UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Distributed Autonomous Systems

Course Project Report

Professors:
Giuseppe Notarstefano
Ivano Notarnicola

Student: Daniyar Zhakyp

Academic year 2024/2025

Abstract

This project explores the application of distributed optimization algorithms in the context of multi-robot systems. The primary objective is to address two key tasks: cooperative multi-robot target localization and aggregative optimization for multi-robot systems. Task 1 focuses on the implementation of the Gradient Tracking (GT) algorithm to solve a consensus optimization problem, where multiple robots cooperatively estimate the positions of targets through noisy distance measurements. Various graph topologies are evaluated, and the evolution of the cost function and gradient norms are analyzed to assess the convergence and effectiveness of the algorithm. Task 2 builds on the GT algorithm to implement a distributed control framework that enables robots to move towards their private targets while maintaining team cohesion. This is achieved through the Aggregative Tracking (AT) algorithm, which ensures that the robots stay close to their targets and preserve a tight fleet. The results are visualized using both Python-based plots and ROS 2 simulations, demonstrating the algorithm's performance across different parameter sets. The results of this project enhance the understanding of distributed optimization techniques in robotic systems, providing valuable insights for cooperative localization and distributed control in multi-robot settings.

Contents

Introduction

This project focuses on undertaking two principal tasks in distributed autonomous systems (DAS), which include a cooperative localization of multiple targets in a multi-robot setting and an aggregative tracking of targets' positions, each private to a specific robot. In both tasks, we make the communication between the agents distributed, namely by using a specific graph topology for a communication graph (e.g., star, ring, etc.).

Motivations

In this project, we begin by solving a consensus optimization problem for N agents using quadratic cost functions via the Gradient Tracking (GT) algorithm. We test several graph topologies to check the effectiveness of the implemented framework. This step is a premise for our main task, for which the implemented GT algorithm is employed to identify the targets' positions by the robots based on the noisy measurements of the distance between the two (i.e., target and robot), all incorporated into the problem-designed cost functions. For our second task, we implement the Aggregative Tracking (AT) approach to solve the distributed optimization problem with an aggregative variable related to the robot team's barycenter. In this task, we try to achieve the robots each approaching its private target as close as possible, while preserving the "tightness" condition as a team.

This report is structured as follows:

- **Chapter 1 - Distributed Consensus Optimization**
  This chapter provides a comprehensive explanation of the theoretical tools used for solving the general distributed consensus optimization problem (Task 1.1), including concepts from the graph theory and the GT framework. The mentioned concepts will be helpful in understanding the foremost task about the target localization (Task 1.2) described in Chapter 2. Lastly, this chapter shares the problem's solution in a form of dedicated plots made for various graph topologies.

- **Chapter 2 - Cooperative Multi-Robot Target Localization**
  This chapter employs the notions described in Chapter 1 to cooperatively localize the targets while minimizing the adopted global cost. The chapter also includes the plots of the chosen metrics and the localization plot indicating the robots' positions and the targets' true and estimated positions.

- **Chapter 3 - Aggregative Optimization for Multi-Robot Systems**
  In this chapter, complementing the theoretical notions outlined in Chapter 1, the use of the cost function and the AT method is explained in the context of the aggregative optimization for our multi-robot and multi-target system (Task 2.1). The chapter presents the necessary plots and an animated visualization of the robots' motion and their barycenter. Lastly, the chapter describes and provides the results for the ROS 2 implementation of Task 2.1.

- **Conclusions**
  This chapter summarizes the results of the tasks and outlines the issues faced during the project development.

Contributions

The project consisting of the code for all the tasks and the report has been developed entirely by myself. During the project development, for innumerable times I referred to the theoretical concepts explained throughout the course and presented in the class materials. Specifically, for the code part, the exercise code files and the templates developed and discussed in the class have been extremely helpful to complete the project.

Chapter 1

Distributed Consensus Optimization

This chapter covers the solution of the distributed consensus optimization problem with the quadratic cost and results of its implementation in Python as part of Task 1.1. It is considered a preliminary task, in which we introduce the Gradient Tracking (GT) algorithm later used in Task 1.2 and described in Chapter 2.

1.1 Implementation

This section sheds a light on the problem setup and some important theoretical concepts used to solve the first task of the project.

1.1.1 Problem Setup

We start by introducing our problem setup, which comprises solving the consensus optimization problem written in the general form as in Equation 1.1 [1].

$$\min_{z} \sum_{i=1}^{N} \ell_i(z), \tag{1.1}$$

where $z \in \mathbb{R}^d, d > 1$, is the decision vector and $\ell_i : \mathbb{R}^d \to \mathbb{R}$, is the local quadratic cost function for $i = 1, \ldots, N$. The local quadratic cost function of instance i and its gradient have been implemented in the code and described by Equation 1.2 and 1.3, respectively [1].

$$\ell_i = \frac{1}{2} z^T Q_i z + r_i^T z, \quad \forall i = 1, \ldots, N \tag{1.2}$$

$$\nabla_{z_i} \ell_i = Q_i z + r_i, \quad \forall i = 1, \ldots, N \tag{1.3}$$

where $Q \in \mathbb{R}^{d \times d}, Q \succ 0, Q = Q^T$, is a positive-definite and symmetric matrix, whose values range from 0.2 to 1.0 selected by uniform probability distribution in our code;

and $r \in \mathbb{R}^d$ is a vector, whose values are selected by the normal probability distribution within our code. By applying the second-order sufficient condition (SSC), we can deduce that $z_i^\star$ is the global minimizer of ℓ_i and is described by Equation 1.4 [1].

$$z_i^\star = -Q_i^{-1} r_i, \quad \forall i = 1, \ldots, N \tag{1.4}$$

The total global minimizer of the minimization problem in Equation 1.1 can be found directly by Equation 1.5 [1].

$$z^\star = -Q^{-1} r, \tag{1.5}$$

where $Q = \sum_{i=1}^N Q_i$ and $r = \sum_{i=1}^N r_i$. Later, we evaluate the total optimal cost at this value and compare it with the result derived from solving the same consensus optimization problem using the Gradient Tracking (GT) algorithm.

1.1.2 Adjacency Matrix Computation

In the distributed optimization problems, the interaction between the nodes is represented by a graph $G = (I, E)$, where $I = 1, \ldots, N$ is the set of our instances - later agents or robots - called nodes, and $E \subset I \times I$ is a set of ordered node pairs called edges [1]. In our project, the graph G is undirected, meaning that for each pair of nodes $(i, j) \in E$, there is also $(j, i) \in E$, which essentially means that every node is both a communicating and a sensing node.

As we know from the theory, in the distributed setting, each node i has its local private data (z_i, ℓ_i) and shares it only with its direct neighbors within the graph G [1]. In fact, the graph G is rather a conceptual formation that sets some rules for the interaction between the nodes, whereas the adjacency matrix A is a numerical representation whose entries represent the existence of an edge between specific nodes, such that $a_{ij} = 1$ if $(i, j) \in E$, and $a_{ij} = 0$ otherwise [1]. Likewise, for a weighted adjacency matrix also denoted as A, its entries are the non-negative weights placed on the edges, such that $a_{ij} > 0$ if $(i, j) \in E$, and $a_{ij} = 0$ otherwise [1]. This matrix plays a pivotal role in the "mixing" of the nodes' states step of the GT algorithm, which contributes to achieving some consensus on all N decision vectors.

In our project, the entries of the weighted adjacency matrix A are computed via the Metropolis-Hastings weighting method described by Equation 1.6 [1].

$$a_{ij} = \begin{cases} \frac{1}{1+\max\{d_i, d_j\}} & \text{if } (i,j) \in E \text{ and } i \neq j \\ 1 - \sum\limits_{h \in \mathcal{N}_i \setminus \{i\}} a_{ih} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

where $\mathcal{N}_i = \{j \in 1, \ldots, N | (i, j) \in E\}$, is the set of neighbors of the node i, and $d_i = |N_i|$, is the degree of i, namely the number of neighbors it has.

By using this method, we ensure that the matrix A is symmetric and doubly-stochastic, where the latter definition means that the sum of rows and the sum of columns both equal 1 [1].

1.1.3 Gradient Tracking Algorithm

The idea of the Gradient Tracking (GT) algorithm is the same as for other local gradient-based update rules, which aim at efficiently updating the local gradients to guide local optimal solutions to a global one in a distributed setting. The GT method is presented completely in Equation 1.7 [1].

$$\begin{aligned} z_i^{k+1} &= \sum_{j \in N_i} a_{ij} z_j^k - \alpha s_i^k & z_i^0 &\in \mathbb{R} \\ s_i^{k+1} &= \sum_{j \in N_i} a_{ij} s_j^k + \nabla \ell_i(z_i^{k+1}) - \nabla \ell_i(z_i^k) & s_i^0 &= \nabla \ell_i(z_i^0) \end{aligned} \tag{1.7}$$

where $\alpha > 0$ is a constant step size. The local estimate s_i^k at iteration k is the gradient tracking variable, which converges to the mean of all the local gradients as k approaches infinity [1]:

$$s_i^k \underset{k \to \infty}{\longrightarrow} \frac{1}{N} \sum_{h=1}^{N} \nabla \ell_h(z_h^k)$$

As we see from Equation 1.7, the update of the estimate s_i consists of two parts: the first one is the consensus part, where each local estimate mixes with its neighbors based on the weights provided by matrix A; the second one is local innovation part based on the difference between the local gradients of the subsequent iterations. In fact, perfect gradient tracking is achieved by s_i, when $\nabla \ell_i(z_i^k)$ converges to a constant value as k approaches infinity [1]. We also choose our descent direction at the iteration k to be equal to the negative of our gradient tracking variable at the same iteration, namely, $d_i^k = -s_i^k$ [1]. The described approach has been implemented in the code using the quadratic cost stated earlier as our local cost function with the initial condition z_i^0 selected by the normal probability distribution with zero mean and standard deviation equal to 0.5.

