Gathering in $\mathbb{R}^d$ with a Randomized Algorithm

Jean-Lou De Carufel, Zhenhuan Cui

March 22, 2023

Table of Contents

Abstract

We report the Honour Project of Analyzing gathering in multiple dimensions with randomized algorithm which can gather robots in a distributed autonomonous system to a single point, superised by Jean-Lou De Carulfel in University of Ottawa. While this algorithm is not completely proven(e.g., Time-Complexity), its computer stimulation and visulization in this project indicate the correctness of this algorithm and performance of gathering in a randomized and multi-dimensional system. In theoratical aspect, the expected values of phases for all robots gather to a single point among different dimensions result in improved number of phases comparing to other convergence algorithm and adherence to sitmulated results. The prediction of final gathering point is a core component of this report, which gave us the basic ability to allocate possible final position range.

Chapter 1

Introduction

This technical report presents analysis of a convergence algorithm that gathers all robots in a system to a single point. Such algorithm is preformed on homogenous robots, theoratical points without mass, valumn, and area, with randomized initial positions in different dimensions.

1.1 Background

This section presents a convergence algorithm developed by Hideki Ando, from the University of TODO, that provides fundamental insights into how robots can gather to a single position under distributed autonomous robot systems. The algorithm operates under the assumption of homogeneity and memoryless among the robots, with each robot are completely the same and controlling its motion based on the available information on current state only. In the context of the algorithm, the initial state is randomized, and it is designed to operate correctly under all possible random initial distributions. The robots have limited Visibility, allowing them to perserve other robots on a limited subset of the system. Moreover, they lack the capacity for communication among all robots, forming a distributed and decentralized approach.

To describe this Algorithm, let $R = \{r_1, ..., r_n\}$ be all robots in the system, $r_i(t)$ be the position of robot i at certain time t, $dist(r_i(t), r_j(t))$ be the distance between robot i and robot j at time t, $P(t) = \{r_1(t), ..., r_n(t)\}$ be the positions of all robots at t and $P(0)$ be the beginning state. For simplicity, we assume that each robots is a dot of size 0 initially at random position (x, y) with Visibility VAll robots execute their movement to converge at a designated point, while respecting to their individual maximum move range σ, either simultaneously or asynchronously. In this report, we are presenting in simultanious manner. When $dist(r_i(t), r_j(t)) = 0$ for all i, j, it means all robots are gathered to a single points. The author defined an undirected graph $G(t) = (R, E(t))$ as $mutualvisibilitygraph$, by $(r_i, r_j) \in E(t)$ if and only if robot i and robot j

are within each other's sights at time t, a set $S_i(t)$ as visible robots of r_i at time t.

In order to accomplish the gathering, robots within a connected component must converge towards the central point while maintaining the distance V of each other. In this algorithm, each robot individually executes actions repeatedly referred to as a *phase*, which consists of the following two steps:

Step1: Calculate the Minimal Enclosing Circle
Determine the smallest circle that enclosing the positions of robots visible from r_i at time t, represented by set $\{r_j(t) \mid r_j \in S_i(t)\}$ which is a connected component.$c_i(t)$ denotes the center of $C_i(t)$.This circle and its center denoted as $C_i(t)$ and $c_i(t)$ provide a position for $r_i(t)$ to move towards a central point of the connected component and serve as a basis for further executions.If there's only $r_i(t)$ is enclosed in the circle, this phase $r_i(t)$ will not move.
Step2: Move to the center of $C_i(t)$
As robots possess a limited move range and move towards $c_i(t)$, the position of $r_i(t+1)$, denoted by x, consistently lies on the line segment $r_i(t)$ and $c_i(t)$. To prevent leaving from the connected component, the direction is determined by the position of $c_i(t)$ and the position x is calculated under the following constraints:
1) $dist(r_i(t), x) \leq \sigma$
2) For every robot $r_j \in S_i(t)$, x lies in the disc D_j whose center is the midpoint m_j of $r_i(t)$ and $r_j(t)$ and whose radius is V/2.(Ando, 1999, p.821)
To satisfy 2 conditions above, $r_i(t)$ may follow 3 distinct cases
1) To compute $dist(r_i(t), x)$ without leaving the connected component $S_i(t)$, we apply the following formula to each r_j in $S_i(t) - r_i$, excluding r_i, to determine the minimum distance l_i that r_i can move towards $c_i(t)$. This minimum distance is denoted by $\min_{r_j \in S_i(t)-r_i} l_j$ and referred to as LIMIT:

