# Multi-Agents Formation Control

Gabriel Paffi

**Robotic Project II**

Under the supervision of:
Kai Ren and Prof. Maryam Kamgarpour

31/01/2025

sycamore lab

SYSTEMS CONTROL AND MULTIAGENT OPTIMIZATION RESEARCH

École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland

# Contents

## List of Figures

# Abstract

Multi-agent formation Control is a critical challenge in modern robotics, particularly in dynamic and cluttered environments. This project explores trajectory optimization methodologies to ensure efficient navigation and collision-free motion for multiple robots. Initially, a literature review was conducted to understand existing models, focusing on the CCILQ Games (Chance Constraints Iterative Linear Quadratic games) [3] model with MPC (Model Predictive Control). However, this approach proved computationally intensive, with prolonged convergence times and significant challenges in real-time obstacle avoidance. To address these limitations, the project transitioned to a Mixed-Integer Linear Programming (MILP) [4] approach incorporating binary decision variables (z) to handle obstacle constraints effectively. This method was implemented and tested with multiple robots navigating environments with varying obstacle densities. Finally, simulations were conducted to validate the performance of the MILP-based trajectory planning model, demonstrating improved convergence speed, collision avoidance reliability and scalability with multiple robots. The results compare the advantages of MILP and CCILQ in real-time multi-robot planning and offer insights for future research and implementation in dynamic robotic systems.

# 1 Introduction

## 1.1 Motivation

In real-world applications such as warehouse automation, environmental monitoring, and autonomous vehicle fleets, multi-robot systems are increasingly being deployed to perform complex tasks collaboratively. These tasks require precise coordination, efficient trajectory planning, and robust collision avoidance, especially in environments with dynamic obstacles and limited space. Achieving these goals demands advanced mathematical models and control strategies capable of managing both the individual behavior of robots and their collective interactions.

Chance-Constrained Iterative Linear-Quadratic (CCILQ) Games have gained attention for their ability to model uncertainties in robot dynamics and interactions while providing a structured approach to trajectory planning. This method is particularly well-suited for scenarios where probabilistic safety measures are essential, such as autonomous vehicle intersections or drone swarms operating in unpredictable environments.

On the other hand, Mixed-Integer Linear Programming (MILP) provides a powerful optimization framework for multi-robot systems by encoding constraints, including collision avoidance and formation rules, into a set of linear equations. MILP's ability to handle binary decision variables makes it particularly effective for defining clear, non-overlapping robot paths in cluttered spaces while maintaining computational tractability.

The motivation for studying these two approaches stems from their potential to address key challenges in real-world multi-robot applications. By exploring and implementing these methods, this project aims to contribute to the development of robust and scalable trajectory planning strategies that can be deployed in practical, real-world scenarios where safety, efficiency, and adaptability are paramount.

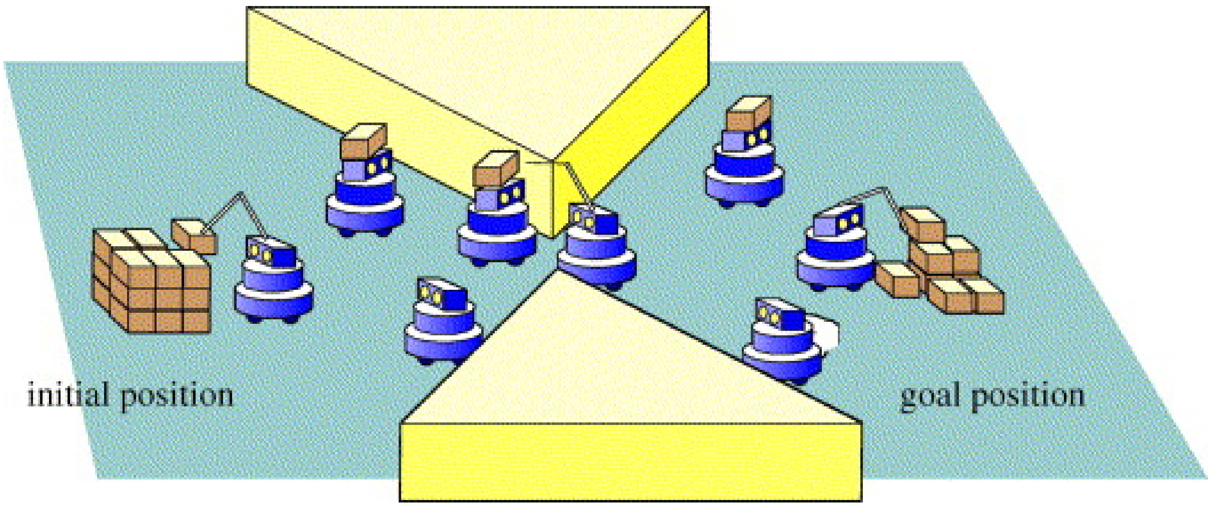


Figure 1: Example of Wheeled Robots in a Warehouse Scenario [1]

## 1.2 Problem Statement

In the context of multi-robot trajectory planning with collision avoidance, ensuring safe navigation while minimizing trajectory costs and avoiding obstacles is a critical challenge. The problem can be mathematically formalized using state transition dynamics, cost functions, and constraints for both static obstacle avoidance and inter-robot collision avoidance. Additionally, in scenarios requiring robots to maintain a specific formation—such as a line, or V-shape, the problem becomes more complex, as constraints must also account for preserving the desired geometric configuration throughout the movement while adapting to dynamic environments.

1. **State Transition Dynamics**
   The state of each robot evolves according to a discrete-time state-space model:

   $$x_{k+1}^{(i)} = f(x_k^{(i)}, u_k^{(i)}, w_k^{(i)}) \tag{1}$$

   - $x_k^{(i)}$: State of robot i at time step k
   - $u_k^{(i)}$: Control input of robot i at time step k
   - $w_k^{(i)}$: Process noise or uncertainty affecting robot i at time step k

2. **Cost Function**
   The objective is to minimize a cost function that captures various aspects of the robots' navigation and formation control. A general form of the cost function is:

   $$J = \sum_{i=0}^{L} \sum_{k=0}^{N-1} [w_{goal} \cdot ||x_{k+1}^{(i)} - x_{goal}^{(i)}||^2 + w_{formation} \cdot \sum_{j \neq i} ||x_{k+1}^{(i)} - x_{k+1}^{(j)} - r_{ij}||^2 + w_{control} \cdot ||u_k^{(i)}||^2] \tag{2}$$

   - J: Total cost function to be minimized
   - $x_{goal}^{(i)}$ : Desired final state (goal position) of robot i
   - $r_{ij}$: Desired relative position between robot i and j for maintaining formation
   - $u_k^{(i)}$ Control input of robot i at time step k
   - L: Total number of robots
   - N: Total number of time steps

3. **Static Obstacles Avoidance Constraints**
   To ensure static obstacle avoidance for each robot at any time, these constraints are imposed:
   $$Lx_k^{(i)} \leq b \quad \forall k \in N, \forall i \tag{3}$$
   - L,b: Matrices and vectors defining the obstacle boundaries

4. **Collision Avoidance Between Robots**
   To prevent inter-robot collisions, the following constraint is enforced:
   $$||x_k^{(i)} - x_k^{(j)}|| \geq d_{min} \quad \forall i \neq j \tag{4}$$
   - $d_{min}$: Minimum safe distance between two robots
   - $i, j$ : Indices of two different robots

## 1.3 Literature Review

Coordinating multiple agents to achieve and maintain a specific geometric formation, such as a V-shape, line, or grid, is critical in various applications, including autonomous drones, robotic swarms, and satellite constellations. Formation control ensures that agents not only avoid collisions but also collectively achieve a desired spatial configuration while navigating dynamic environments. To address these challenges, advanced methodologies like Chance-Constrained Iterative Linear-Quadratic (CCILQ) games and Mixed-Integer Linear Programming (MILP) have gained prominence due to their ability to optimize trajectories under constraints and uncertainty.

### 1.3.1 Multi-Agents Formation Control

Multi-agent formation control focuses on coordinating multiple agents to achieve a specific geometric configuration or task. The research in this domain can be categorized based on the sensing capability and interaction topology of agents into three primary control schemes [5]:

- Position-Based Control: Agents sense their positions relative to a global coordinate system and actively control their positions to achieve a desired formation [6], [7].
- Displacement-Based Control: Agents control their neighbors' relative displacements under the assumption of orientation-aligned local coordinate systems [8].
- Distance-Based Control: Agents control inter-agent distances while sensing relative positions with respect to local coordinate systems [9].

Each control scheme balances requirements on sensing capability and interaction topology, presenting trade-offs between sensor accuracy, system scalability, and communication requirements. Additionally, formation control schemes avoid strict centralization by relying on local sensing and distributed algorithms. Applications span from autonomous vehicle convoys to satellite formations [5].

Figure 2: New Year's Drone Light Show in Shanghai, China [2]

### 1.3.2 CCILQ Games

Chance-Constrained Iterative Linear-Quadratic (CCILQ) stochastic games address multi-agent decision-making in uncertain dynamic environments. These games extend traditional Iterative Linear-Quadratic Games (ILQGames) by introducing probabilistic constraints to ensure safety and feasibility under uncertainty. Key contributions include:

- Stochastic Dynamics: Agents incorporate Gaussian uncertainties in their state and observation models.
- Augmented Lagrangian Method: This approach ensures chance constraints are met without requiring extensive manual tuning of penalty weights.
- Interactive Planning: Agents optimize trajectories iteratively while considering the reactions of others, balancing safety and performance.

CCILQ has been successfully applied in autonomous driving scenarios, including merging, intersection navigation, and roundabout traversal. These algorithms improve safety without overly conservative behavior, leveraging Monte Carlo simulations for validation [3].

### 1.3.3 MILP

Mixed-Integer Linear Programming is widely used in multi-agent systems for trajectory optimization, trajectory planning, and decision-making under constraints. Key characteristics include:

- Optimization Framework: MILP transforms complex trajectory and control problems into solvable linear programming forms with integer decision variables.
- Constraint Handling: Agents can incorporate collision avoidance, operational limits, and resource allocation constraints effectively.
- Scalability and Efficiency: Although computationally intensive, modern solvers enable real-time applications in scenarios like warehouse logistics and autonomous fleets.

Recent advancements in MILP focus on improving computational efficiency through decomposition techniques, heuristic approaches, and hybrid frameworks that combine MILP with dynamic potential games [4] [10].

# 2 CCILQ Games (Constrained Chance Iterative Linear Quadratic)

## 2.1 Problem Formulation

The **Constrained Chance Iterative Linear-Quadratic (CCILQ) Games** framework addresses multi-agent trajectory optimization under uncertainty by balancing safety, performance, and computational feasibility. The formulation integrates stochastic dynamics, probabilistic safety constraints, and iterative optimization strategies.

### 2.1.1 System Dynamics

The system dynamics of each agent (robot) are modeled as a **linear stochastic system**:

$$x_{k+1}^{(i)} = A_k x_k^{(i)} + B_k u_k^{(i)} + w_k^{(i)}, \quad w_k^{(i)} \sim \mathcal{N}(0, W) \tag{5}$$

where:

- $x_k^{(i)}$: State vector of agent $i$ at time step $k$.
- $u_k^{(i)}$: Control input vector of agent $i$ at time step $k$.
- $A_k$: State transition matrix at time step $k$.
- $B_k$: Control input matrix at time step $k$.
- $w_k^{(i)}$: Process noise vector following a Gaussian distribution with covariance $W$.

The above equation captures the **stochastic nature** of the agent's dynamics due to process noise and environmental uncertainty.

### 2.1.2 Cost Function

The cost function for each agent is defined as an **finite-horizon quadratic cost function**, representing the trade-off between state deviation and control effort:

$$J^{(i)} = \mathbb{E}\left[\sum_{k=0}^{N-1}\left(x_k^{(i)^T} Q x_k^{(i)} + u_k^{(i)^T} R u_k^{(i)}\right)\right] \tag{6}$$

where:

- $Q$: State cost matrix (positive semi-definite).
- $R$: Control cost matrix (positive definite).
- $\mathbb{E}$: Expectation operator.

