if the result is true, goes back to normal state, else finds a new recovery point then continues recovering.

Although this kind of deadlock prediction and recovery mechanism cannot eliminate deadlock, it significantly reduces the occurrence of deadlock.

4. Simulation

In this section, I will provide the simulation result using the proposed buffered Voronoi cell collision avoidance algorithm with obstacles. The analysis of the computational complexity is also provided.

The simulation is executed on a laptop running Windows 10 with Intel Core i7-11800H @2.30GHz and 16G RAM. In the simulation, the robots are running in a 10*10 map with 10

square obstacles with side length 10. The safety radius r_s is set as 0.1. The maximum linear velocity of the robots is also 0.1.

The simulation program is based on Python 3.8. To build the Voronoi cell, I used the Voronoi Library in scipy.

I first demonstrate the feasibility of the deadlock prediction and recovery between robots and obstacles.

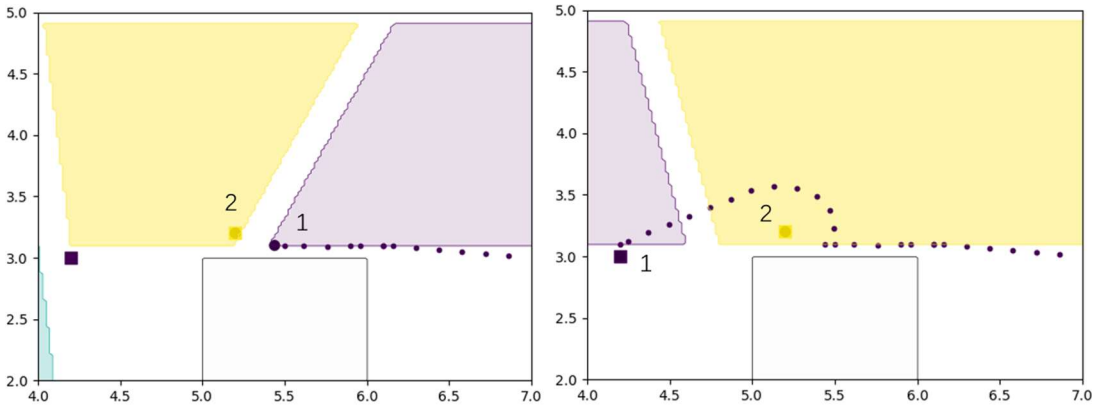

Figure 4. 1: The deadlock detection and recovery process

As shown in Figure 4.1, the two robots are represented as points in purple and yellow. The two trapezoids are the buffered Voronoi cells of them. The trajectory of the robot is represented as small dots with the same color as robots, and the goal of the robot is represented as a square with same color. In this simulation, Robot 1 comes from the right side of the map and goes to the goal at the left side. Robot 2 is a robot that already reached the goal and stay there. The distance between robot 2 and the obstacle is smaller than 3 r_s, which is not enough for a robot to go through. The left figure in Figure 4.1 shows the moment when robot 1 detected the deadlock. It then goes to the deadlock recovery process. The right figure

shows the trajectory of the deadlock recovery. When robot 1 avoids robot 2 and meets the condition of recovery success, it goes to normal state and goes directly to the goal.

Next, I test the feasibility of the algorithm running in the environment with 10 obstacles.