To understand how the convergence of local gradients to a constant value is related to convergence of z_i^k to the consensual global solution across all the nodes, we need to mention the theorem about the convergence result of the GT algorithm itself. It is based on two main assumptions, which we satisfied by making our weighted adjacency matrix A, connected to the undirected and connected graph G with self-loops, doubly-stochastic, and by ensuring that each cost function has quadratic lower and upper bounds due to its strong convexity and Lipschitz continuity of the gradient [1]. From that we can state that there exist an optimal step size α^* such that that for all $\alpha \in (0, \alpha^*)$, the following condition holds [1]:

$$\lim_{k \to \infty} \left\| z_i^k - z^* \right\| = 0$$

where z_i^k is the local solution estimate generated by the GT algorithm at iteration k, and z^* is the global optimal solution to the distributed consensus problem, which all

the nodes agreed upon. Moreover, we expect the convergence rate to be linear, and the global optimal solution to be exponentially stable.

1.2 Results

This section presents the results of the GT algorithm launched with different parameters in the form of plots of the evolution of the total cost and norm of its gradient along the iterations.

1.2.1 Choice of Parameters

To test the efficiency of the GT approach, we use four different graph patterns, which include the *path*, *ring*, *star*, and *Erdős–Rényi (ER)* topologies, with the probability of an edge equal to 0.35 ($p = 0.35$) for the *ER* graph. The number of nodes N has been set to 10, 20 and 30, with the dimension d of the decision vector z equal only to 4 and 8. The maximum number of iterations for the GT algorithm to run has been chosen to be 2000, with the stopping condition that terminates the GT update rule once the norm of the gradient reaches the value of 10^{-6}. Lastly, the value of the step size α has been to 0.05 ($\alpha = 0.05$).

1.2.2 Metrics' Plots

By having the ability to compute the global optimal solution z^* directly using Equation 1.5, we decided that for the evolution of the cost function plot, we consider the absolute deviation of our total cost at z^k generated by GT from the total optimal cost $\ell(z^*)$. Figure 1.1 shows the deviation of the total cost and the evolution of the gradient norm of the total cost across iterations for four graph patterns with $N = 10$ and $d = 4$. Figure 1.2 shows the deviation of the total cost and the evolution of the gradient norm of the total cost across iterations for four graph patterns with $N = 20$ and $d = 4$. Figure 1.3 shows the deviation of the total cost and the evolution of the gradient norm of the total cost across iterations for four graph patterns with $N = 20$ and $d = 8$. Lastly, Figure 1.4 shows the deviation of the total cost and the evolution of the gradient norm of the total cost across iterations for four graph patterns with $N = 30$ and $d = 8$.

For all four cases and all four graph patterns, we see either an entirely linear or linear in some parts type of behavior, where both metrics steadily decrease. We can observe, that the *ER* and *star* graphs always converge under 1500 iterations, whereas the *cycle* graph tends to converge slower once the number of nodes and the dimensions increase. On the contrary, the *path* graph never converges to the tolerance value within the allocated iterations, which is understandable due to the specific connections of its nodes. It is also worth noticing that the metrics' plots for different graph topologies become more spread out when N and d are increased.

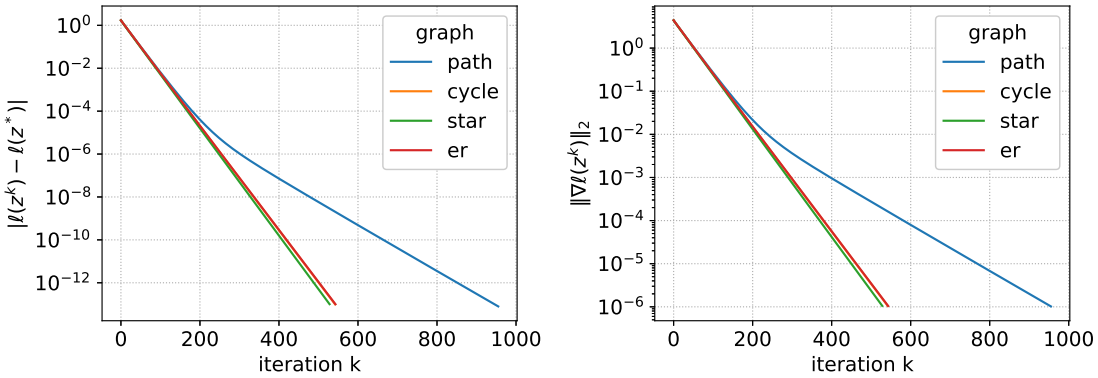

Figure 1.1: Deviation of the total cost (on the left) and the evolution of the gradient norm of the total cost (on the right) across iterations for four graph patterns with $N = 10$ and $d = 4$

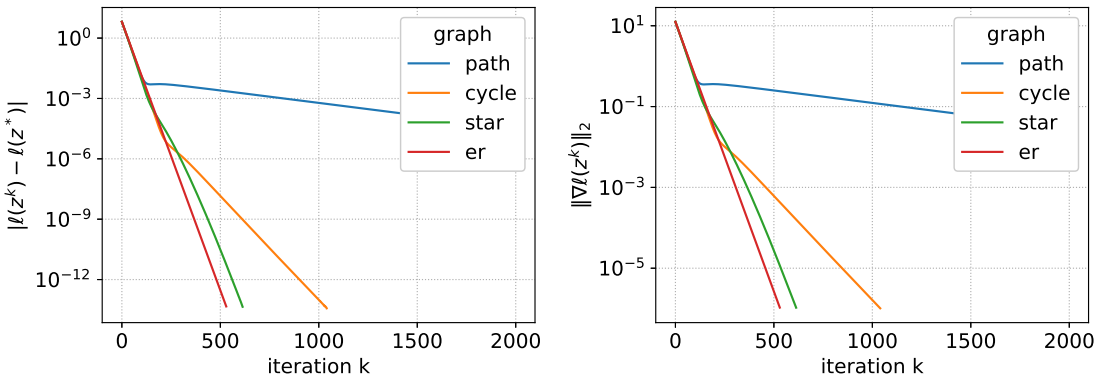

Figure 1.2: Deviation of the total cost (on the left) and the evolution of the gradient norm of the total cost (on the right) across iterations for four graph patterns with $N = 20$ and $d = 4$

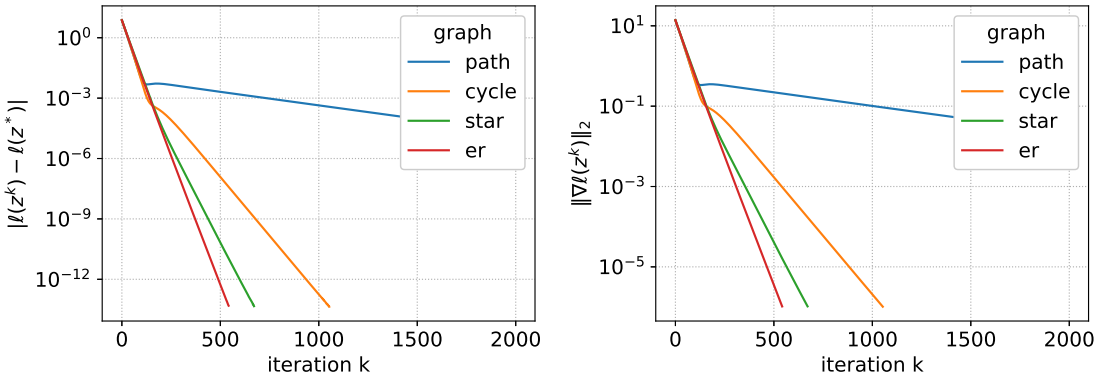

Figure 1.3: Deviation of the total cost (on the left) and the evolution of the gradient norm of the total cost (on the right) across iterations for four graph patterns with $N = 20$ and $d = 8$

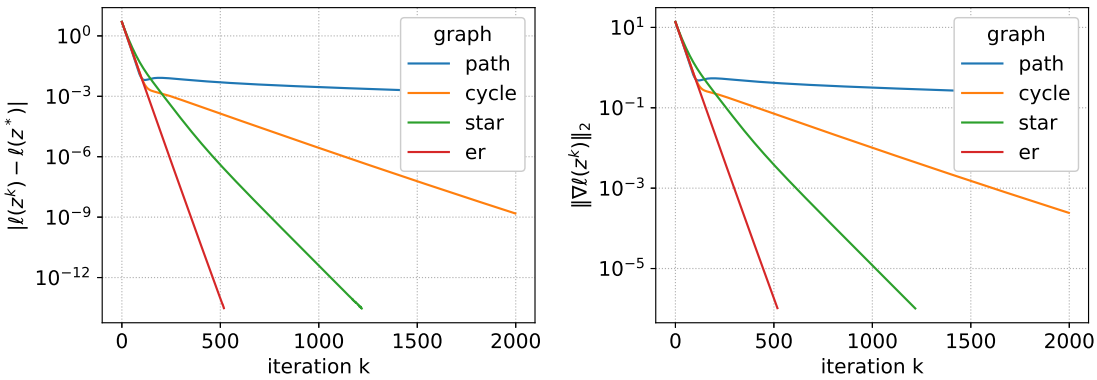

Figure 1.4: Deviation of the total cost (on the left) and the evolution of the gradient norm of the total cost (on the right) across iterations for four graph patterns with $N = 30$ and $d = 8$

Chapter 2

Cooperative Multi-Robot Target Localization

In this chapter, we use the theoretical tools described in Chapter 1 to implement a mechanism that enables a team of robots cooperatively localize the targets as part of Task 1.2. To assess the effectiveness of the framework, we plot the dedicated metrics and a localization plot with true and estimated targets' positions.