Each agent aims to minimize its cost function while satisfying both **state constraints** and **collision avoidance constraints**.

### 2.1.3 Chance Constraints

To ensure probabilistic safety and feasibility under uncertainty, **chance constraints** are imposed on both static obstacle avoidance and inter-agent collision avoidance:

**1. Static Obstacle Avoidance Constraint:**

$$P\left(Lx_k^{(i)} \leq b\right) \geq 1 - \epsilon \tag{7}$$

where:

- $L$, $b$: Matrices defining the obstacle boundaries.
- $\epsilon$: Tolerable probability of violating the constraint.

**2. Inter-Agent Collision Avoidance Constraint:**

$$P\left(\|x_k^{(i)} - x_k^{(j)}\| \geq d_{min}\right) \geq 1 - \delta, \quad \forall i \neq j \tag{8}$$

where:

- $d_{min}$: Minimum allowable distance between any two agents.
- $\delta$: Tolerable probability of collision.

These constraints ensure that the probability of constraint violation remains below a predefined threshold.

## 2.2 CCILQ Games for Shape Formation

The application of CCILQ Games in shape formation focuses on coordinating multiple agents to achieve and maintain a specific geometric configuration. The agents (robots) must navigate dynamically while avoiding collisions and ensuring convergence to their target positions, forming a desired shape (e.g., regular polygon, line, or custom configuration). The overall structure of the algorithm can be seen at Algorithm 1.

**Algorithm 1** Chance-Constrained Iterative Linear-Quadratic Stochastic Game

```
Require: Initial trajectory $\hat{x}$, nominal control $\hat{u}$
Ensure: Feedback control policy $\{\gamma_i\}_{i=1}^N$
 1: while max chance constraint error $>$ tolerance do
 2:     Evaluate chance constraint violations
 3:     Update Lagrangian Multipliers
 4:     while not converge do
 5:         Linearize the system and chance constraints around the nominal trajectory $\hat{x}, \hat{u}$
 6:         Add the Lagrangian terms to the running cost to obtain the quadratic cost
 7:         $\{\gamma_i\} \leftarrow$ Solve the linear-quadratic Gaussian game
 8:         Update $\hat{x}, \hat{u}$
 9:     end while
10: end while
```

The feedback control policy is a linear-state feedback control policy of the form :

$$u_k^{(i)} = \gamma_k^{(i)}(x_k^{(i)}) = -K^{(i)}x_k^{(i)} + g^{(i)} \tag{9}$$

where:

- $K^{(i)}$: Optimal feedback gain matrix.
- $g^{(i)}$: Feedforward term.

The feedback gain matrix $K^{(i)}$ is computed using the discrete-time Riccati equation, which is solved iteratively via dynamic programming.

## 2.3 Preliminary Results

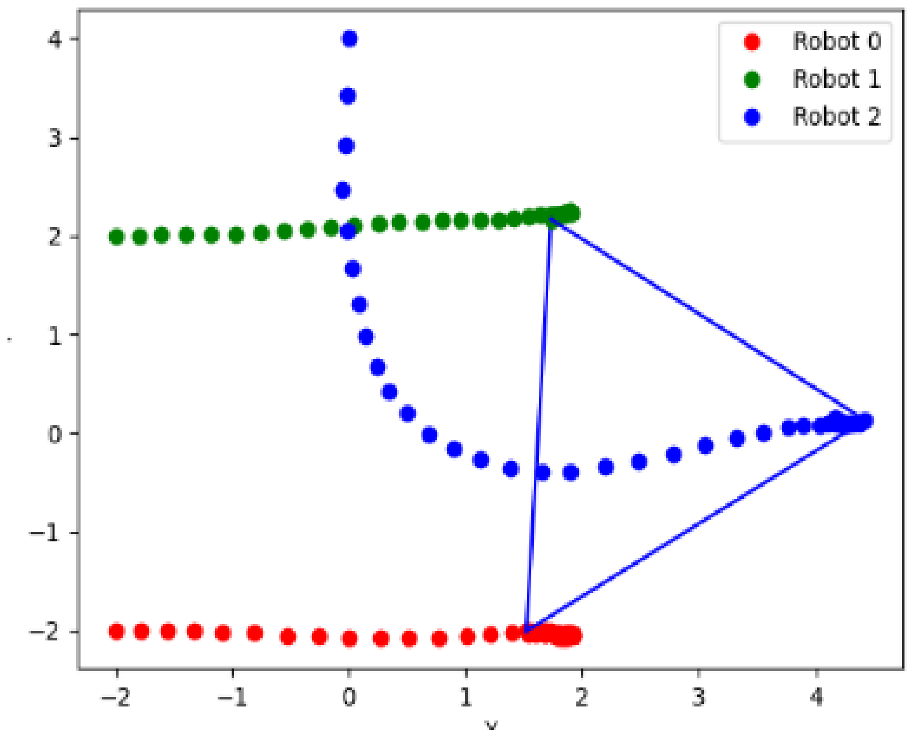


Figure 3: Multi-Agent Trajectory Optimization Using CCILQ Games for Triangular Shape Formation

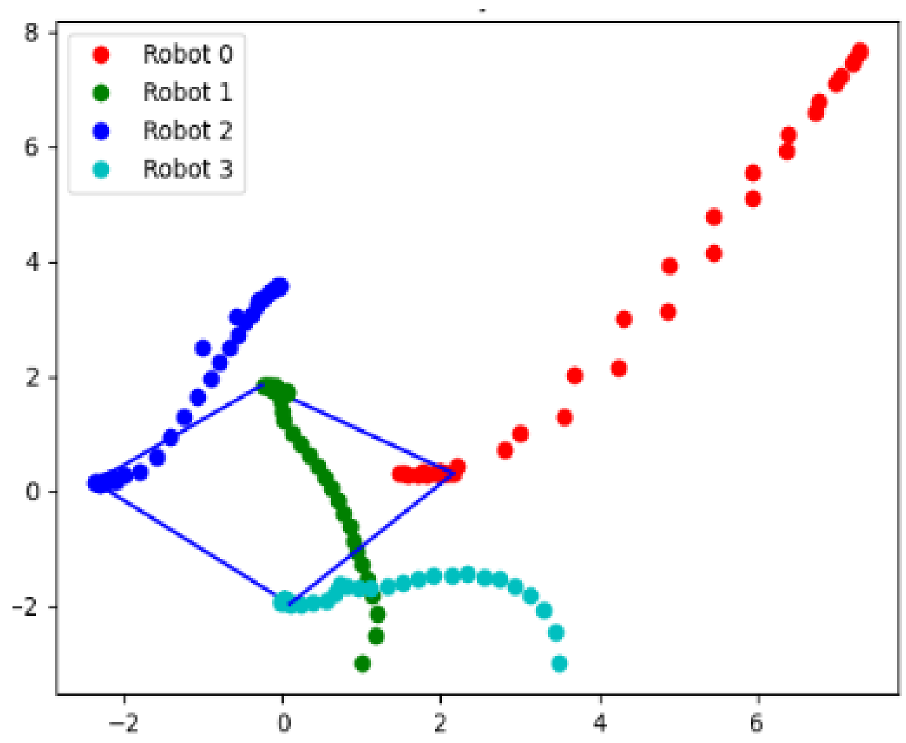


Figure 4: Multi-Agent Trajectory Optimization Using CCILQ Games for Diamond Shape Formation

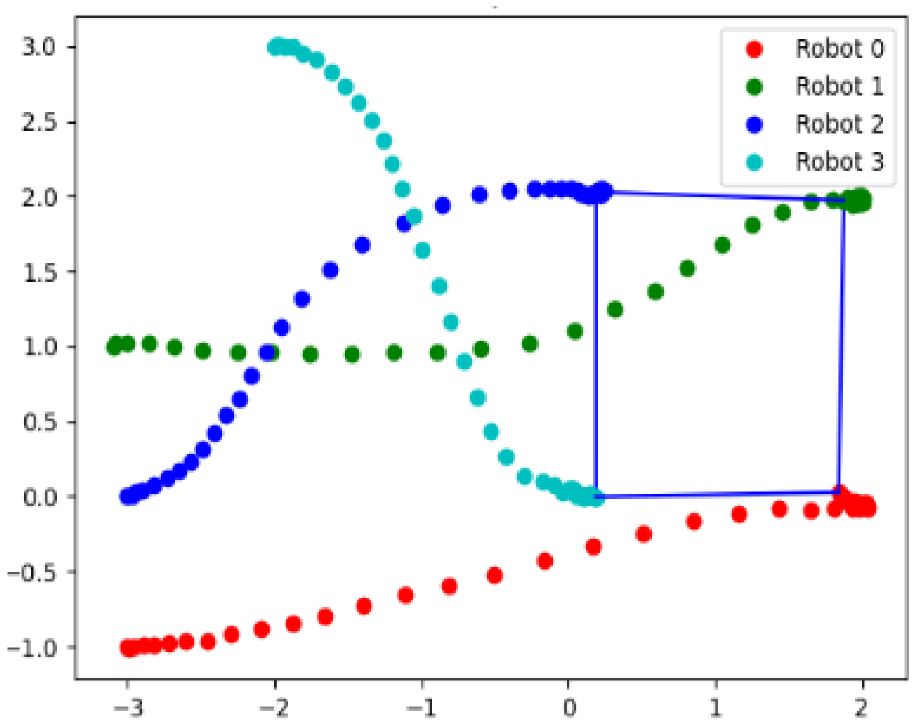


Figure 5: Multi-Agent Trajectory Optimization Using CCILQ Games for Square Shape Formation

## 2.4 Analysis of Time Convergence using CCILQ Games

The Constrained Chance Iterative Linear-Quadratic (CCILQ) Games framework provides a robust approach to multi-agent formation control, addressing both stochastic robot dynamics and probabilistic safety constraints. However, the efficiency of convergence is significantly influenced by the number of robots, their initial positions, and the potential for inter-agent collisions. These simulations were conducted on a system equipped with an Intel Core i7-8565U processor (8 cores, base clock speed 1.80 GHz), 16 GB of RAM, and integrated Intel UHD Graphics 620.

### 2.4.1 Influence of the Number of Robots on Convergence Time

The CCILQ model relies on solving iterative optimization problems for each robot while considering interactions with every other agent in the system. As the number of robots increases, the computational complexity grows exponentially, leading to a significant rise in convergence time:

- Line (2 robots): Average convergence time of 8 seconds.
- Triangle (3 robots): Average convergence time of 45 seconds (Figure3).
- Diamond (4 robots): Average convergence time of 92 seconds (Figure 4).
- Square (4 robots): Average convergence time of 110 seconds (Figure 5).

This increase in convergence time is primarily driven by the growing number of constraints that must be satisfied between agent pairs. More robots mean a higher probability of constraint violations, requiring more iterative adjustments and prolonging convergence.

### 2.4.2 Influence of Configuration and Collision Risks

The spatial configuration of the robots and the risk of collisions play a crucial role in determining convergence efficiency. For example in the diamond formation figure 4, the risk of inter-robot collisions is relatively lower because of the more distributed spatial arrangement of the robots. In the square formation figure 5, there are significantly more potential collision points, because of the initial and goal positions. This results in additional safety constraints that slow down convergence. Thus, even with the same number of robots, the geometric configuration directly affects the complexity of the optimization problem solved by the CCILQ model.

Based on the results obtained from the CCILQ Games model, it is evident that while the approach offers robust multi-agent formation control with collision avoidance, it faces significant computational challenges, particularly as the number of agents increases or when complex spatial constraints, such as obstacle-dense environments, are introduced. These limitations highlight the importance of exploring alternative strategies for multi-agent formation control that can efficiently handle global obstacle avoidance while maintaining scalability and computational tractability. By investigating these alternative approaches, we aim to address the observed convergence delays and improve the overall adaptability of multi-agent systems in dynamic and constrained environments.

# 3 MILP Problem Formulation

## 3.1 General MILP Formulation

The general Mixed-Integer Linear Programming (MILP) problem for multi-agent formation control is formulated as follows:

### 3.1.1 System Dynamics

Each agent follows a discrete-time linear state-space model:

$$x_{k+1}^{(i)} = Ax_k^{(i)} + Bu_k^{(i)} \tag{10}$$

where:

- $x_k^{(i)}$: State vector of agent $i$ at time step $k$.
- $u_k^{(i)}$: Control input vector of agent $i$ at time step $k$.
- $A, B$: State transition and control input matrices.