$$l_j = (d_j/2)cos\theta_j + sqrt(V/2)^2 - ((d_j/2)sin\theta_j)^2$$

2) r_i arrives at $c_i(t)$ directly. The distance is given by $dist(r_i(t), c_i(t))$ and called GOAL. 3) r_i move by its maximum move range which is simply σ

Since we uphold both conditions, we choose the minimum value of LIMIT, GOAL, σ, denoted by MOVE. Consequently, x is the point on the segment connecting $r_i(t)$ and $c_i(t)$ at a distance of MOVE from $r_i(t)$.

Following the execution of a phase, if robots have not yet converged to a single point, another phase is initiated simultaneously for all robots.

1.2 Objectives

The primary objectives of this technical report are to present the Analysis of another convergence algorithm which building upon the previously discussed one (Section 1.1). we have proposed a different approach to addressing the same problem under same distributed system as introduced, which we called Randomized convergence algorithm. We will first discuss the similarities and differences between our algorithm and previous algorithm, then delving into the specifics of how the robots are gathered to a single point.In this section, we will use same notations as we discussed in Background.

1.2.1 Similarities with the Previous Algorithm

Our algorithm shares some fundamental concepts and strategies with the previous approach, including:

1. Memoryless operation, where the robots do not rely on prior information from previous steps.
2. Round-based operation, where the algorithm executes *phase* simultaneously on all robots.
3. Randomized initial positions for the robots, ensuring a versatile solution for various starting configurations.

1.2.2 Differences from the Previous Algorithm

While our algorithm builds upon the foundations laid by the previous work, there are notable differences that distinguish our approach:

1. The robots in our algorithm have infinite visibility, allowing them to detect other robots' positions regardless of distance.
2. Our algorithm allows robots to have an infinite move range, which provides flexibility in maneuvering towards the target point.
3. Due to the robots' infinite visibility, there is no need to rely on connected components in our algorithm. Instead, we introduced a random algorithm to select the positions towards which the robots will move.

1.2.3 Convergence Process within a *phase* for r_i

In this section, we will discuss a single phase within the convergence process, detailing the steps the r_i takes to gather to a single point with other robots:

1. **Step 1**: Pick k robots randomly from R and let $P_i = \{r_{i,1}, r_{i,2}, ..., r_{i,k}\}$ be the set of robots we picked in the manner of sampling without replacement, which means it is possible to pick r_i more than once.
2. **Step 2**: Let $1 \leq j \leq k$ be such that $dist(r_i, r_{i,j})$ is minimized. Choose the position of the robot with the minimum distance, denoted as $r_{min}(t)$, as the target position for r_i to move towards.
3. **Step 3**: Move r_i to r_min

Same as the previous algorithm, all robots will execute the same *phase* simultanously until all robots gather to a point and will not affeact each other at any phase. Therefore, it is possible for robots to exchange their position, which achieved by $r_j(t+1) = r_i(t)$ and $r_i(t+1) = r_j(t)$. All robots will not move until all robots have deciede their target position, which can be considered as a high level of synchronization.

1.2.4 Look, Compute, Move Model with Randomized Convergence Algorithm

In the context of the randomized convergence algorithm, we introduce a modified version of the Look, Compute, Move model, which accounts for the random selection of a subset of k robots. This modification provides a more efficient and focused approach to decision-making for robots in a swarm.