2.1 Implementation

In this section, we describe the problem setup, which is in this case pertained to a real-life scenario and not an abstract optimization problem. We employ the same adjacency matrix and Gradient Tracking (GT) algorithm defined when solving Task 1.1 in Chapter 1.

2.1.1 Problem Setup

We start by introducing our problem setup, which is visually presented in Figure 2.1. Each red vehicle $i \in \{1, \dots, N\}$ positioned at $p_i \in \mathbb{R}^d$ cooperatively with other red vehicles tries to estimate the positions of black, blue, and green vehicles (targets) based on the noisy measurements of distance $d_{i\tau} \in \mathbb{R}_{\geq}0$ between the robot (red vehicle) and the target $\tau \in \{1, \dots, N_T\}$.

As part of the task, we generate $N \in \mathbb{N}$ arbitrary positions of the robots and $N_T \in \mathbb{N}$ arbitrary positions of the targets selected by the uniform probability distribution between 0 and the value of the *field*, where the localization takes place (in our case *field* = 10). To provide the noisy distance measurements to the robots of every targets' positions, we add some zero-mean Gaussian noise with a controlled by us standard deviation to the true distance between target τ and robot i, and pick a maximum value between the resulted value and 0 to ensure non-negativity.

For this task, we adopt local cost functions for all $i \in \{1, \dots, N\}$ defined by Equation 2.1 [1].

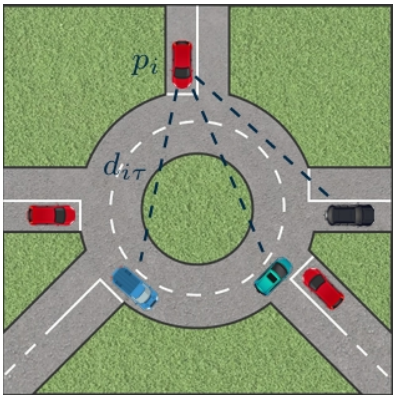

Figure 2.1: A multi-target localization problem with black, blue, and green vehicles being a pool of targets, and red vehicles being a team of robots [1]

$$\ell_i(z) := \sum_{\tau=1}^{N_T} \left(d_{i\tau}^2 - \|z_\tau - p_i\|^2 \right)^2 \tag{2.1}$$

where $z = \text{col}(z_1, \ldots, z_{N_T}) \in \mathbb{R}^{dN_T}$ is the optimization variable, and $z_\tau \in \mathbb{R}^d$ is the the τ^{th} block component of z and also the estimated target position, which the robots should agree upon. For the GT update, we need the local gradients of the cost functions for all $i \in \{1, \ldots, N\}$ described by Equation 2.2.

$$\nabla_{z_\tau} \ell_i(z) = 4 \left(\|z_\tau - p_i\|^2 - d_{i\tau}^2 \right) (z_\tau - p_i) \tag{2.2}$$

As was mentioned in Chapter 1, the total cost and the total gradient is the sum of all N local cost functions and gradients, correspondingly. Like in Task 1.1, in Task 1.2, we also want to track our local gradients, which eventually must converge to a constant value. The GT algorithm described in Equation 1.7 ensures that we reach a stable global optimal solution, which all the robots agree upon.

2.2 Results

In this section, we present the results of the convergence of the GT algorithm in the form of plots of the evolution of the total cost and norm of its gradient along the iterations, as well as the localization graph with targets' true and estimated positions.

2.2.1 Choice of Parameters

The choice of a graph pattern for the target localization task has been restricted to the *Erdős–Rényi (ER)* topology with $p = 0.35$, as no significant difference between the metrics results obtained for other graph patterns was observed. The number of robots N and targets N_T is set to 10, whereas the dimension d of the target position z_τ is set to

2. The maximum number of iterations for the GT algorithm to run has been set to 2000. The value of the step size α is 10^{-4}, as the larger step sizes does not make the algorithm converge effectively. Lastly, the standard deviations (std) values of the Additive White Gaussian Noise (AWGN) added to the distance measurements are equal to 0, 0.02, 0.1, and 0.2 for four different plots.

2.2.2 Metrics' Plots and Simulations

We plot the evolution of the cost and gradient norm across 2000 iterations along with the localization graph for four different noise std values. Figures 2.2 and 2.3 demonstrate the evolution of the aforementioned metrics for the *ER* graph and the localization plot with true and estimated targets' positions without the noise being added, respectively. Figures 2.4 and 2.5 demonstrate the evolution of the aforementioned metrics for the *ER* graph and the localization plot with true and estimated targets' positions with the noise std equal to 0.02, respectively. Figures 2.6 and 2.7 demonstrate the evolution of the aforementioned metrics for the *ER* graph and the localization plot with true and estimated targets' positions with the noise std equal to 0.1, respectively. Figures 2.8 and 2.9 demonstrate the evolution of the aforementioned metrics for the *ER* graph and the localization plot with true and estimated targets' positions with the noise std equal to 0.2, respectively.

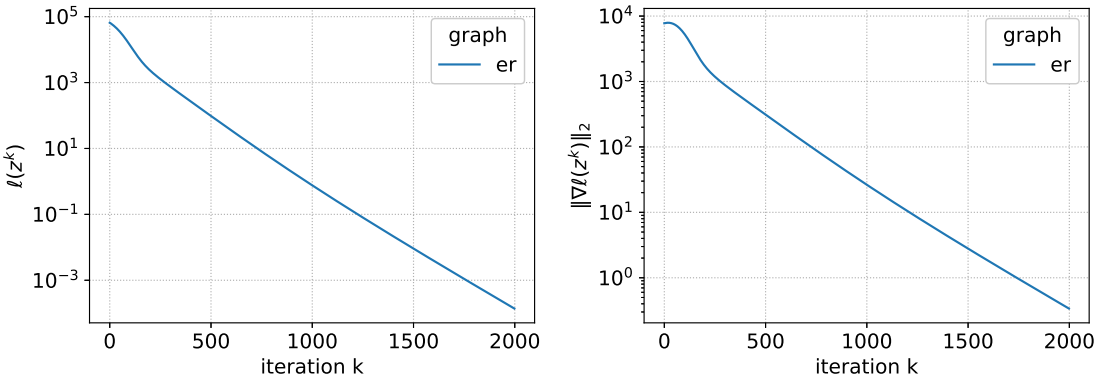

Figure 2.2: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *ER* graph pattern with no noise added

From Figures 2.2 and 2.3, we can clearly observe a faster and more prominent decrease in the metrics and more precise localization of the targets by the robots in comparison with other cases. With increase of the noise std, we see a clear pattern, where the cost converges to a greater value and takes less iterations to do it, while the gradient norm converges to the same or almost the same constant value regardless of the difference in the noise. However, the localization still tends to deteriorate as the noise std increases, which we can observe in Figure 2.9, where some targets' estimated positions are clearly more off in comparison with Figures 2.5 and 2.7.