### 3.1.2 Objective Function

The global cost function to minimize is:

$$J = \sum_{i=1}^{L} \sum_{k=0}^{N-1} \mathbf{C}^T \cdot x^i \tag{11}$$

where:

- $\mathbf{c}$: Weight vector for trajectory tracking and control effort.

### 3.1.3 Constraints

**Obstacle Avoidance**

$$\bigvee_{i=1}^{F_j} l_{ij}^\top x_t + b_{ij} > 0 \iff \bigwedge_{i=1}^{F_j} \left( l_{ij}^\top x_t + b_{ij} + Mz_{ij}^t > 0 \right) \tag{12}$$

where :

- $l_{ij}$:Defines the linear boundary conditions of an obstacle's face.
- $b_j$: Represents the offset of the boundary from the origin.
- $M$: Big-M constant for binary relaxation.
- $z_{j,k}^{(i)} \in \{0, 1\}$ ensures logical constraints.

**Inter-Agent Collision Avoidance**

$$\|x_k^{(i)} - x_k^{(j)}\| \geq d_{\min} \quad \forall i \neq j \tag{13}$$

## 3.2 Discrete-Time MILP Formulation for Gurobi Solver

To enable practical implementation and solver compatibility, the general MILP problem is discretized in time and structured for integration with Gurobi, a widely used optimization solver for MILP problems.

### 3.2.1 Linearization Unicycle Model Dynamics

We used a unicycle model with a linearization around a nominal trajectory to represent the state dynamics of our robots. We start from our Euler discretization model, for each robot we have the following system:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \phi_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \phi_k \end{bmatrix} + \Delta t \cdot \begin{bmatrix} v_k \cos(\phi_k) \\ v_k \sin(\phi_k) \\ \omega_k \end{bmatrix} \tag{14}$$

We can expressed our system dynamics as : $x_{k+1} = f(x_k, u_k)$ with :

$$f(x_k, u_k) = \begin{bmatrix} x_k + \Delta t v_k \cos(\phi_k) \\ y_k + \Delta t v_k \sin(\phi_k) \\ \phi_k + \Delta t \omega_k \end{bmatrix}. \tag{15}$$

We then linerazied around the nominal trajectory with $\bar{x_k} = [\bar{x_k}, \bar{y_k}, \bar{\phi_k}]$ and $\bar{u_k} = [\bar{v_k}, \bar{w_k}]$ to obtain our state dynamics :

$$A_k = \frac{\partial f}{\partial x_k}\bigg|_{(\bar{x}_k, \bar{u}_k)} = \begin{bmatrix} 1 & 0 & -\Delta t v_k \sin(\bar{\phi}_k) \\ 0 & 1 & \Delta t v_k \cos(\bar{\phi}_k) \\ 0 & 0 & 1 \end{bmatrix}. \quad B_k = \frac{\partial f}{\partial u_k}\bigg|_{(\bar{x}_k, \bar{u}_k)} = \begin{bmatrix} \Delta t \cos(\bar{\phi}_k) & 0 \\ \Delta t \sin(\bar{\phi}_k) & 0 \\ 0 & \Delta t \end{bmatrix}. \tag{16}$$

We finally obtain a Linear time-varying (LTV) system, namely :

$$x_{k+1} = A_k \cdot x_k + B_k \cdot u_k \tag{17}$$

### 3.2.2 Cost function

We implemented multiple cost functions to ensure the feasibility, controllability, and smoothness of our trajectories. First, we defined a cost function to prioritize reaching the goal position accurately :

$$C_1 = \sum_{i=1}^{L} \sum_{k=0}^{N-1} w_{goal} \cdot ||x_{k+1}^{(i)} - x_{goal}^{(i)}||^2 \tag{18}$$

Next, we added a cost term to minimize displacement between consecutive points, ensuring a smoother trajectory :

$$C_2 = \sum_{i=1}^{L} \sum_{k=0}^{N-1} w_{smooth} \cdot ||x_{k+1}^{(i)} - x_k^{(i)}||^2 \tag{19}$$

Additionally, a cost was imposed on reference trajectory tracking to keep the robots as close as possible to their desired reference trajectories :

$$C_3 = \sum_{i=1}^{L} \sum_{k=0}^{N-1} w_{ref} \cdot ||x_k^{(i)} - \bar{x}_k^{(i)}||^2 \tag{20}$$

Finally, we included a leader-following cost weight to maintain the formation's structure, ensuring the follower robots stay within a specific position relative to the leader robot ($r_{lf}^{(i)}$).

$$C_4 = \sum_{i=1}^{L} \sum_{k=0}^{N-1} w_{leader} \cdot ||x_k^{(i)} - x_k^{(leader)} - r_{lf}^{(i)}||^2 \tag{21}$$

We eventually obtained the following cost function for our system :

$$min \quad C_1 + C_2 + C_3 + C_3$$

### 3.2.3 Constraints

**State Dynamics**

$$x_{k+1} = A_k \cdot x_k + B_k \cdot u_k \tag{22}$$

**Initial Conditions**

$$x_{k=0}^{(i)} = x_{initial}^{(i)} \quad \forall i \in L \tag{23}$$

**Obstacles Avoidance**

$$\bigwedge_{i=1}^{F_j} \left( l_{ij}^\top x_t + b_{ij} + M z_{ij}^t > 0 \right) \tag{24}$$

with z, binary variable ensuring logical constraints.

**Inter-Agent Collision Avoidance**

$$||x_k^{(i)} - x_k^{(j)}||_2 \geq d_{\min} \quad \forall i \neq j \tag{25}$$

where $d_{min}$ is equal to two times the radius of the robot times a safety factor.

# 4 Simulations

For the simulations, we focused on trajectory planning using the MILP algorithm with multiple robots, multiple obstacles, and formation control, where the map, obstacles, and robot dynamics were adapted from real-world scenarios observed in laboratory environments.

## 4.1 Scenarios

**Scenario 1: Single Robot with one Obstacle**
In the first scenario, we simulate only one robot, with an initial and goal position and an obstacle between these two points. The reference trajectory is based on the shortest trajectory between the initial and goal points without considering an obstacle. We have the following conditions:

$$x_{k=0}^{(0)} = (3, 0.41) \quad x_{k=N}^{(0)} = (-3, 0.41)$$

$$L_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \quad b_0 = \begin{bmatrix} -2 & 0 \\ 0 & -0.6 \\ 0 & -0.4 \\ 1 & 0 \end{bmatrix} \quad M = 5000$$

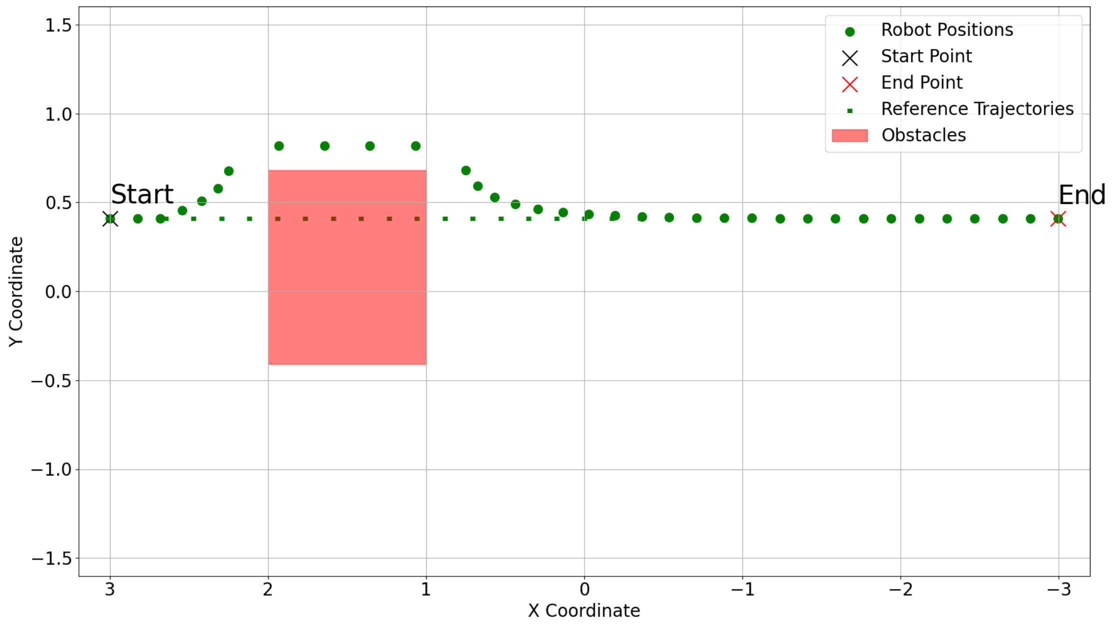


Figure 6: Trajectory Optimization of One Robot with one Obstacle with MILP

As we can observe on figure 6, the robot is avoiding the obstacle while maintaining the minimum distance with the reference trajectory. The optimized trajectory (green dots) deviates from this line to avoid the obstacle, indicating that constraints are enforced. The trajectory shown seems to follow the shortest feasible route around the obstacle while satisfying constraints.

**Scenario 2: Single Robot with multiple Obstacles**
In the second scenario, we simulate only one robot, with an initial and goal position and three obstacles between these two points. The reference trajectory is based on the shortest trajectory between the initial and goal points without considering obstacles. We have the following obstacles constraints:

$$L_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \quad b_0 = \begin{bmatrix} -2 & 0 \\ 0 & -0.6 \\ 0 & -0.4 \\ 1 & 0 \end{bmatrix}$$

$$L_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \quad b_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0.0 \\ 0 & -0.95 \\ -2 & 0 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \quad b_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1.2 \\ 0 & 0.3 \\ -2 & 0 \end{bmatrix}$$

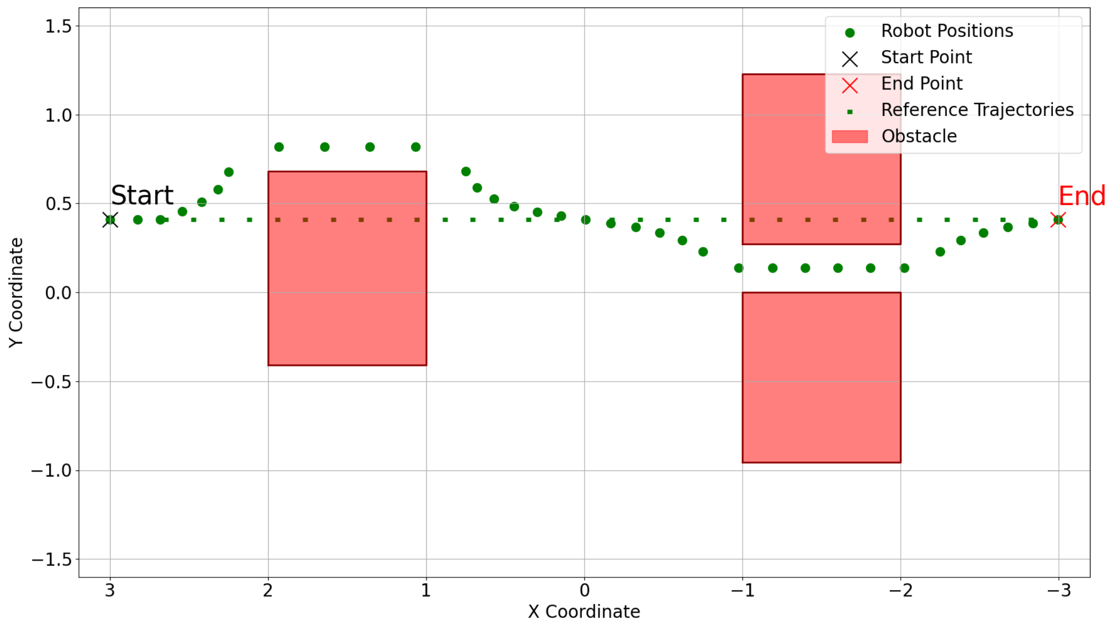


Figure 7: Trajectory Optimization of One Robot with Multiple Obstacles with MILP

As we can observe on figure 7, the robot is avoiding the two obstacles on his trajectory while maintaining the minimum distance with the reference trajectory. The optimized trajectory deviates from this line to avoid the obstacle, indicating that constraints are enforced.