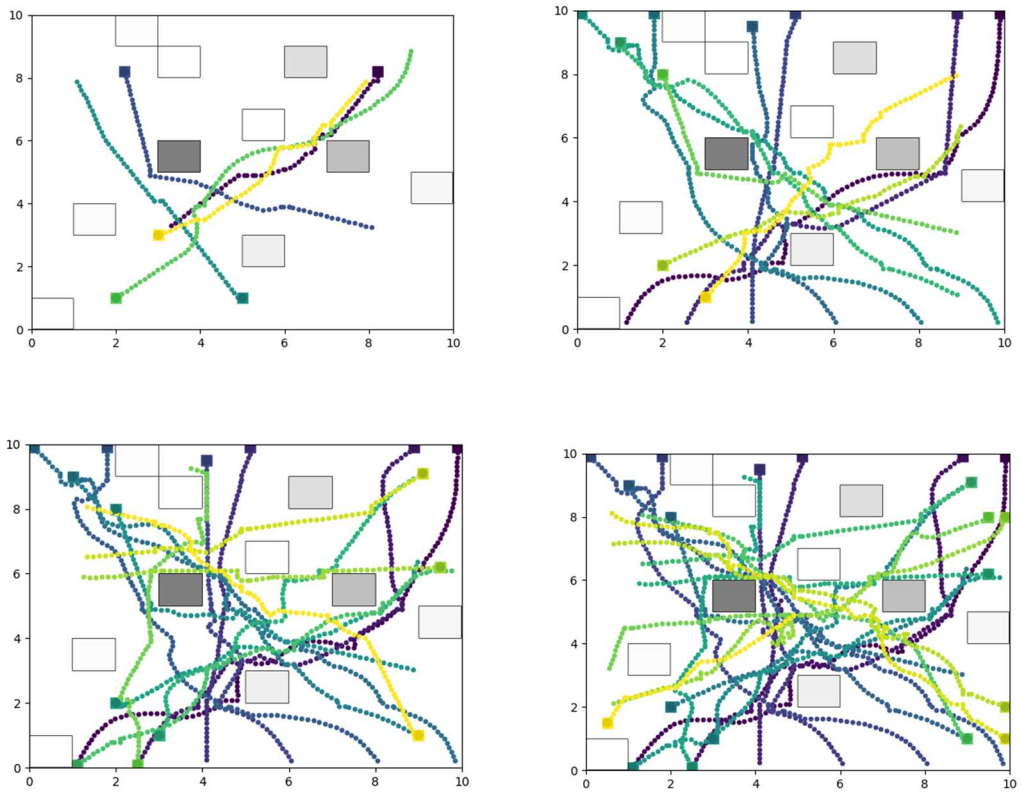

Figure 4. 2: The trajectory of the robots in simulation. The number of the robots in these four cases are 5, 10, 15 and 20.

Figure 4.2 shows the simulation result of 5, 10, 15 and 20 robots. In each case, in order to test the algorithm's ability to handle multi-robot systems, the starting and goal points of robots are set on different sides of the map, or at the opposite corners, so that all the robots need to cross the central area of the map and will encounter each other. According to the simulation, the proposed algorithm can handle the case with at least 20 robots and solve the problem in 125 iterations of the algorithm. The actual time for most robots meeting at the

center area, then leaving each other and going to their own destination is only 50 iterations. In these 50 iterations, more than 15 robots are moving without collision in the 2.5*2.5 center area of the map, and then generating their control signal only by the position information of other robots and solve the problem.

The computational complexity of this algorithm highly depends on the complexity of the map. In each time step, the robot needs to generate the buffered Voronoi cell, calculate the shortest path by A*, finding the nearest point in BVC. Constructing a Voronoi diagram typically takes $O(nlogn)$ times by using divide-and-conquer algorithm, where n is the number of the robots. To find the nearest point in a convex area to an outside point, the time complexity is $O(k)$, where k is the number of the vertices of this convex area. The worst case for A* algorithm is without useful information from the heuristic and becomes a Dijkstra's algorithm. The time complexity of Dijkstra's algorithm is $O((V+E)logV)$, where V is the number of vertices, E is the number of edges. For a grid map in A* algorithm, the number of edges is linear to the number of vertices, so the computational complexity of A* can be represented as $O(VlogV)$, where V is the number of the nodes in the map. Therefore, the total computational complexity of the proposed algorithm is $O(VlogV)$, where V is the number of the nodes in the map.

5. Conclusion and Future Work

The paper introduces a decentralized motion planning algorithm for multi-robot systems with obstacles based on the buffered Voronoi cell. The algorithm guarantees collision avoidance between robots, and can not only handle the deadlock caused between robots, but

also deadlock caused between robots and obstacles. There are still some improvements that can be made.

First, the algorithm can be extended into environments with more complex obstacles by changing the rectangle area used in the algorithm to the largest convex area. Second, although A* is a simple and fast algorithm, the path it calculated has the problem that is not smooth and the robot is too close to obstacles. Third, the robot in the paper is first-order integrator robot without non-holonomic constraints. To implement the algorithm in reality, more constraints need to be considered.

In future work, I will attempt to solve the above problems. First, the IRIS algorithm by [19] is a fast algorithm to get a collision-free area within large numbers of irregular obstacles that only takes a few seconds to get a convex region in an environment containing one million obstacles. By combining this algorithm, the BVC algorithm can be used in more cluttered environments with complex obstacles. Second, there are many other global path planning algorithms that can generate paths that are in the middle among obstacles, such as Vector Field Histogram methods [22]. Another way is, instead of using Voronoi cells, using the edges of the Voronoi diagram as the path, which may have some similar points with the BVC algorithm used in this paper. Third, the kinematic model of this algorithm needs to add more constraints to generate velocity consistent with more complicated robots. The corresponding control signal can be calculated by optimization method.