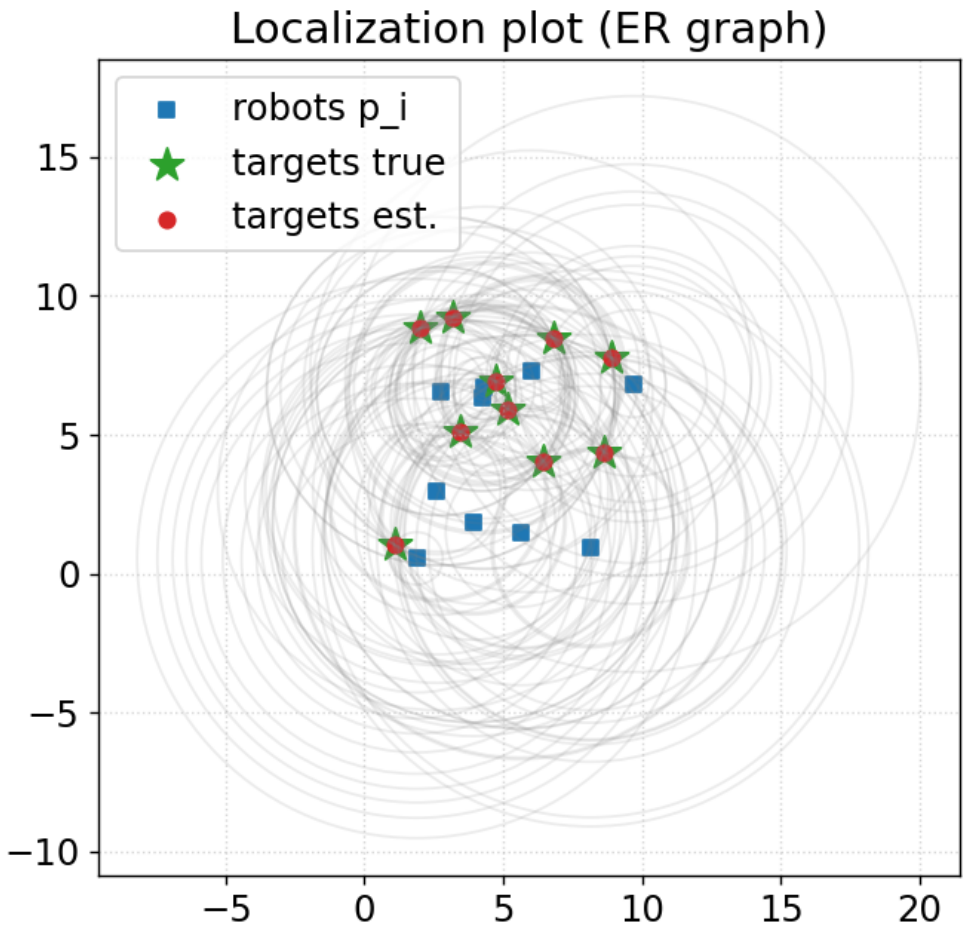

Figure 2.3: Target localization plot when no noise added

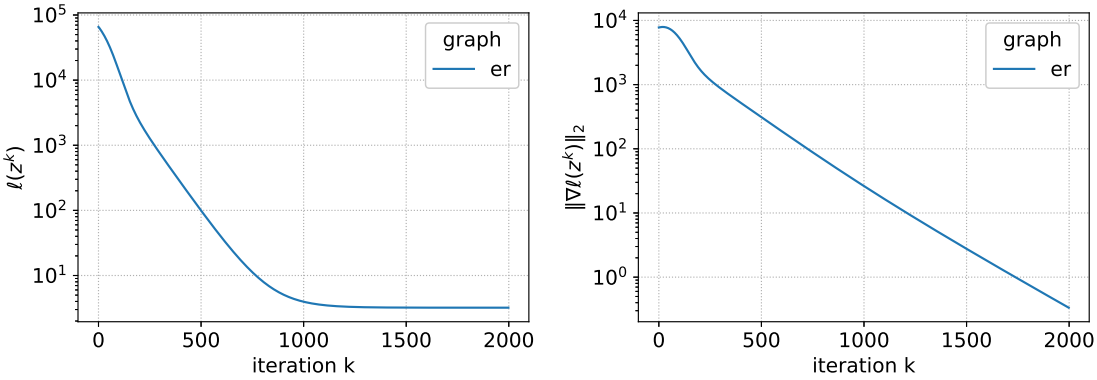

Figure 2.4: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *ER* graph pattern with AWGN's std equal to 0.02

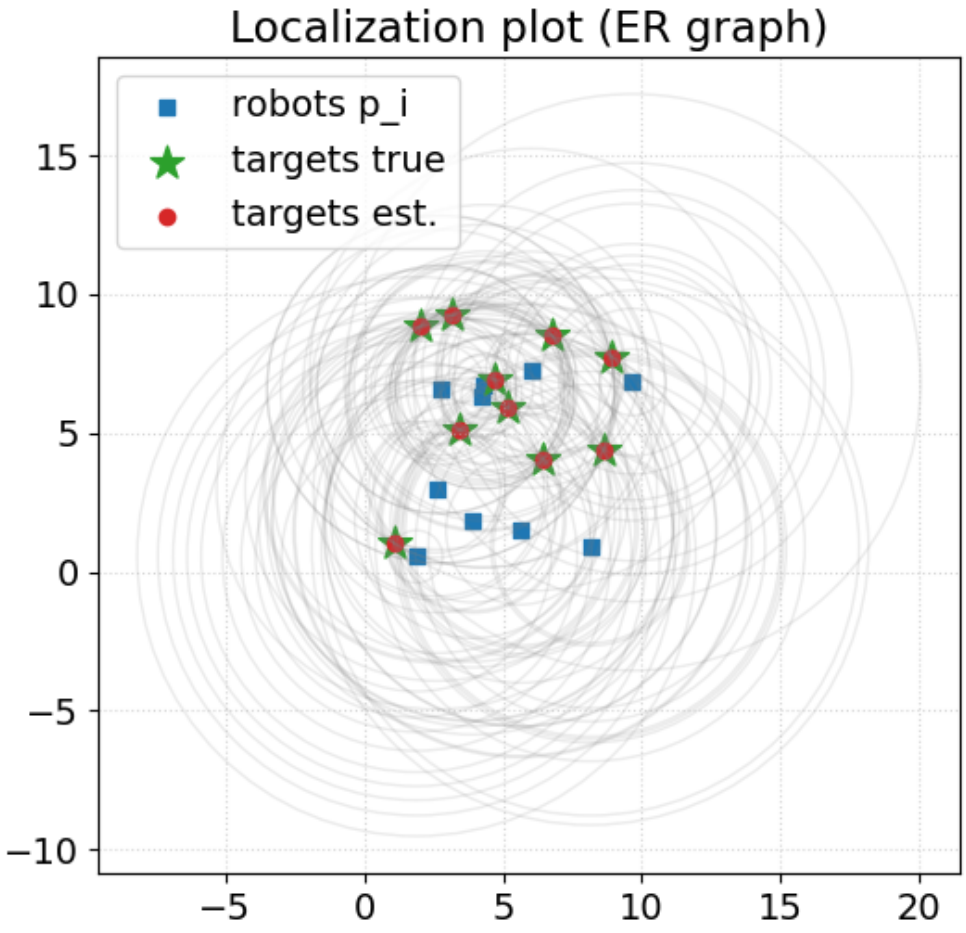

Figure 2.5: Target localization plot when AWGN's std equal to 0.02

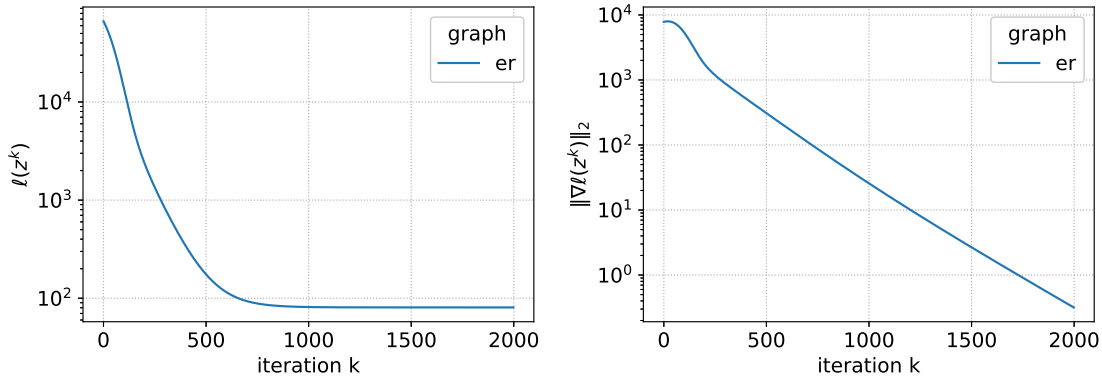

Figure 2.6: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *ER* graph pattern with AWGN's std equal to 0.1