**Scenario 3: Multiple Robots with one Obstacle**
For this scenario, we use 3 different robots with only one obstacle (same obstacles as 6. They are starting in a triangle formation and the objective is to maintain this formation while avoiding obstacles. We have the following initial and final conditions :

$$x^{(0)}_{k=0} = (3, 0.41) \quad x^{(0)}_{k=N} = (-3, 0.41)$$

$$x^{(1)}_{k=0} = (3, -0.14) \quad x^{(1)}_{k=N} = (-3, -0.14)$$

$$x^{(2)}_{k=0} = (3.5, 0.14) \quad x^{(2)}_{k=N} = (-2.5, 0.14)$$

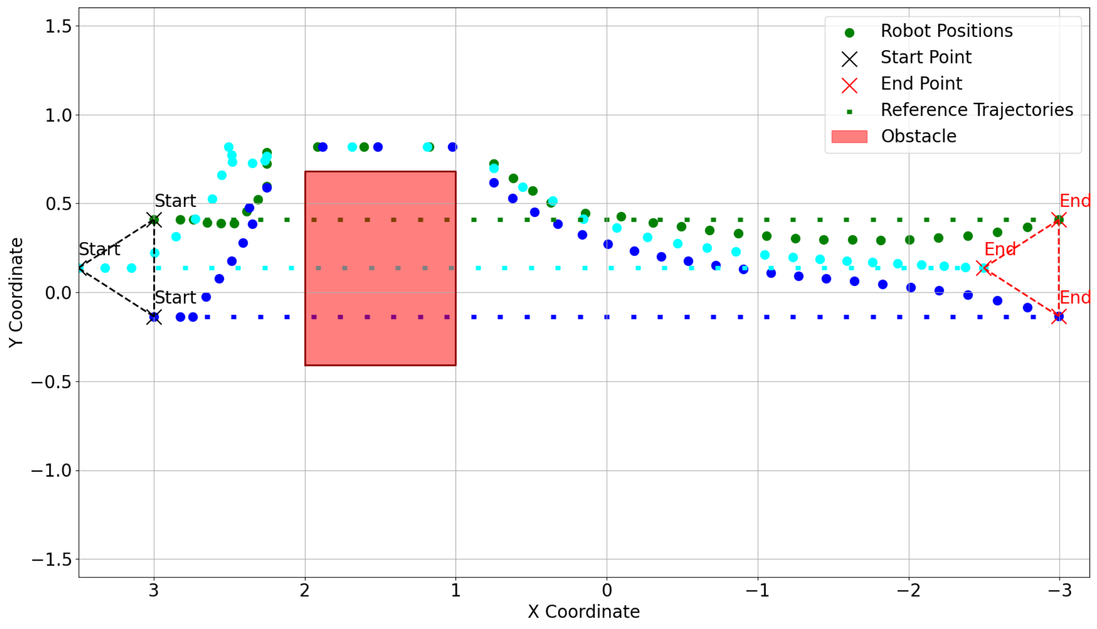


Figure 8: Trajectory Optimization of three Robot with one Obstacle with MILP

From figure 8 we can first observe that the three robots are reaching their final positions while avoiding the obstacle. The second key observation concerns their trajectories: all the robots avoid the obstacle by moving upward. Ideally, one of the robots should have moved downward, as it represents the shortest trajectory to the goal. However, due to the leader cost associated with the first robot positioned above, they all follow the same upward trajectory.

**Scenario 4: Multiple Robots with Multiple Obstacles**
In this scenario, we simulate three distinct robots navigating around three separate obstacles. The goal of the simulation is twofold: first, to ensure the robots successfully avoid all three obstacles, and second, to have them temporarily break formation when passing through a narrow trajectory and then seamlessly reform their original arrangement afterward. We have the same matrices than scenario 2 and the boundary conditions are :

$$x^{(0)}_{k=0} = (3, 0.41) \quad x^{(0)}_{k=N} = (-2.5, 0.41)$$

$$x^{(1)}_{k=0} = (3, -0.14) \quad x^{(1)}_{k=N} = (-2.5, -0.14)$$

$$x^{(2)}_{k=0} = (2.5, 0.14) \quad x^{(2)}_{k=N} = (-3, 0.14)$$

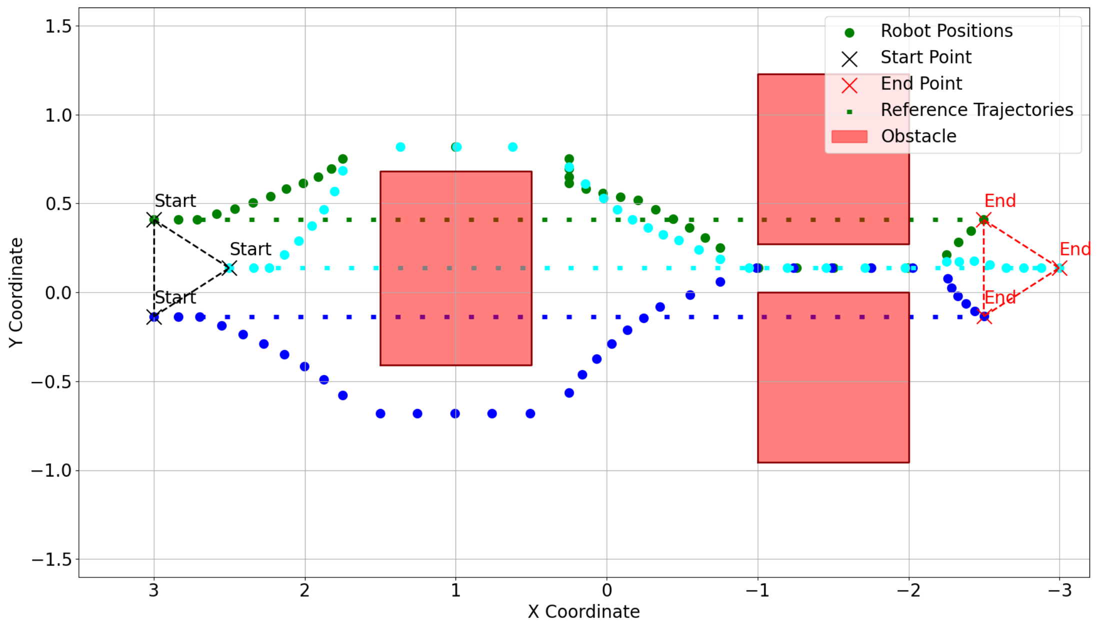
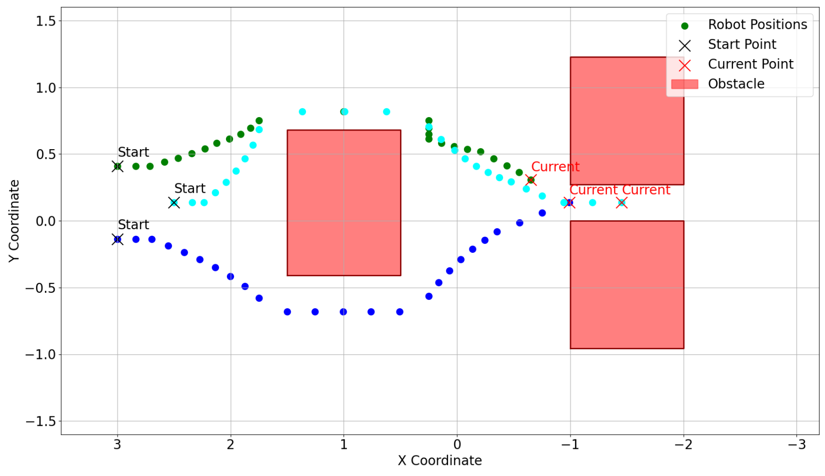


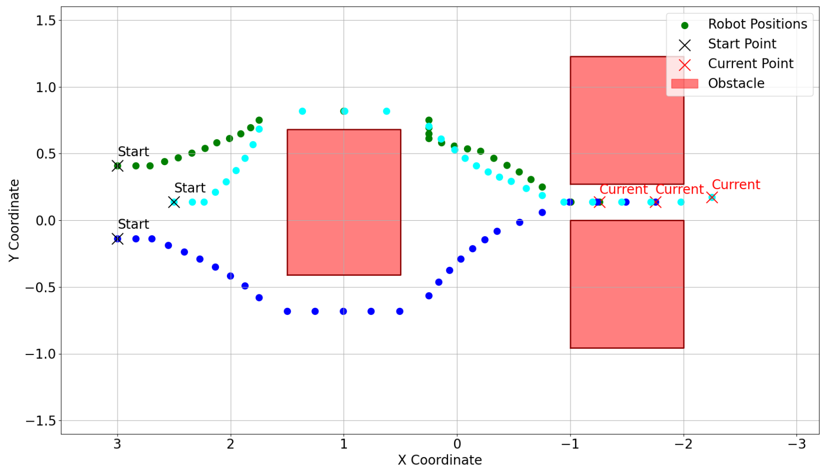
Figure 9: Trajectory Optimization of three Robot with three Obstacles with MILP

From figure 9 we can first observe that the three robots are reaching their final positions while avoiding the three obstacles. We can also notice that only two robots out of three, avoid the obstacle by moving upward instead of all of them in scenario 3. From figure 10 we can observe the evolution of the three robots when they reach the narrow trajectory. In fact, at this stage when the formation needs to be broken to avoid obstacles and inter-collisions, we observe that the three robots are not entering the narrow trajectory at the same time and with the triangle formation. Depending on the cost associated with each of the conditions (**??**),(19), (20),(21) the optimizer will choose which conditions prioritize and so which trajectory to follow.

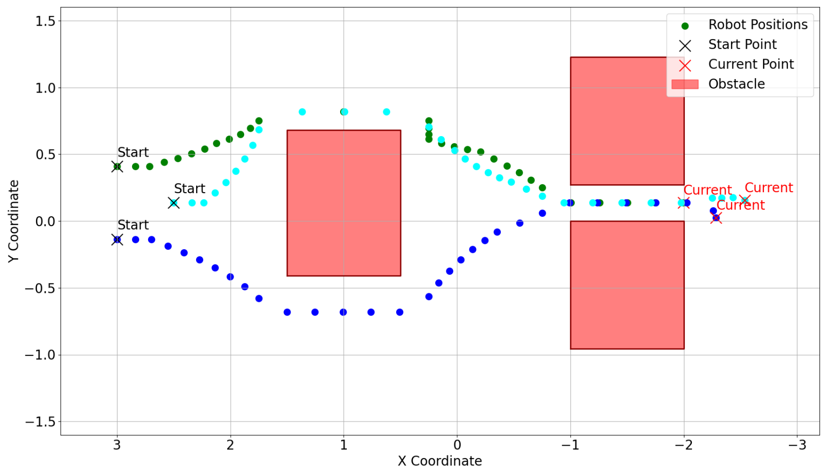
Other simulations have been conducted with different numbers of robots and different formations such as line formation with two robots (17, 18, 19) and diamond formation with four robots (20, 21, 22) to validate our model.