References

[1] G. Lionis and K. J. Kyriakopoulos, "Centralized Motion Planning for a Group of Micro Agents Manipulating a Rigid Object," Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005., Limassol, Cyprus, 2005, pp. 662-667, doi: 10.1109/.2005.1467093.

[2] L. Aguilar, R. Alami, S. Fleury, M. Herrb, F. Ingrand and F. Robert, "Ten autonomous mobile robots (and even more) in a route network like environment," Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, Pittsburgh, PA, USA, 1995, pp. 260-267 vol.2, doi: 10.1109/IROS.1995.526170.

[3] Lumelsky, V., Harinarayan, K. Decentralized Motion Planning for Multiple Mobile Robots: The Cocktail Party Model. Autonomous Robots 4, 121–135 (1997).

[4] Dit-Yan Yeung and G. Bekey, "A decentralized approach to the motion planning problem for multiple mobile robots," Proceedings. 1987 IEEE International Conference on Robotics and Automation, Raleigh, NC, USA, 1987, pp. 1779-1784, doi: 10.1109/ROBOT.1987.1087768.

[5] Yi Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), Washington, DC, USA, 2002, pp. 2612-2619 vol.3, doi: 10.1109/ROBOT.2002.1013625.

[6] M. Abdullhak and A. Vardy, "Deadlock Prediction and Recovery for Distributed Collision Avoidance with Buffered Voronoi Cells," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 2021, pp. 429-436, doi: 10.1109/IROS51168.2021.9636609.

[7] Zhou, Dingjiang, et al. "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells." IEEE Robotics and Automation Letters 2.2 (2017): 1047-1054.

[8] Fiorini P, Shiller Z. Motion Planning in Dynamic Environments Using Velocity Obstacles. The International Journal of Robotics Research. 1998;17(7):760-772. doi:10.1177/027836499801700706

[9] J. van den Berg, Ming Lin and D. Manocha, "Reciprocal Velocity Obstacles for real-time multi-agent navigation," 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, 2008, pp. 1928-1935, doi: 10.1109/ROBOT.2008.4543489.

[10] J. Alonso-Mora, A. Breitenmoser, P. Beardsley and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 2012, pp. 360-366, doi: 10.1109/ICRA.2012.6225166.

[11] van den Berg, J., Guy, S.J., Lin, M., Manocha, D. (2011). Reciprocal n-Body Collision Avoidance. In: Pradalier, C., Siegwart, R., Hirzinger, G. (eds) Robotics Research. Springer Tracts in Advanced Robotics, vol 70. Springer, Berlin, Heidelberg.

[12] Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., & Siegwart, R. (2013). Optimal reciprocal collision avoidance for multiple non-holonomic robots. In Distributed autonomous robotic systems: The 10th international symposium (pp. 203-216). Springer Berlin Heidelberg.

[13] J. van den Berg, J. Snape, S. J. Guy and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 3475-3482, doi: 10.1109/ICRA.2011.5980408.

[14] J. Snape, J. v. d. Berg, S. J. Guy and D. Manocha, "The Hybrid Reciprocal Velocity Obstacle," in IEEE Transactions on Robotics, vol. 27, no. 4, pp. 696-706, Aug. 2011, doi: 10.1109/TRO.2011.2120810.

[15] D. Wilkie, J. van den Berg and D. Manocha, "Generalized velocity obstacles," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 2009, pp. 5573-5578, doi: 10.1109/IROS.2009.5354175.

[16] M. Wang and M. Schwager, "Distributed Collision Avoidance of Multiple Robots with Probabilistic Buffered Voronoi Cells," 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), New Brunswick, NJ, USA, 2019, pp. 169-175, doi: 10.1109/MRS.2019.8901101.

[17] H. Zhu and J. Alonso-Mora, "B-UAVC: Buffered Uncertainty-Aware Voronoi Cells for Probabilistic Multi-Robot Collision Avoidance," 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), New Brunswick, NJ, USA, 2019, pp. 162-168, doi: 10.1109/MRS.2019.8901092.

[18] Zhu, Hai, Bruno Brito, and Javier Alonso-Mora. "Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware Voronoi cells." Autonomous Robots 46.2 (2022): 401-420.

[19] Deits, Robin, and Russ Tedrake. "Computing large convex regions of obstacle-free space through semidefinite programming." Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics. Cham: Springer International Publishing, 2015.

[20] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4.2 (1968): 100-107.

[21] Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022. 287-290.

[22] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," in IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 278-288, June 1991,