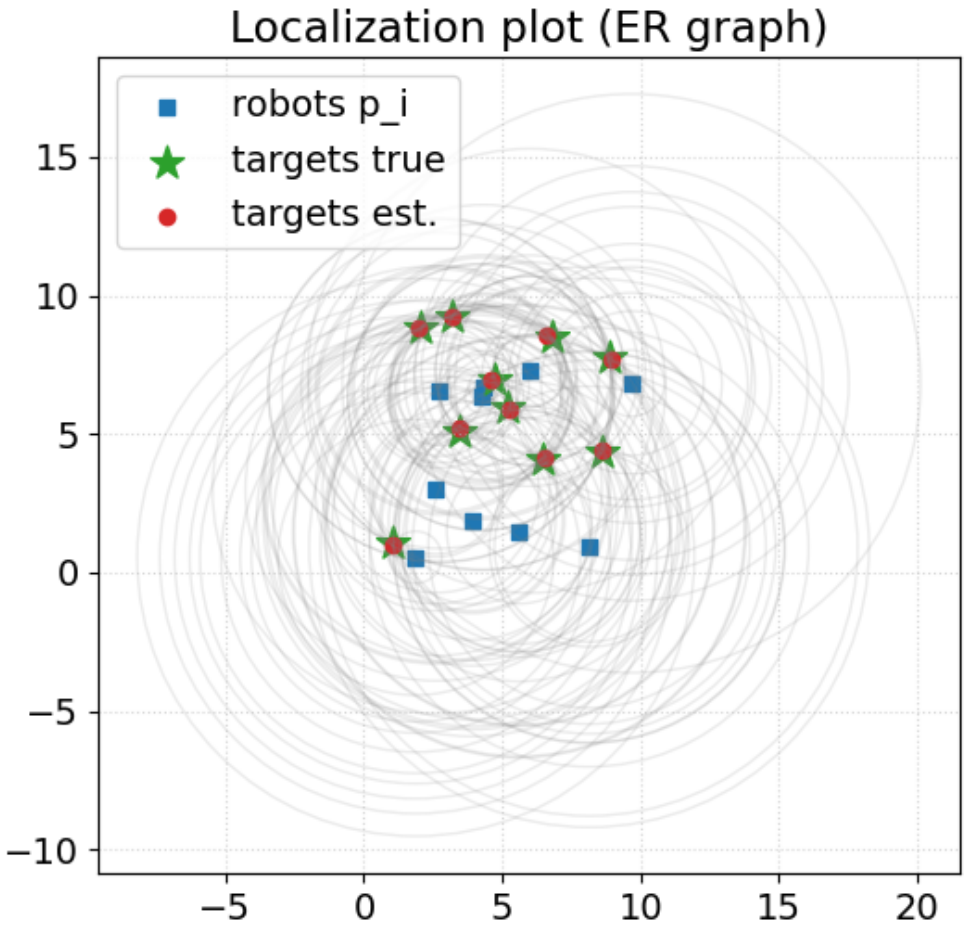

Figure 2.7: Target localization plot when AWGN's std equal to 0.1

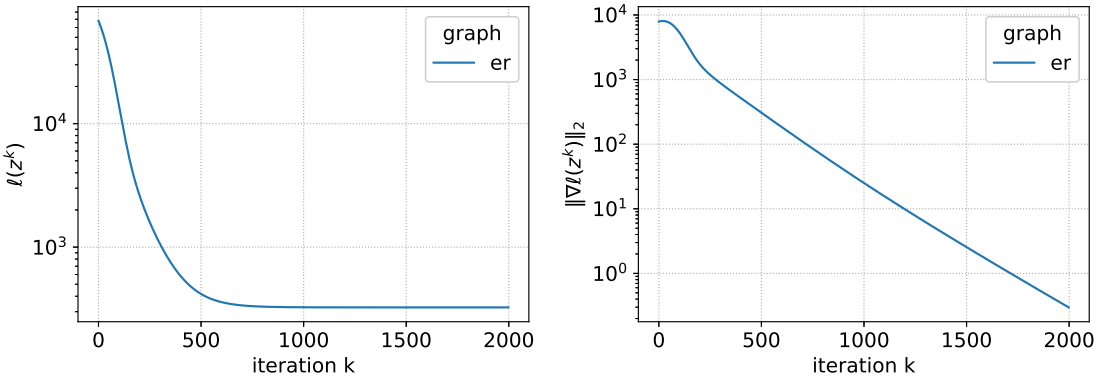

Figure 2.8: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *ER* graph pattern with AWGN's std equal to 0.2

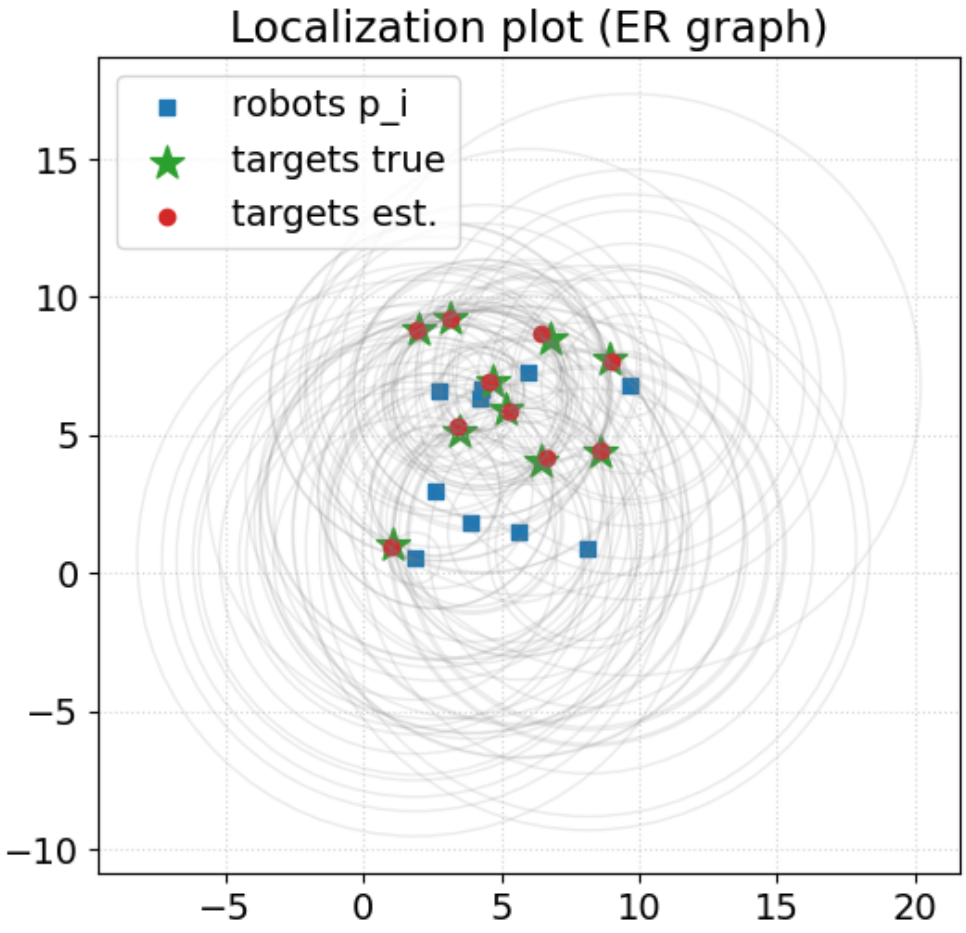

Figure 2.9: Target localization plot when AWGN's std equal to 0.2

Chapter 3

Aggregative Optimization for Multi-Robot Systems

This chapter discusses the implementation of a distributed control algorithm based on the Aggregative Tracking (AT) approach that establishes robots' behavior based on two criteria: moving towards their own private targets and keeping the fleet tight simultaneously. In the end of the chapter, we present necessary plots of our metrics along with a comprehensive visualization of the discussed robots' motions. Lastly, we share the plots and simulations retrieved from implementing the framework in ROS 2 environment.

3.1 Implementation

This section presents the problem setup of Task 2.1 and introduces the Aggregative Tracking (AT) algorithm employed to solve our aggregative optimization problem.

3.1.1 Problem Setup

The problem is visually presented in Figure 3.1. We consider a team of N robots, where the position of the robot $i \in \{1, \dots, N\}$ at the iteration k is defined as $z_i^k \in \mathbb{R}^d$, where d is set to 2. The goal is to update the robots' positions in the way that will make them steadily approaching their private targets, but also satisfying the tightness constraint that imposes the robots to stay close to the team. This can be generalized to the aggregative optimization problem described in Equation 3.1 [1].

$$\min_{z \in \mathbb{R}^{dN}} \sum_{i=1}^{N} \ell_i(z_i, \sigma(z)), \quad \text{with} \quad \sigma(z) := \frac{1}{N} \sum_{i=1}^{N} \phi_i(z_i) \tag{3.1}$$

where $\ell_i : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $\phi_i : \mathbb{R}^d \to \mathbb{R}^d$. This aggregative variable $\sigma(z)$ is considered a barycenter of the team in our project, therefore $\phi_i(z_i) = z_i$ for all $i \in \{1, \dots, N\}$. To satisfy the goal of the task, the local cost functions and local gradients wrt z_i have the formula as shown in Equation 3.2 for all $i \in \{1, \dots, N\}$ [1].

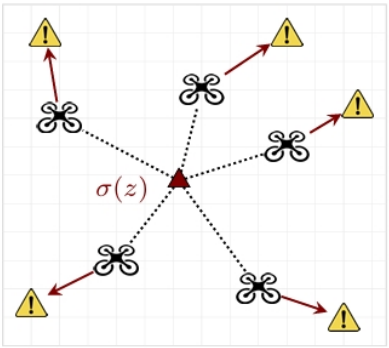

Figure 3.1: An aggregative optimization task, where each robot wants to move towards its fixed private target (in yellow), while preserving the tightness constraint of the fleet [1]

$$\begin{aligned} \ell_i(z_i, \sigma(z)) &= \frac{1}{2}a\|z_i - p_i\|^2 + \frac{1}{2}b\|z_i - \sigma(z)\|^2 \\ \nabla_1 \ell_i(z_i, \sigma(z)) &= a(z_i - p_i) + b(z_i - \sigma(z)) \end{aligned} \tag{3.2}$$

where $p_i \in \mathbb{R}^d$ is the fixed position of target i, and a, b are the trade-off parameters. To find the gradient of the total cost, which will be useful for plotting, we need refer to Equation 3.3 [1].

$$\nabla \left(\sum_{j=1}^{N} \ell_j \left(z_j, \sigma(z) \right) \right) = \nabla_1 \ell_i(z_i, \sigma) + \frac{1}{N} \nabla_1 \phi_i(z_i) \sum_{j=1}^{N} \nabla_2 \ell_j(z_j, \sigma), \quad \forall i = 1, \ldots, N \tag{3.3}$$

For our project, the total gradient simplifies to the following form [1]:

$$\nabla \left(\sum_{j=1}^{N} \ell_j \left(z_j, \sigma(z) \right) \right) = a(z_i - p_i) + b(z_i - \sigma(z)) - \frac{b}{N} \sum_{j=1}^{N} (z_j - \sigma(z)), \quad \forall i = 1, \ldots, N$$

3.1.2 Aggregative Tracking Algorithm

The idea of the AT algorithm is similar to that of the GT algorithm, but instead of tracking the gradients, the AT algorithm aims at tracking the aggregative variable $\sigma(z)$, which is the barycenter of robots' team in our case. The AT algorithm is described by Equation 3.4 [1].

$$\begin{aligned} z_i^{k+1} &= z_i^k - \alpha \nabla_1 \ell_i(z_i^k, s_i^k) & z_i^0 &\in \mathbb{R}^d \\ s_i^{k+1} &= \sum_{j \in N_i} a_{ij} s_j^k + z_i^{k+1} - z_i^k & s_i^0 &= z_i^0 \end{aligned} \tag{3.4}$$

In this case, s_i is a dynamic barycenter tracking variable possessed by the robot i, and v_i, which maintains the sum of the gradients of ℓ_i wrt $\sigma(z)$, is zero in our project. The

initial position z^0 of the robots has been selected by the uniform probability distribution between 0 and the value of the *field*, which is 10. The principal difference between the AT and GT algorithms lies in the way how AT updates the robots' positions: unlike GT, in AT, a particular robot does not mix its state with the neighbors, but preserves it from iteration k and updates it using the local gradient of the cost function computed at the state and barycenter estimates of current iteration.