(a) $t = 18s$



(b) $t = 19s$



(c) $t = 20s$

Figure 10: Trajectory Evolution of Three Robots Through a Narrow Path Using MILP

## 4.2 Influence of cost weights on MILP

As highlighted in the various scenarios, the trajectories taken by each robot depend significantly on the weight assigned to each function. That's why we decided to study the influence of two different weights :

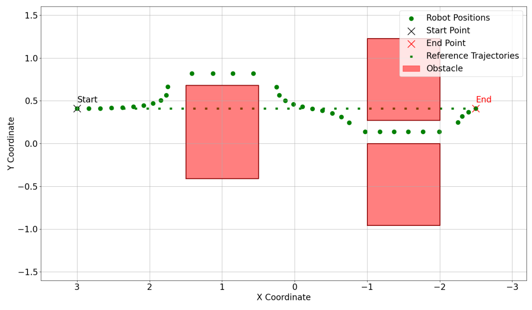
1. $w_{ref}$ : the weight assigned to following the reference trajectories
2. $w_{leader}$ : the weight assigned to following the leader robot

### 4.2.1 Influence of reference trajectory weight on MILP trajectory Planning

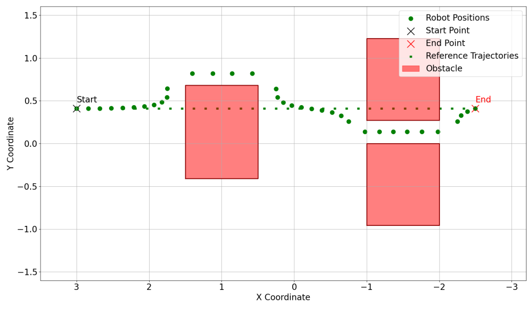
We will first study the impact of reference trajectory with one single robot with multiple obstacles. These simulations will enable to find for which values of weight the robot continue to avoid obstacles or choose to follow only the reference trajectory. We want to observe the impact of the reference trajectory cost compared to the other costs so we will normalize our cost to have a better understanding :

$$w^*_{ref} = \frac{w_{ref}}{\sum w_i} = \frac{w_{ref}}{w_{ref} + w_{smooth} + w_{goal} + w_{leader}}$$

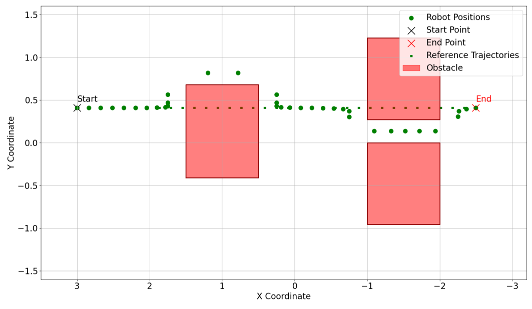
We will simulate with four different weights :



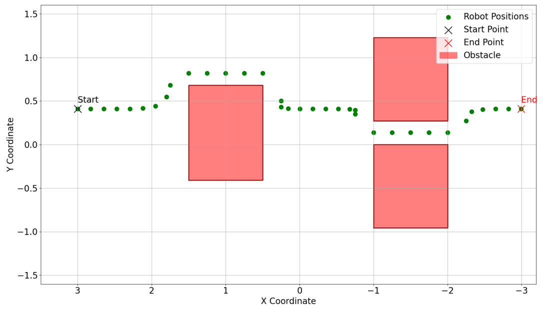
(a) $w^*_{ref} = 0.01$



(b) $w^*_{ref} = 0.1$



(c) $w^*_{ref} = 0.3$



(d) $w^*_{ref} = 0.5$

Figure 11: Trajectory Planning using MILP with four different reference trajectory weights

From Figure 11, we observe that increasing $w_{ref}$ causes the robot to prioritize following the reference trajectory more closely. This results in straighter and less curvier trajectories, which can impact smoothness. Higher $w_{ref}$ values (e.g., 11d for $w^*_{ref} = 0.5$) lead to trajectories that align more strictly with the reference but may compromise smoothness compared to smaller $w_{ref}$ values (e.g., 11a, 11b).

### 4.2.2 Influence of leader weight on MILP trajectory Planning

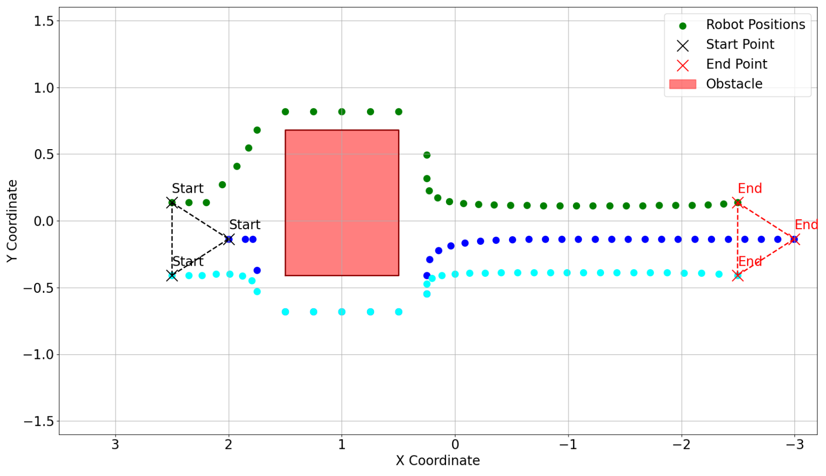
We will now study the impact of leader cost on trajectory planning with MILP. We have selected three different weights, showing three cases that can be simulated for our trajectory planner. We first simulated with only one obstacle and three robots. For the weight associated with leader-follower, we use the same method as the reference weight:

$$w^*_{leader} = \frac{w_{leader}}{\sum w_i} = \frac{w_{leader}}{w_{ref} + w_{smooth} + w_{goal} + w_{leader}}$$

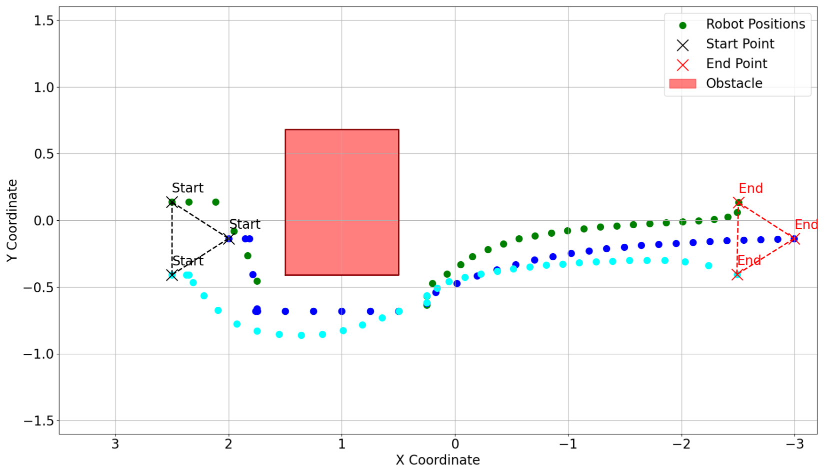
The three scenarios from figure 12 illustrate the impact of varying the leader-following weight ($w_{leader}$) on the behavior of three robots navigating around an obstacle while maintaining formation. In the first scenario 12a ($w^*_{leader} = 0.1$), the weight is insufficient, causing the follower robots to deviate from the leader's trajectory, prioritizing smoothness and goal-reaching over formation maintenance.

In the second scenario 12b ($w^*_{leader} = 0.3$), the robots maintain a cohesive formation but when they avoid the obstacles they break their formation to focus on minimizing the distance (reference trajectories), with one robot intelligently passing below the obstacle to preserve the group's structure. This scenario represents a well-tuned balance of objectives.

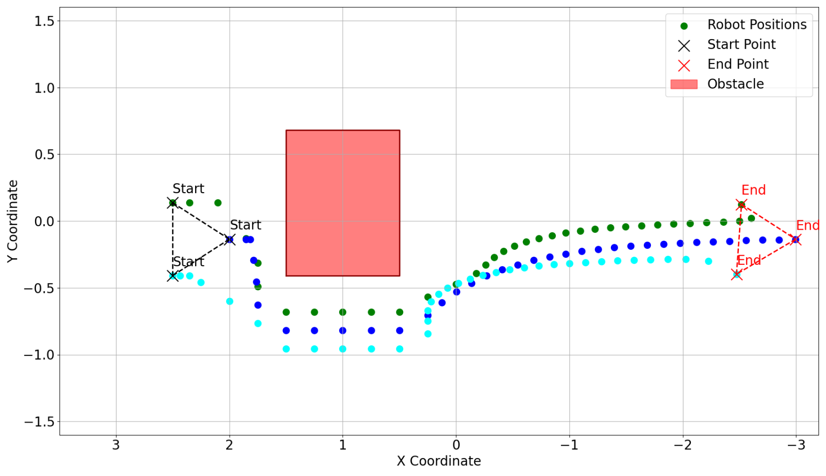
In contrast, in the third scenario 12c ($w^*_{leader} = 0.5$), the three robots successfully maintain their formation at nearly every time step. It can be observed that the two followers prioritize maintaining the formation over minimizing the deviation from the reference trajectory. This behavior occurs because the formation cost is given greater importance than the trajectory-following cost.



(a) $w^*_{leader} = 0.1$



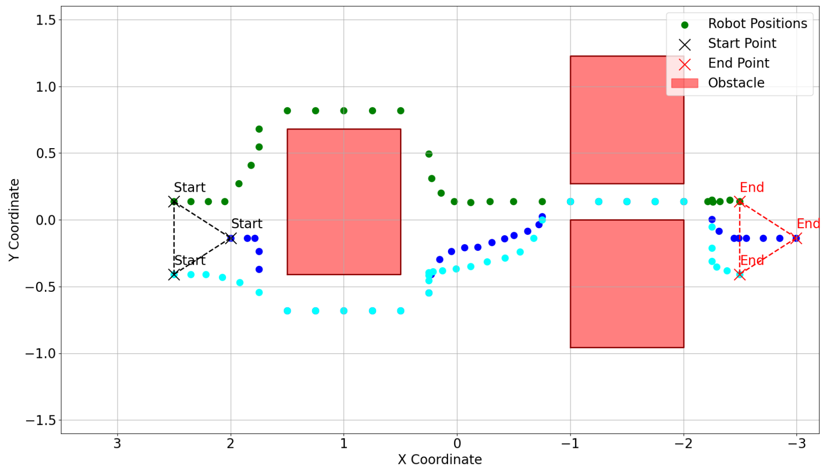
(b) $w^*_{leader} = 0.3$



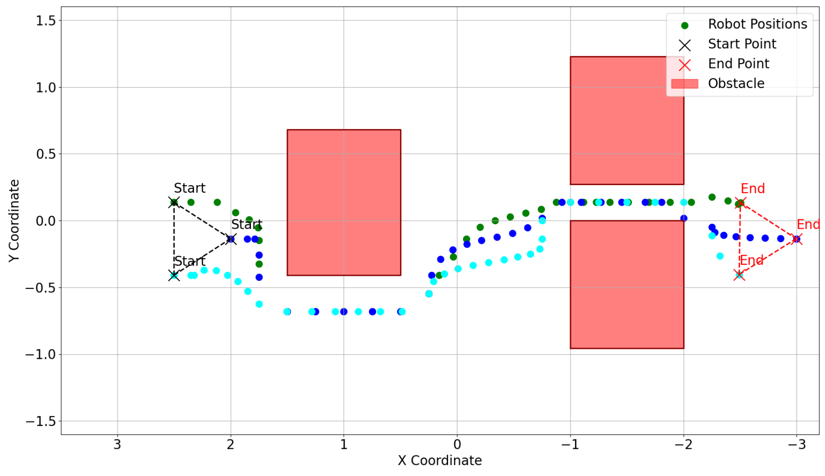
(c) $w^*_{leader} = 0.5$

Figure 12: Trajectory Evolution with one obstacle and three different leader weights

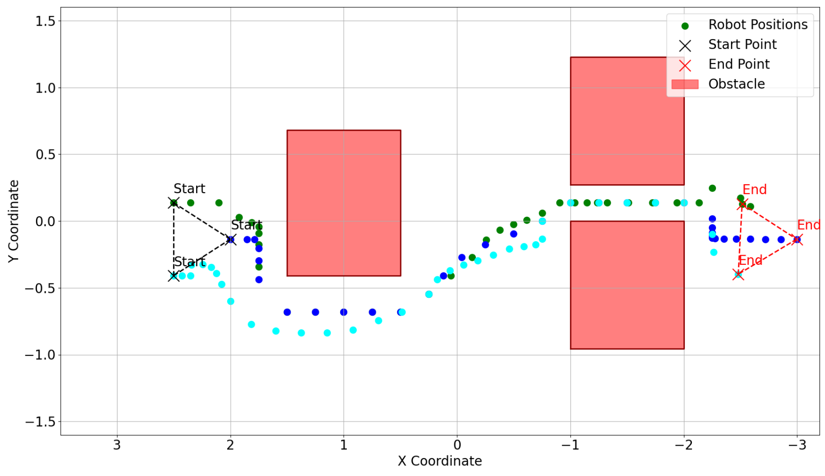
We then simulated with three obstacles to confirm our choice of leader weight.