Look Step: Pick and Observe

During the Look step, each robot randomly selects k robots from the set of all robots

R and forms a subset P_i. Instead of observing the positions of all robots, the robot now focuses on gathering position information only for the robots within the subset P_i. This selective observation enables the robot to process a smaller amount of data, reducing computational complexity and enhancing the efficiency of subsequent steps.

Compute Step: Identify Target

In the Compute step, the robot analyzes the positions of the robots in the subset P_i. It identifies the robot with the minimum distance from itself within P_i and uses its position as the target position for the next step. This step ensures that the robot's movement is directed towards a robot that is relevant and close by, further optimizing its behavior in the swarm.

Move Step: Approach Target

Finally, during the Move step, the robot moves towards the target position identified in the Compute step. This movement brings the robot closer to the chosen target, helping the swarm to converge over time. By following the modified Look, Compute, Move model, robots can efficiently work together to achieve convergence in a randomized fashion, minimizing the amount of time and energy spent on unnecessary movements.

By incorporating the "Pick" action into the Look step, we create a more accurate representation of the robots' decision-making process in the context of the randomized convergence algorithm. This adaptation aligns the Look, Compute, Move model with the specific requirements of the algorithm and results in a more efficient and streamlined behavior for robots in a swarm.

1.3 scope

In this section, we will outline the key aspects of our analysis of the proposed algorithm. Our primary focus is to examine the algorithm's behavior and performance, as well as to explore its properties through stimulation. Specifically, we will address the following points:

1. **Convergence Analysis**: Investigate whether the algorithm truly converges all robots to a single point. We will observe the behavior of the algorithm and explore visualization methods to better understand the convergence process.

2. **Performance Evaluation**: Use mathemetical methods to investigate gathering performance in term of probablity. Conduct simulations to assess the performance of the algorithm under various arguments and multiple dimensions. Through these calculations and simulations, we will gather data on various performance metrics, such as convergence time, the number of *phases*, and the efficiency of the algorithm.

3. **Property Exploration**: While we will not provide a formal proof for the algorithm, we will utilize simulation-based approaches to investigate its properties. By analyzing the results, and changing different scenarios of stimulation, we aim to gain insights into predicting final positions, understanding the impact of broken robtos, and other potential aspects of the algorithm's behaviour.

Through this analysis, we hope to achieve a comprehensive understanding of the proposed algorithm's capabilities and potential applications in the context of robotic gathering.

Chapter 2

Methodology

2.1 Stimulation Design

2.1.1 Overview

In this section, we present the Simulation Design for the Methodology chapter, where we introduce the implementation details of our code. The primary goal of this section is to provide an understanding of the underlying structure of the simulation and the various components involved in its execution. In this project, we are not using any AI or Multi-thread programming technologies.

2.1.2 Software and Libraries

Our simulation is implemented using the following programming language and libraries:

- **Programming Language**: Python (version 3.10.5)
- **Libraries**: Random, NumPy, Matplotlib, argparse.

These libraries are utilized for random picks, as previously mentioned in algorithm, numerical computations, visualization, mannually input of robot positions purposes. An object named Robot with argument of positions, name and moving method is introduced to simplify calculations.

2.1.3 Constants

In this section, we introduce the constants used throughout our simulation code. These constants serve as fixed parameters that help to define the behavior and constraints of the system. They are as follows:

- **PLAIN_FIGURE_SIZE**: The dimensions of the figure used to visualize the simulation (Defaultly 15 units wide by 5 units high).

- **ROBOT_NUMBER**: The total number of robots participating in the simulation.
- **BOUND**: Represents the upper and lower bounds for the x, y (or any additional dimensions) of the robots' positions, ensuring that the robots are confined within a predefined spatial region.
- **RANDOM_PICK_NUMBER**: The number of robots randomly picked in each *phase*.

These constants are set at the beginning of the simulation and remain unchanged throughout its execution. By adjusting the values of these constants, one can modify the conditions of the simulation to explore various scenarios and observe the impact on the algorithm's performance and properties.