3.2 Results

This section provides the plots for the evolution of the cost and the gradient norm obtained by the means of the AT method.

3.2.1 Choice of Parameters

The choice of a graph pattern for the aggregative optimization task has been restricted to the *Erdős–Rényi (ER)* topology with $p = 0.35$, as no clear difference between the metrics results obtained for other graph patterns was spotted. The number of robots N is chosen to be 10, and the maximum number of AT iterations is set to 2000. The number of AT iterations might be less if the norm of the gradient reaches the tolerance value that is set to 10^-6. The step size α is set to 0.01. Lastly, different plots and simulations have been prepared for different sets of trade-off parameters (a,b): those include (1,1), (2,1), and (1,0).

3.2.2 Metrics' Plots and Simulations

Along with well-known plots of the metrics, we provide the plot of the animated behavior frozen at the last iteration, which shows the positions of the targets, robots, and their barycenter. Figures 3.2, 3.4, and 3.6 demonstrate the evolution of our metrics for the *ER* graph for the set of the trade-off parameters (a,b) equal to (1,1), (2,1), (1,0), correspondingly. Figures 3.3, 3.5, and 3.7 depict the animated behavior of the robots' team observed at the last iteration for the set of the trade-off parameters (a,b) equal to (1,1), (2,1), (1,0), correspondingly.

From the figures, we can see that when increasing the trade-off parameter a related to the closeness to the private target from 1 to 2 (see Figures 3.3 and 3.5), the robots tend to appear closer to their targets and do it in less iterations almost by a half. Additionally, we can observe that when the trade-off parameter b related to the tightness constraint is 0 (see Figure 3.7, the robots stay exactly on their private targets. Worth mentioning that in all the cases the barycenter has the same coordinates as it is just the mean of the robots' positions.

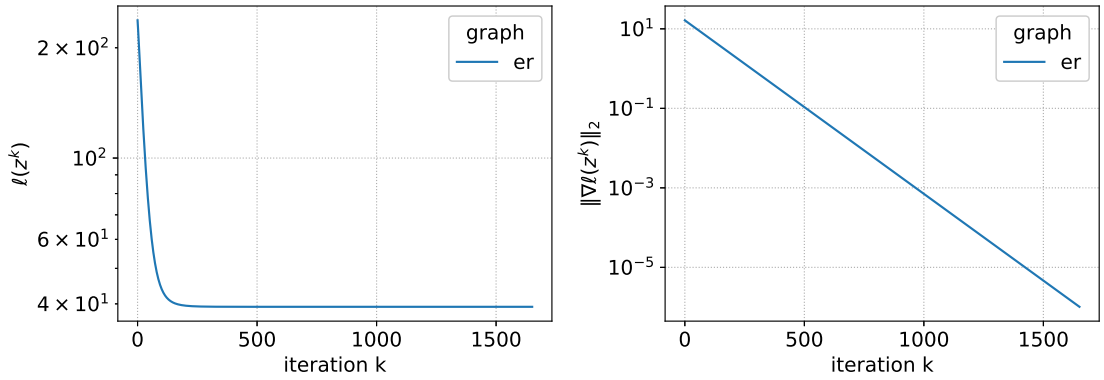

Figure 3.2: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *ER* graph pattern with the trade-off parameters $a = 1$ and $b = 1$

3.3 Results for ROS 2 Implementation

This section concerns the implementation of Task 2.2 of the project. The package containing the file that spawns N robots and updates their positions using the AT algorithm, the file that visualizes the team behavior using RViz, the file that saves the metrics to a csv file, and also the file launches all the nodes, with all the publishing and subscribing happening in-between, has been created and run in ROS 2 environment [1] [2]. Some additional Python files has been implemented to read the metrics from the csv file and plot them as shown in Figures 3.8, 3.10, and 3.12. The results of the RViz animation of the team behavior for the trade-off parameter sets from Task 2.1 are shown in Figures 3.9, 3.11, and 3.13. All those figures imply the communication happening through the *star* graph topology, with the AT step size α of 0.05.

Recalling that the metrics results and robots' team behavior are almost same for different graph patterns, we indeed observe that the metrics plots for the *star* graph in Task 2.2 correlate strongly with the results for the *ER* graph in Task 2.1. From the RViz visualizations, we can observe how the change of the trade-off parameters affects the localization of the robots: when a is changed from 1 to 2, the robots become more spread, assuming that they moved closer to their targets (see Figures 3.9 and 3.11); when b is set to 0, the robots become even more spread, which implies that robots seat exactly on their targets (see Figure 3.13.

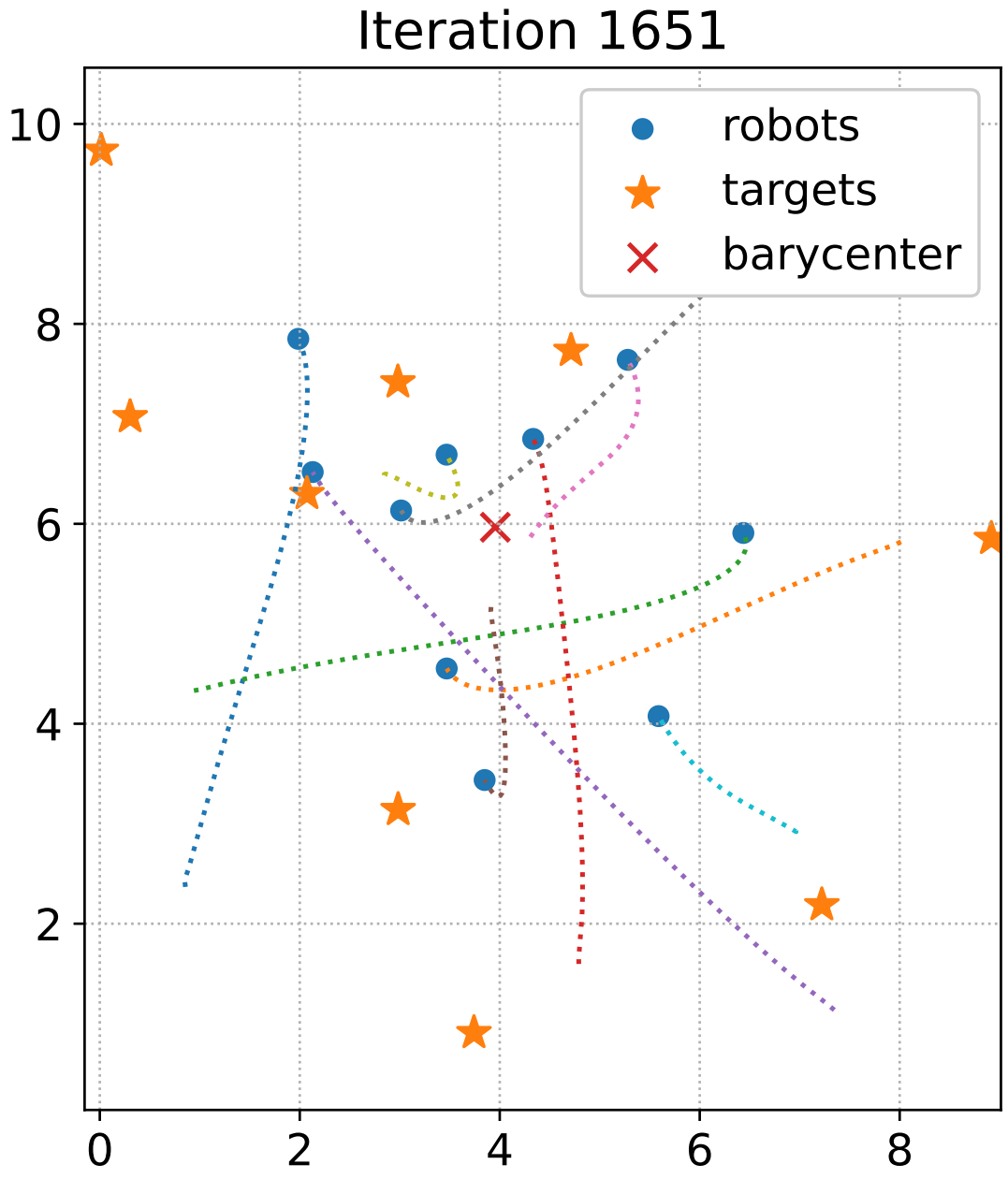

Figure 3.3: The plot of the aggregative optimization scenario at the last iteration with the trade-off parameters $a = 1$ and $b = 1$