(a) $w^*_{leader} = 0.1$



(b) $w^*_{leader} = 0.3$



(c) $w^*_{leader} = 0.5$

Figure 13: Trajectory Evolution with three obstacles and three different leader weights

The three scenarios from Figure 13 demonstrate how varying the leader-following weight ($w_{leader}$) impacts the trajectories of three robots navigating a trajectory with **three obstacles** while maintaining formation.

In the first scenario 13a ($w^*_{leader} = 0.1$), the leader-following weight is too low, causing the robots to focus primarily on avoiding obstacles and following smooth individual trajectories. The formation is preserved but not strongly emphasized, as the robots prioritize smooth trajectories over strictly maintaining their triangular formation.

In the second scenario 13b ($w^*_{leader} = 0.3$), the robots all deviate downward, avoiding obstacles while maintaining their triangular formation. However, the weight on formation is insufficient to highlight the triangular pattern clearly, as the focus shifts to minimizing the reference trajectory. The robots' trajectories demonstrate a balanced trade-off between maintaining formation and obstacle avoidance, but the formation goal is secondary.

In the third scenario 13c ($w^*_{leader} = 0.5$), the leader-following weight is high enough to prioritize maintaining the triangular formation even at the expense of closely following a reference trajectory. The robots preserve the formation effectively, with clear coordination among the group, although the trajectories deviate from the reference trajectory due to the increased emphasis on formation.

Overall, the second scenario ($w^*_{leader} = 0.1$) strikes a reasonable balance, but the third scenario ($w^*_{leader} = 0.2$) demonstrates the strongest emphasis on formation retention at the cost of less adherence to the reference trajectory. This trade-off highlights the importance of tuning $w_{leader}$ based on the desired objectives for formation control and trajectory following.

## 4.3 Computational performances of MILP

We aim to analyze the **trajectory planning computation time using MILP** and investigate its dependency on two key factors: the **number of robots** and the **number of obstacles**. Building on insights from our previous analysis, we carefully selected the following parameters for our study:

$$w^*_{ref} = 0.1, \quad w^*_{leader} = 0.1, \quad w^*_{goal} = 0.35, \quad w^*_{smooth} = 0.45$$

The evaluation is conducted across **twelve distinct scenarios**, varying both the **number of obstacles (1 to 3)** and the **number of robots (1 to 4)**. This systematic approach ensures a comprehensive understanding of how these variables influence computational performance in multi-agent trajectory planning.

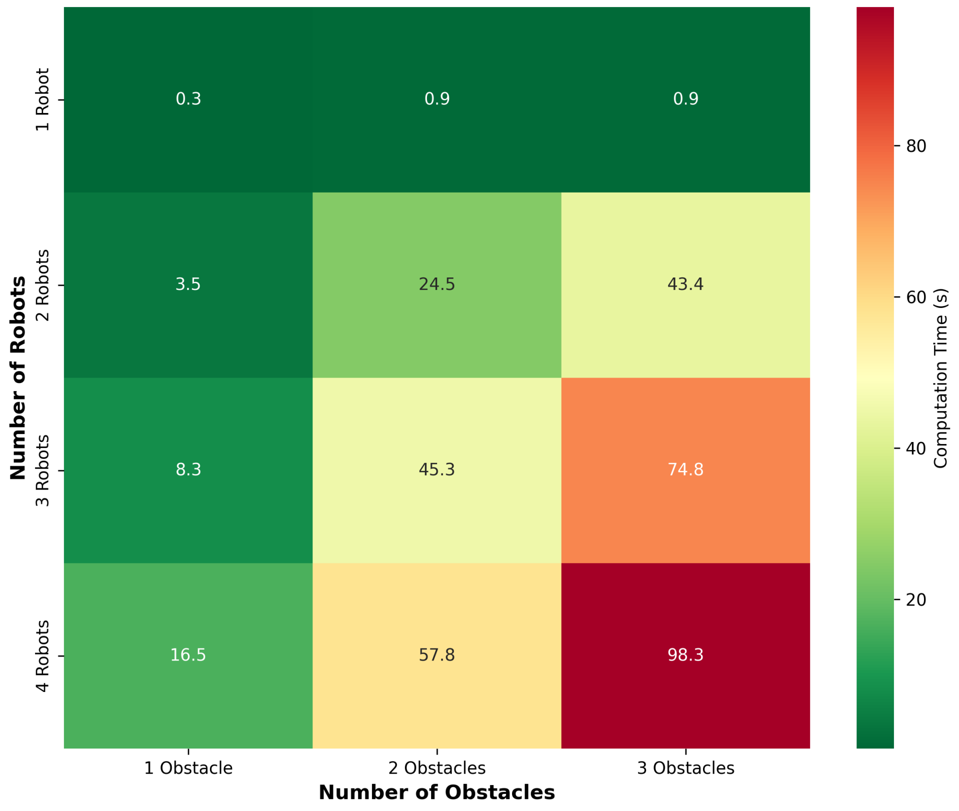


Figure 14: Computation Time Heatmap for Trajectory Planning (MILP)

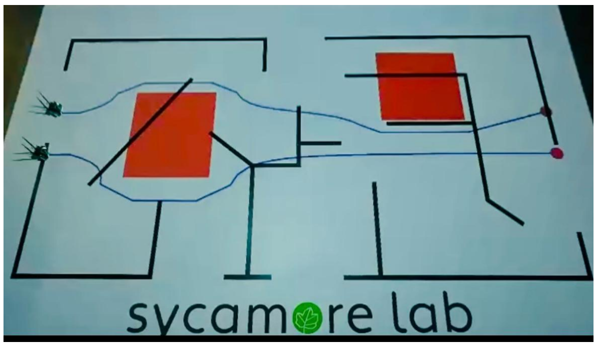
The heatmap on figure 14 reveals a clear trend: as the number of robots and obstacles increases, the computation time for solving the trajectory-planning problem using MILP (Mixed-Integer Linear Programming) grows significantly. For a single robot and a minimal number of obstacles, computation time remains low, indicating manageable computational complexity. However, as the number of robots and obstacles increases, the computation time scales non-linearly, with a sharp rise observed beyond 2 robots and 2 obstacles. The highest computation time (98.3 seconds) occurs with 4 robots navigating through 3 obstacles, highlighting a significant computational bottleneck.

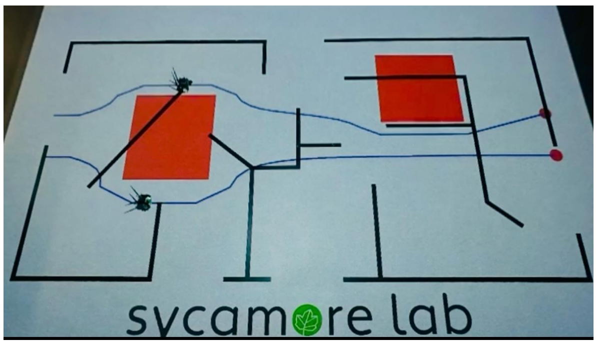
However, despite this significant growth, there is a notable improvement in computation time compared to simulations using CCILQ Games (Chance-Constrained Iterative Linear Quadratic Games). Indeed, even with an increasing number of obstacles, simulations using MILP (Mixed-Integer Linear Programming) consistently remain less time-consuming than those based on CCILQ Games for similar configurations. This indicates that, while MILP faces scalability challenges, it remains a more efficient approach in terms of computation time for complex scenarios involving multiple robots and obstacles. This observation highlights the potential of MILP for applications where the balance between precision and computation time is critical, especially in environments requiring fast and reliable decision-making.

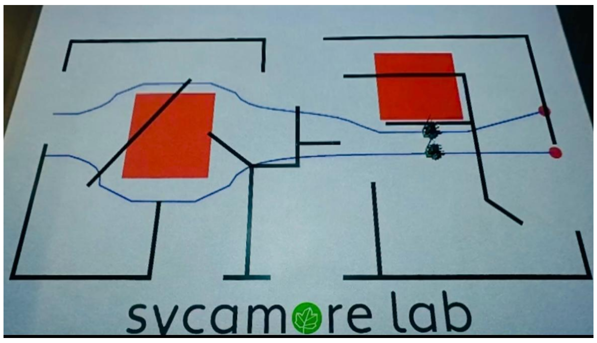
## 5 Applications

To validate our simulations, we conducted tests using real robots with obstacles projected onto the ground, utilizing NVIDIA JetBots as our robotic platform. We evaluated two distinct scenarios: the first scenario was relatively simple, involving two robots navigating around two obstacles, with a low leader-follower cost allowing more flexibility in their formation. The second scenario was more complex, featuring three robots and three obstacles navigating through a narrow trajectory. In this case, a high leader-follower cost was imposed, requiring the robots to strictly maintain their formation throughout the trajectory.

### 5.1 Two Robots with two Obstacles


(a) $t = 0s$


(b) $t = 9s$


(c) $t = 21s$

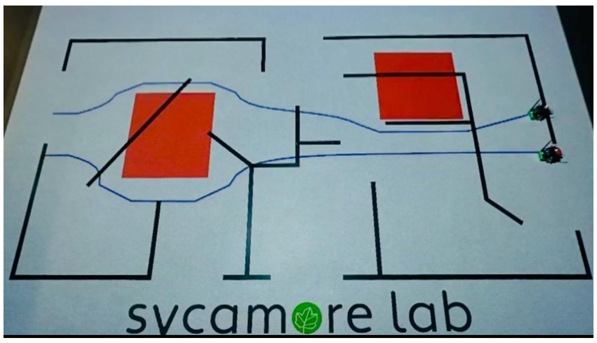
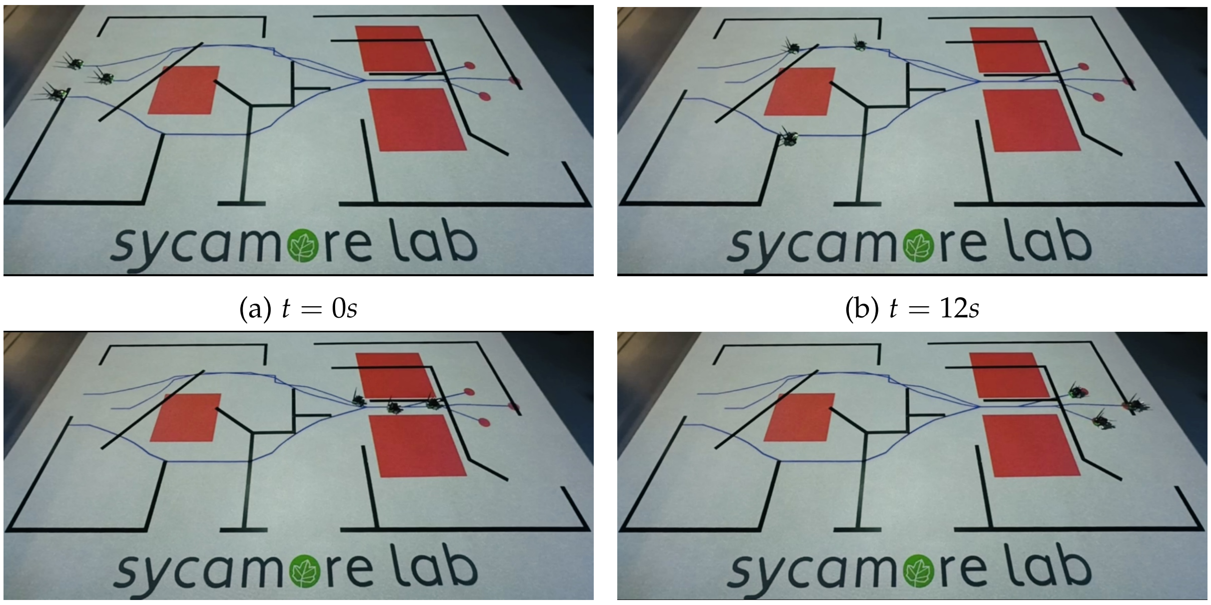

(d) $t = 24s$

Figure 15: Application with two robots and two obstacles

From figure 15, we can observe the evolution of the trajectories of the two robots as they successfully navigate toward their goals while avoiding obstacles. In the first phase, from $t = 0s$ (15a) to $t = 9s$ (15b) they are breaking their formation to avoid the obstacle then from $t = 9s$ (15b) to $t = 21s$ (15c) they are rebuilding their formation and in the last part of their trajectory from $t = 21s$ (15c) to $t = 25s$ (15d) they are maintaining their formation while avoiding the second obstacle until reaching their goal positions.

## 5.2 Three Robots with Multiple Obstacles



(a) $t = 0s$ (b) $t = 12s$

(c) $t = 24s$ (d) $t = 28s$

Figure 16: Application with three robots and three obstacles

From Figure 16, we observe the evolution of the trajectories of the three robots as they successfully navigate toward their goals while avoiding obstacles. In the first phase, from $t = 0$ s (Figure 16a) to $t = 12$ s (Figure 15b), the robots break their formation to avoid the obstacle: one robot moves downward while the other two move upward to bypass the obstacle. During the second phase, from $t = 12$ s (Figure 16b) to $t = 21$ s (Figure 16c), they begin to rebuild their formation. Finally, in the last segment of the trajectory, from $t = 21$ s (Figure 16c) to $t = 25$ s (Figure 16d), the formation is broken again to navigate a narrow passage. The leader robot passes through first, followed by the other two robots, ensuring collision avoidance.

# 6 Discussion

The results of this study demonstrate the strengths and limitations of two prominent methodologies for multi-agent trajectory optimization and formation control: CCILQ Games and MILP. Each approach was evaluated based on computational efficiency, scalability, and adaptability to dynamic environments with multiple obstacles and robots.

## 6.1 Performance of CCILQ Games

The CCILQ Games approach provided a robust framework for handling uncertainties in robot dynamics and probabilistic constraints. The iterative optimization process demonstrated precise trajectory planning with effective robot formation control. However, the method's scalability is a notable limitation, as the convergence time increased consequently with the number of robots and complexity. For example, simulations with four robots required over 100 seconds for convergence, making real-time applications challenging. Another aspect to mention is the difficulty in integrating obstacle avoidance constraints into this model, such as simple polygonal obstacles, which is why we turned to alternative models such as Mixed-Integer Linear Programming (MILP).

### 6.2 Performance of MILP

The MILP approach showcased improvements in computational efficiency compared to CCILQ. The discrete linearization of system dynamics and the inclusion of binary variables enabled precise trajectory planning while ensuring collision-free navigation. MILP maintained scalability with a moderate increase in computation time as the number of robots and obstacles grew. The leader-following strategy further demonstrated the flexibility of MILP in maintaining formations under varying cost weights, allowing dynamic adjustments based on environmental complexity. However, the rigid formulation of constraints can limit adaptability in highly dynamic environments, as real-time recalculations may still be computationally expensive.

### 6.3 Trade-offs Between the Approaches

While CCILQ Games excel in handling stochastic scenarios with high precision, their computational demand restricts scalability. Conversely, MILP offers a practical balance between efficiency and reliability, making it more suitable for real-time multi-robot applications. However, tuning the cost weights in MILP requires careful consideration to ensure a balance between formation maintenance, obstacle avoidance, and trajectory smoothness.

## 7 Conclusion and Future Work

This project successfully compared and implemented CCILQ Games and MILP for multi-agent trajectory optimization in dynamic environments. The key findings are as follows:

- CCILQ Games provide robust probabilistic safety guarantees but are computationally intensive, limiting their scalability in scenarios with multiple robots.
- MILP offers superior computational efficiency and scalability, with the ability to handle dynamic formations effectively. However, its performance depends on precise weight tuning and may face challenges in real-time applications with rapid environmental changes.
- Real-world implementations on NVIDIA JetBots validated the effectiveness of the MILP approach, demonstrating its applicability for practical robotic systems in structured environments.

Future research will focus on enhancing the MILP approach by incorporating uncertainties and chance constraints through a hybrid formulation, this would improve its adaptability to dynamic environments. Another research direction will aim to eliminate the need for integer variables [11], reducing computational complexity as the number of robots and objects increases.

## References

[1] J. Ota, “Multi-agent robot systems as distributed autonomous systems,” *Advanced Engineering Informatics*, vol. 20, no. 1, pp. 59–70, Jan. 2006.

[2] International Centre for Defence and Security, “Flying with the Dragons: China’s Global Dominance in Civilian Drones and Risks for Europe,” *Online Article*, 2023. Available: `https://icds.ee/en/flying-with-the-dragons-chinas-global-dominance-in-civilian-drones-and-risks-for-`

[3] H. Zhong, Y. Shimizu, and J. Chen, “Chance-constrained iterative linear-quadratic stochastic games,” *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 440–447, Jan. 2023.

[4] M. Bhatt, Y. Jia, and N. Mehr, “Efficient constrained multi-agent trajectory optimization using dynamic potential games,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7303–7310, Oct. 2023.

[5] K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control,” *Automatica*, vol. 53, pp. 424–440, Oct. 2015.

[6] W. Ren and E. Atkins, “Distributed multi-vehicle coordinated control via local information exchange,” *International Journal of Robust and Nonlinear Control*, vol. 17, no. 10, pp. 1002–1033, Nov. 2006. DOI: 10.1002/rnc.1147.

[7] W. Dong and J. A. Farrell, “Consensus of multiple nonholonomic systems,” in *Proceedings of the 47th IEEE Conference on Decision and Control*, pp. 2270–2275, 2008.

[8] S. Coogan and M. Arcak, “Scaling the size of a formation using relative position feedback,” *Automatica*, vol. 48, no. 10, pp. 2677–2685, Oct. 2012.

[9] K.-K. Oh and H.-S. Ahn, “Distance-based formation control using Euclidean distance dynamics matrix: three-agent case,” in *Proceedings of the 2011 American Control Conference*, pp. 4810–4815, 2011.

[10] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, “Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1475–1482, Aug. 2020.

[11] X. Zhang , A. Liniger , and F. Borrelli “Optimization-Based Collision Avoidance” in *: IEEE Transactions on Control Systems Technology*, pp.,972-983, May. 2021

[12] A. Prorok “A Framework for Real-World Multi-Robot Systems Running Decentralized GNN-Based Policies,” *Online Video*, 2020. Available: `https://www.youtube.com/watch?v=COh-WLn4iO4`