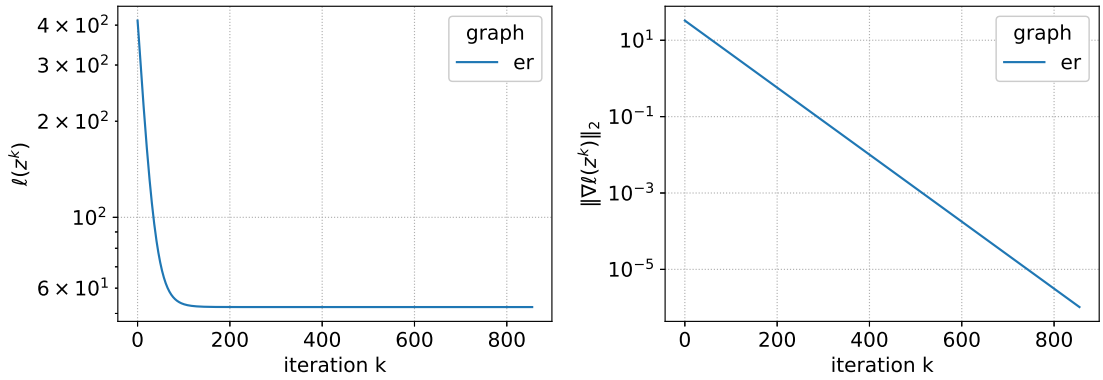

Figure 3.4: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *ER* graph pattern with the trade-off parameters $a = 2$ and $b = 1$

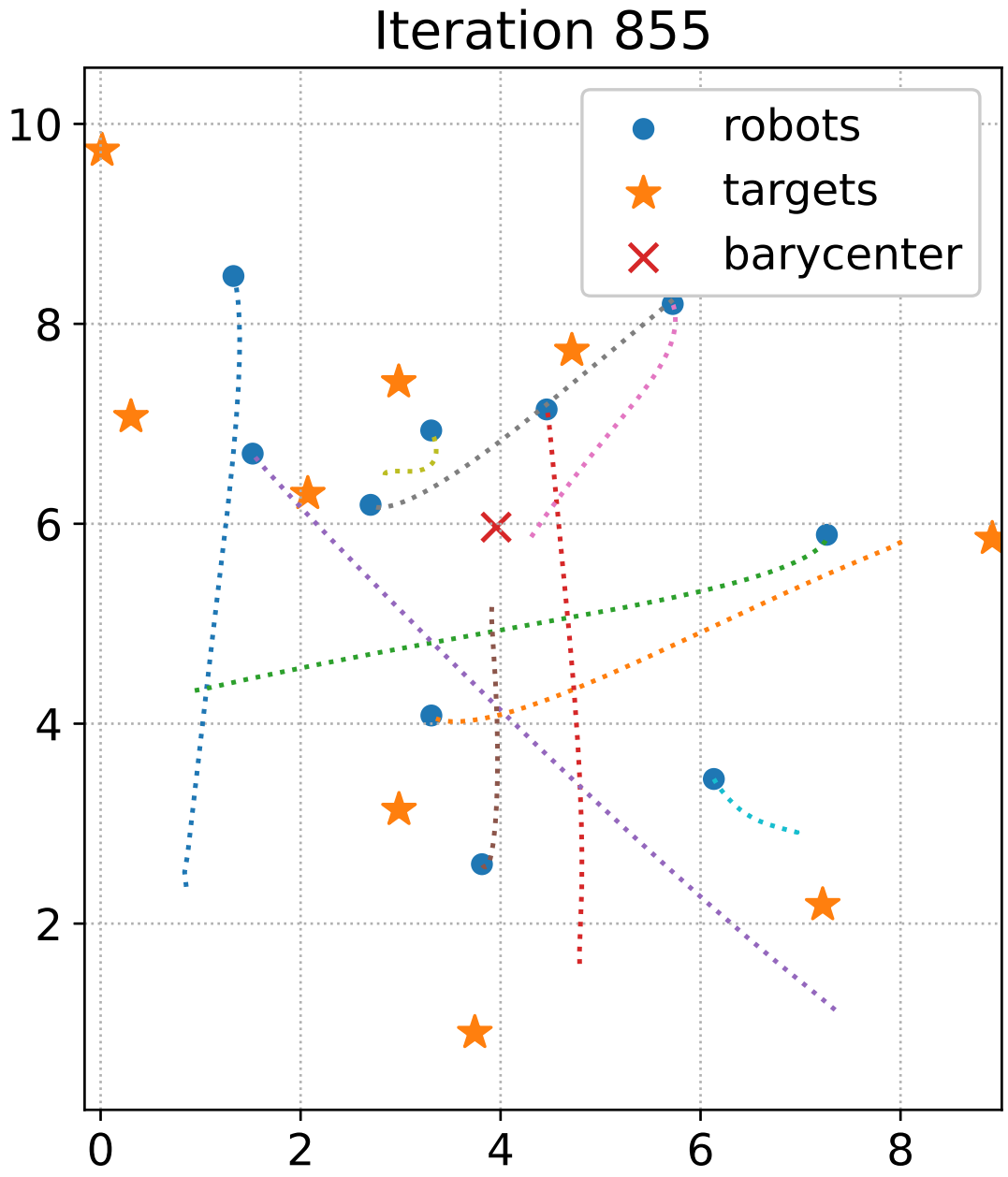

Figure 3.5: The plot of the aggregative optimization scenario at the last iteration with the trade-off parameters $a = 2$ and $b = 1$

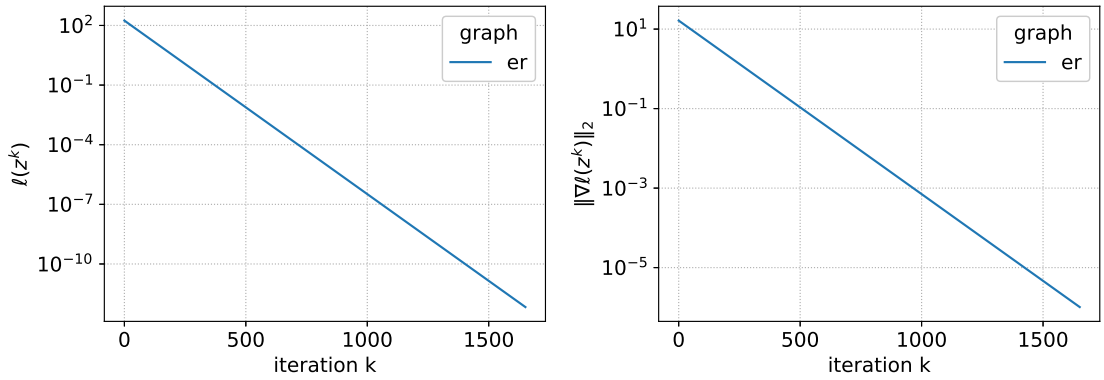

Figure 3.6: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *ER* graph pattern with the trade-off parameters $a = 1$ and $b = 0$