# Appendices

## Trajectory planning with Line Formation

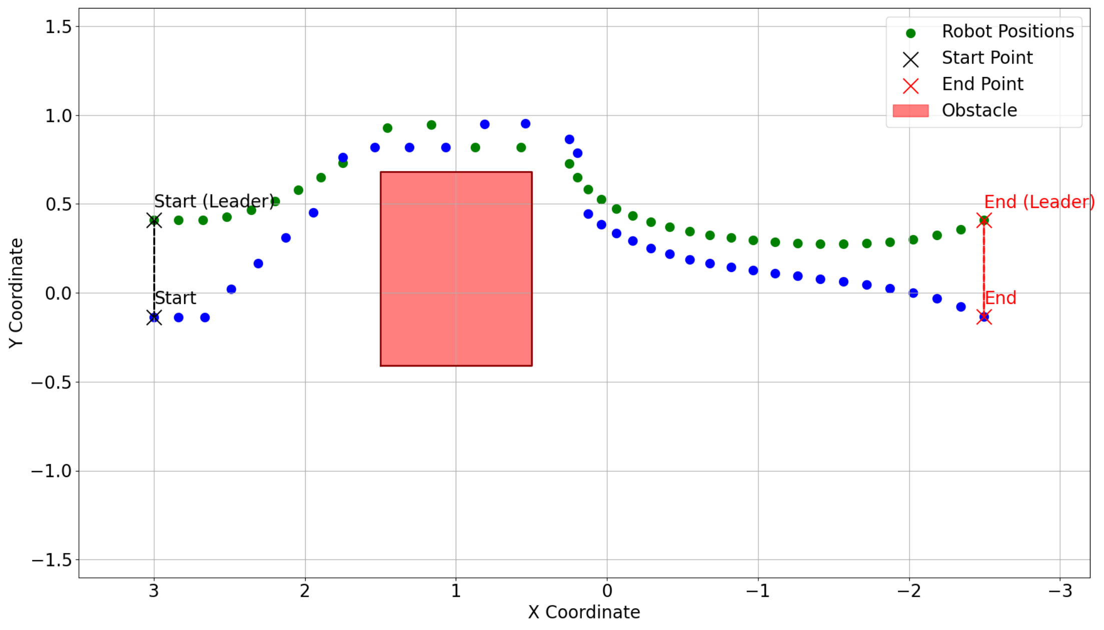


Figure 17: Trajectory Optimization of two robots with one obstacle with MILP

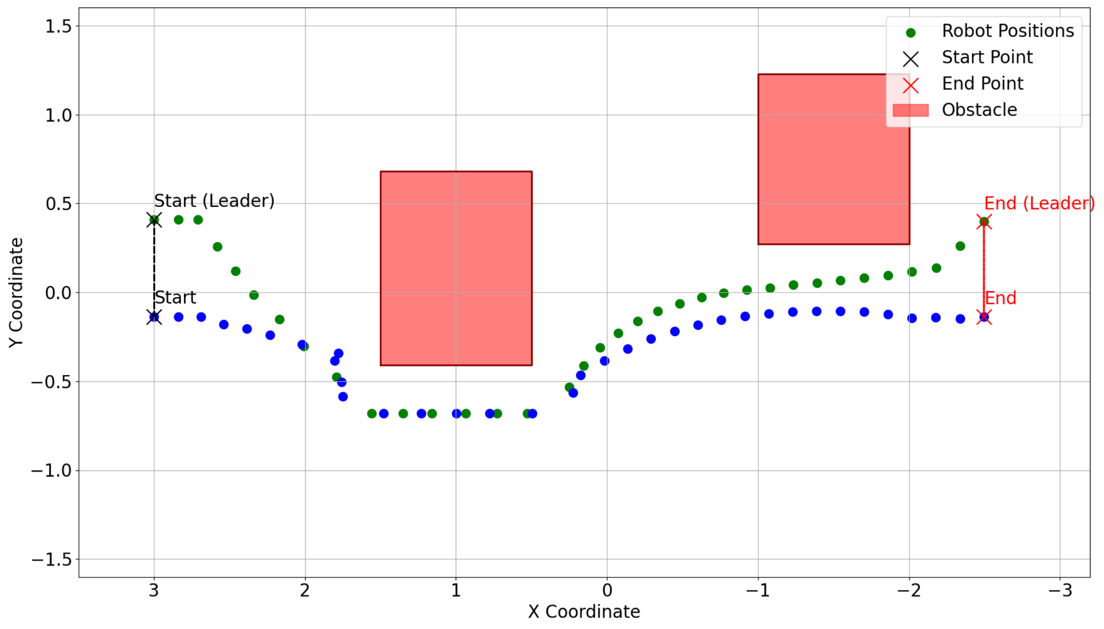


Figure 18: Trajectory Optimization of two robots with two obstacles with MILP

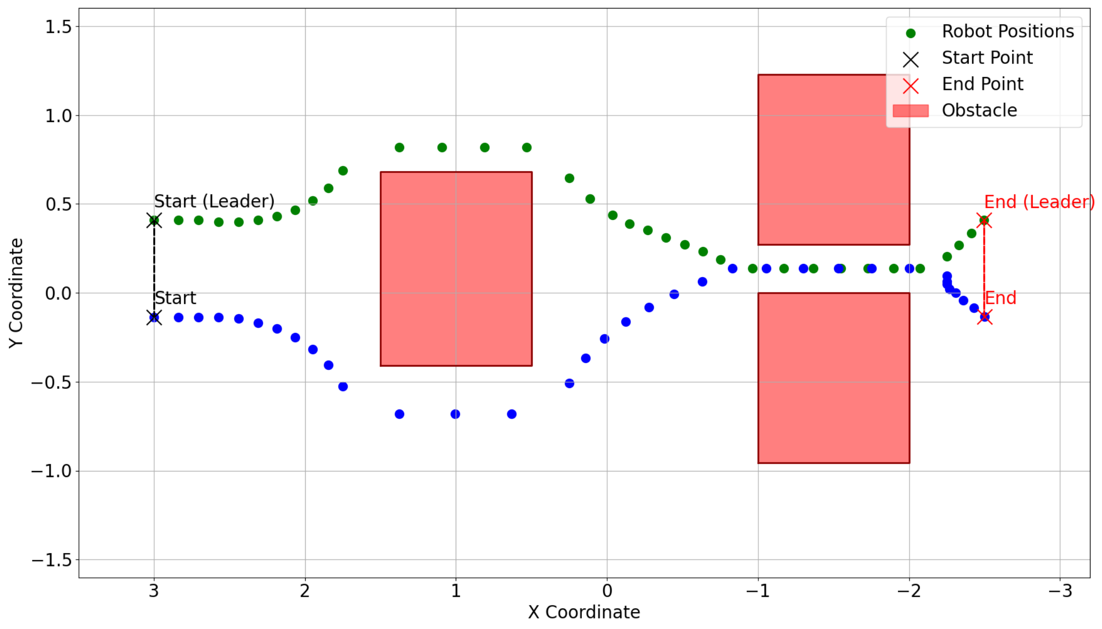


Figure 19: Trajectory Optimization of two robots with three obstacles with MILP

### Trajectory planning with Diamond Formation

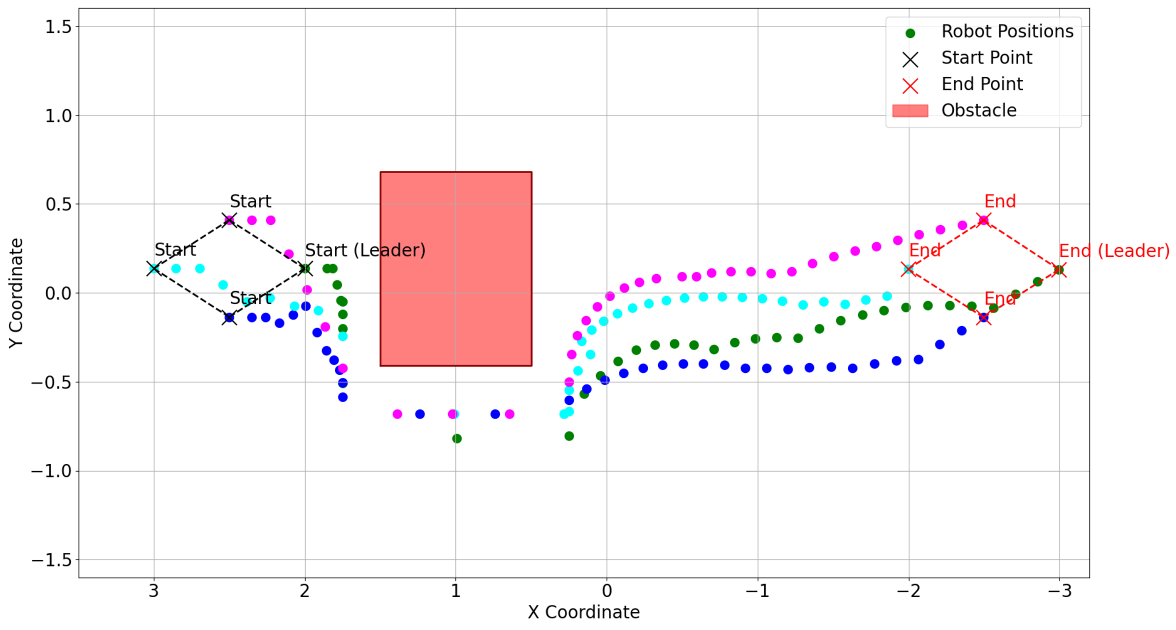


Figure 20: Trajectory Optimization of four robots with one obstacle with MILP

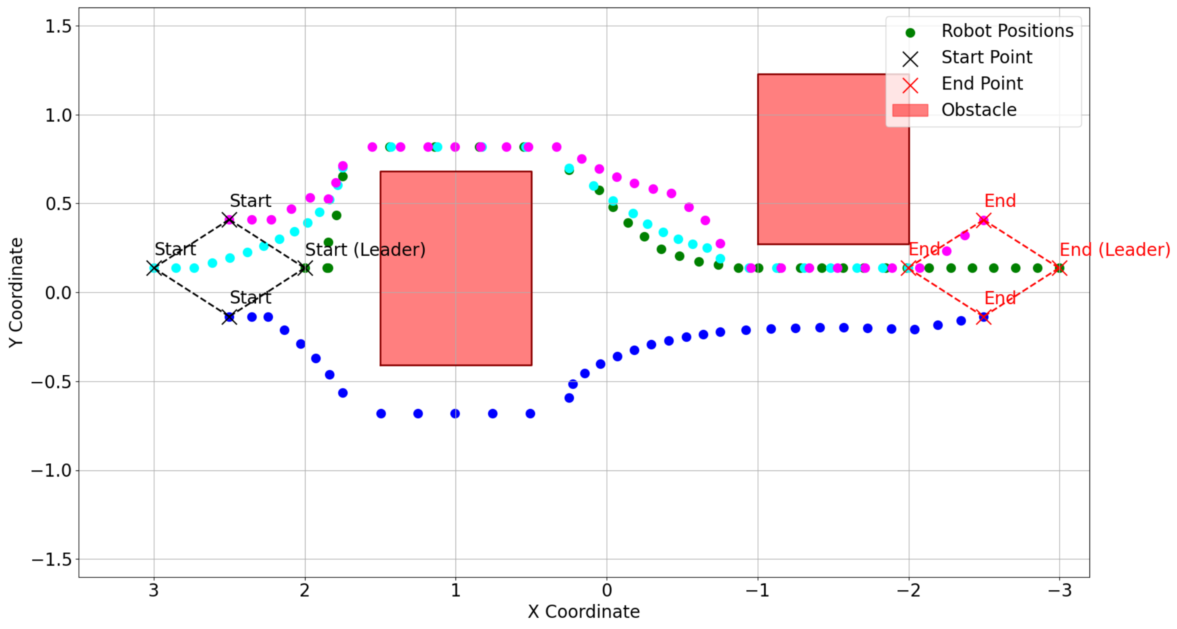


Figure 21: Trajectory Optimization of four robots with two obstacles with MILP

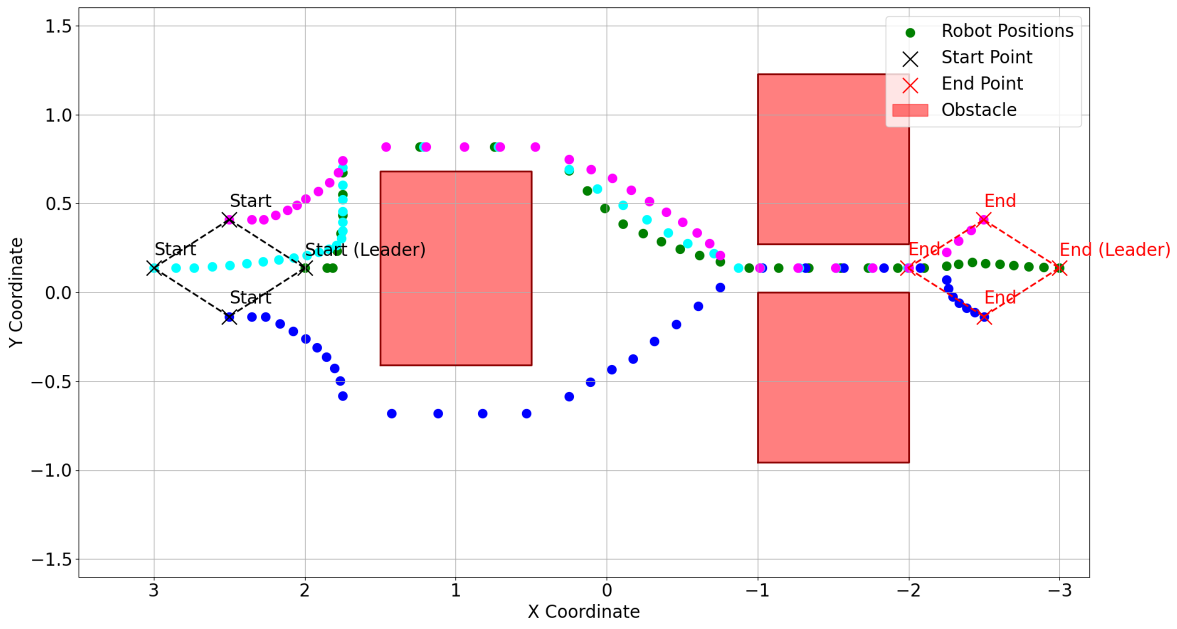


Figure 22: Trajectory Optimization of four robots with three obstacles with MILP

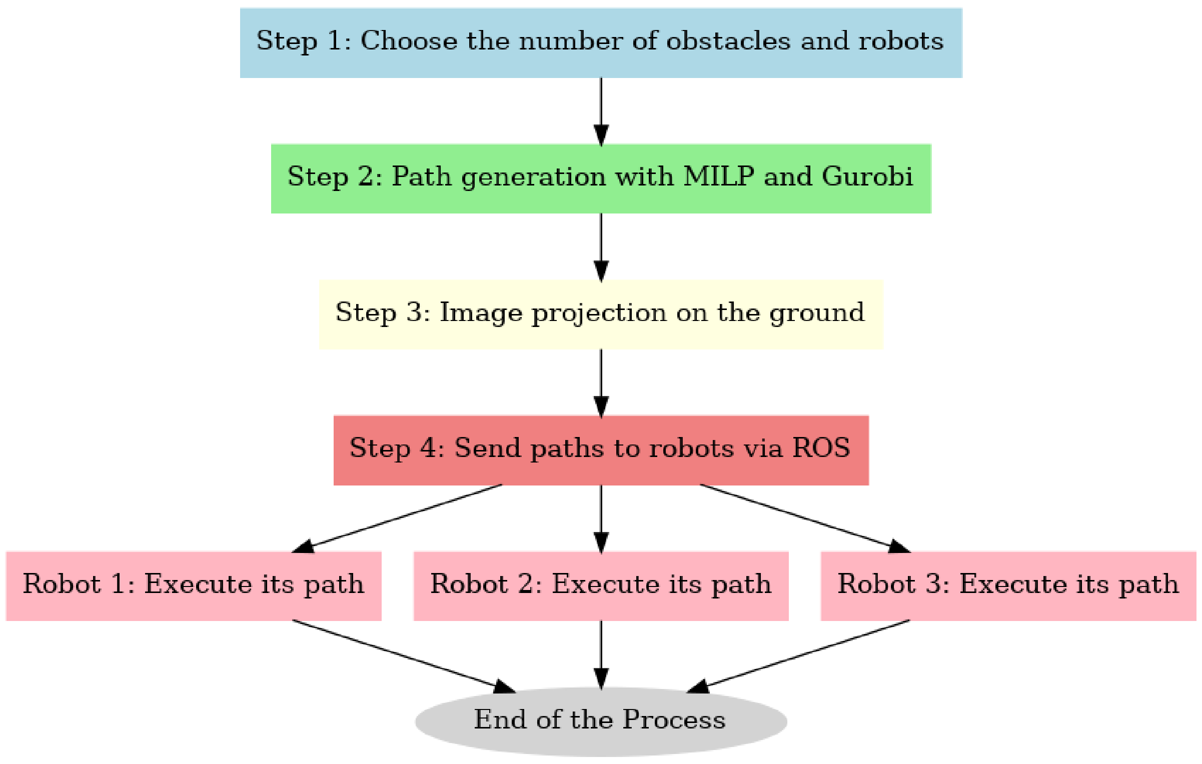

Step 1: Choose the number of obstacles and robots
Step 2: Path generation with MILP and Gurobi
Step 3: Image projection on the ground
Step 4: Send paths to robots via ROS
Robot 1: Execute its path
Robot 2: Execute its path
Robot 3: Execute its path
End of the Process

Figure 23: Flowchart of the process