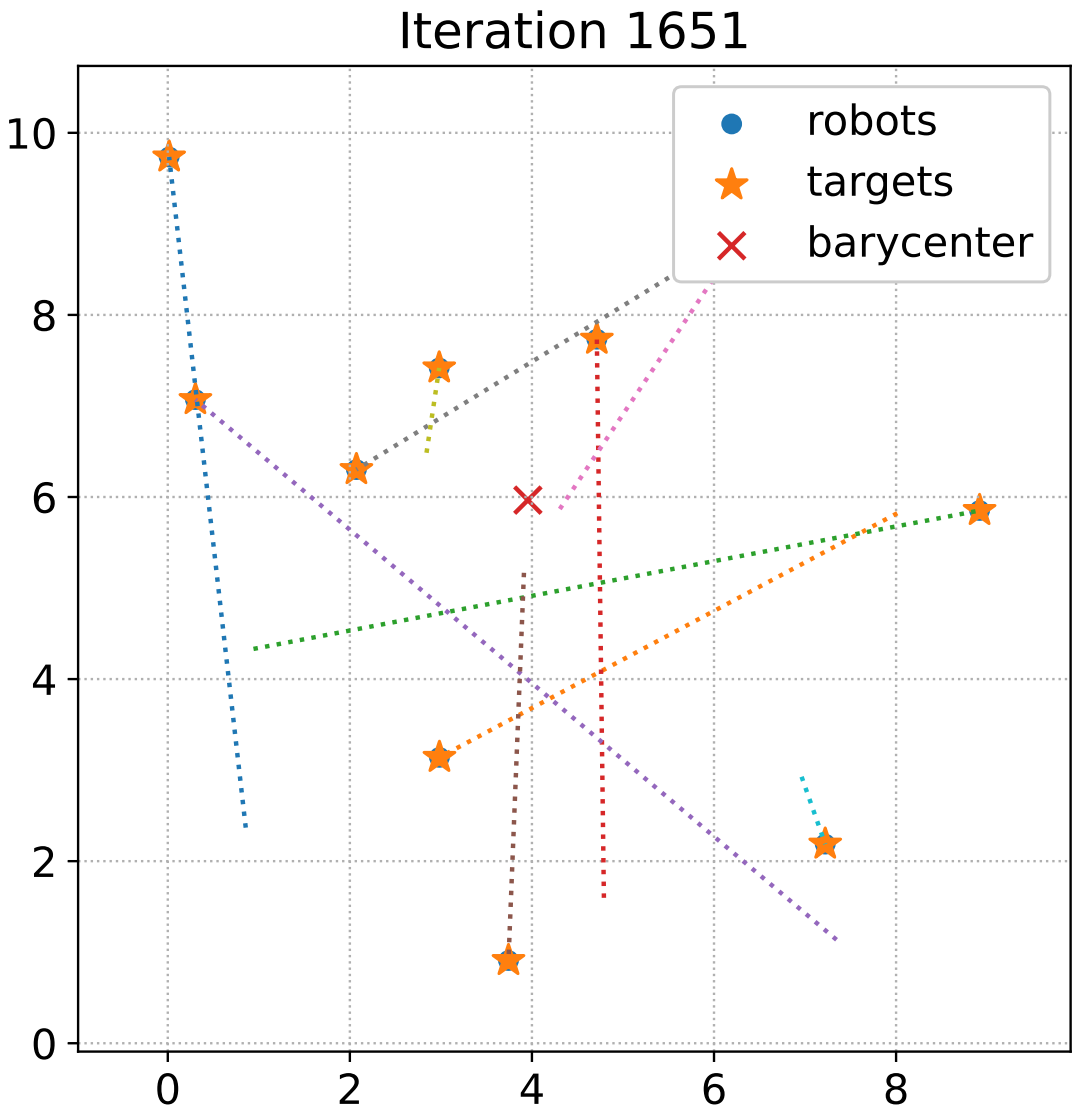

Figure 3.7: The plot of the aggregative optimization scenario at the last iteration with the trade-off parameters $a = 1$ and $b = 0$

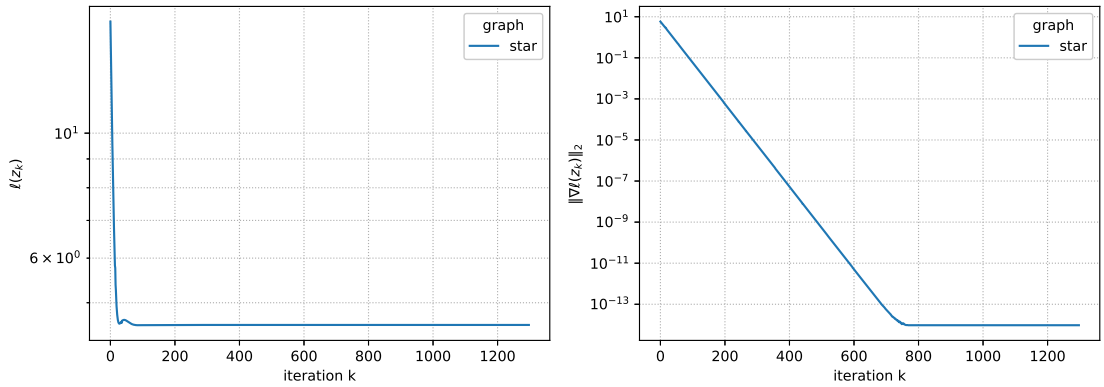

Figure 3.8: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *star* graph pattern with the trade-off parameters $a = 1$ and $b = 1$ retrieved from ROS 2

Figure 3.9: The RViz visualization of the aggregative optimization scenario at the last iteration with the trade-off parameters $a = 1$ and $b = 1$

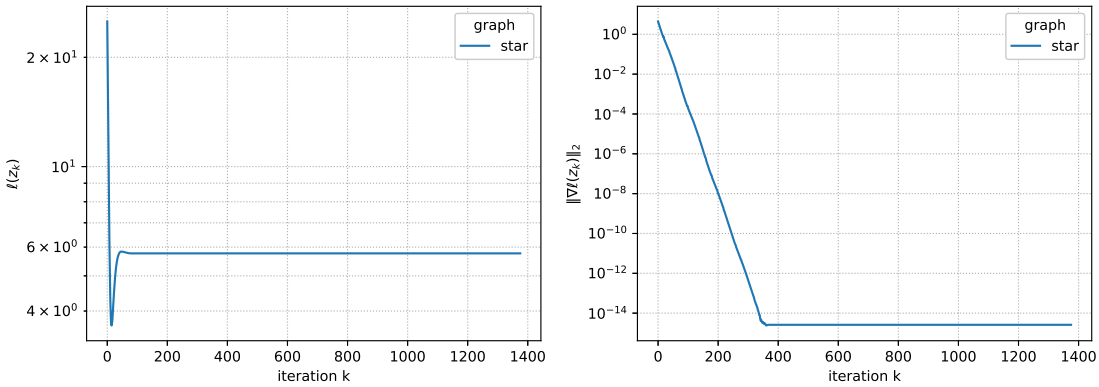

Figure 3.10: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *star* graph pattern with the trade-off parameters $a = 2$ and $b = 1$ retrieved from ROS 2

Figure 3.11: The RViz visualization of the aggregative optimization scenario at the last iteration with the trade-off parameters $a = 2$ and $b = 1$

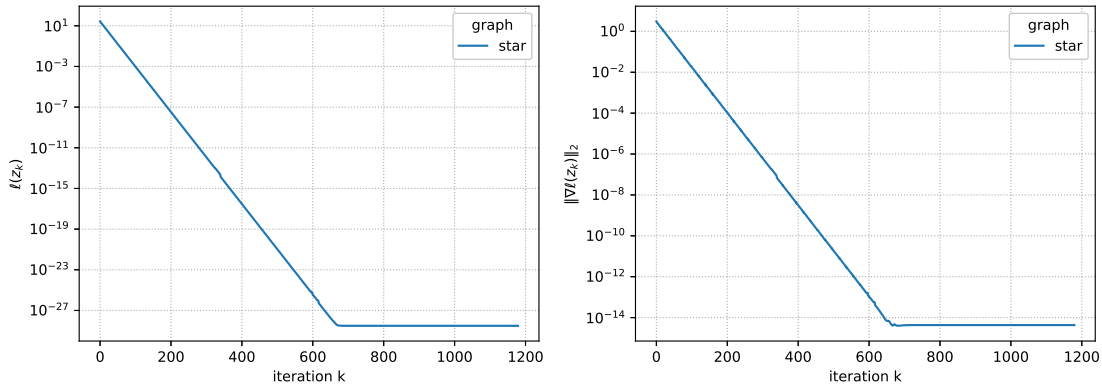

Figure 3.12: Evolution of the total cost (on the left) and the gradient norm of the total cost (on the right) across iterations for the *star* graph pattern with the trade-off parameters $a = 1$ and $b = 0$ retrieved from ROS 2

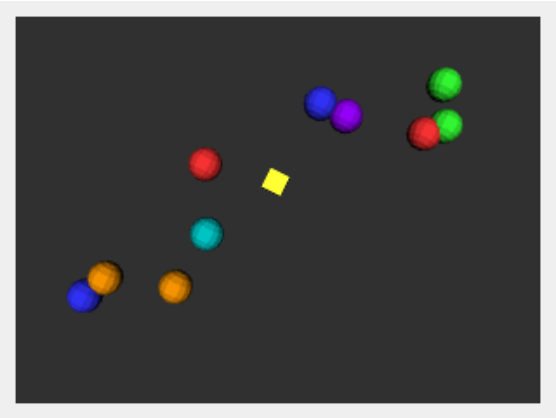

Figure 3.13: The RViz visualization of the aggregative optimization scenario at the last iteration with the trade-off parameters $a = 1$ and $b = 0$

Conclusions

The project comprises of two major tasks, where the first task concerns solving the distributed optimization problem using the Gradient Tracking (GT) algorithm with the aim of cooperatively localizing the targets in a multi-robot system, and the second task tackles the problem of solving a particular aggregative optimization problem using the Aggregative Tracking (AT) method. Chapter 1 introduced several important theoretical concepts like adjacency matrices and the GT algorithm necessary for solving the first task. In Chapter 1, the GT method has been implemented on the quadratic local functions, which provided appealing results for reduction of both the cost and the gradient norm. Chapter 2 used those notions from Chapter 1 to apply on a practical setting concerning the localization of the targets. The results in Chapter 2 demonstrated an excellent convergence of the metrics and therefore, a precise localization of our targets for four different standard deviation values of the noise added to the distance measurements. Lastly, Chapter 3 introduced the AT algorithm to efficiently solve the aggregative optimization scenario, where robots move towards their private targets, while preserving the vicinity relative to each other. The results of Chapter 3 also included good convergence to an optimal solution depicted in the metrics plots, plots of the animated behavior in Python and ROS 2.

Despite the achievements of the major goals related to the project, there were some obstacles faced by me during the implementation. The obstacles were mostly related to the ROS 2 implementation, which in itself has been a new experience for me. Even if the class tutorials regarding ROS 2 and Docker were quite self-explanatory and the necessary templates were provided, I still encountered several issues when tried to create my own package. At the start of the project, there were problems even with creating a container correctly. Nevertheless, as the time passed by, I successfully addressed those mistakes and bugs, which helped me to accomplish tangible results at the end.

Bibliography

[1] G. Notarstefano and I. Notarnicola. Lecture notes and slides from distributed autonomous systems m course. https://www.unibo.it/en/study/course-units-transferable-skills-moocs/course-unit-catalogue/course-unit/2024/454490, 2025. University of Bologna.

[2] Open Robotics. Rviz user guide. https://docs.ros.org/en/humble/Tutorials/Intermediate/RViz/RViz-User-Guide/RViz-User-Guide.html, 2025. Accessed: August 2025.