SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

Distributed Drone Patrolling

INTERNET OF THINGS

Professor:
Novella Bartolini

Students:
Andrea Di Marco
Jemuel Espiritu Fuentes
Michele Granatiero
{dimarco.1835169, fuentes.1803530, granatiero.1812623}@studenti.uniroma1.it

---

Academic Year 2022/2023

Contents

1 Introduction

In this report we are going to explain our proposed solution to the Internet of Things course challenge. The key concept of our implementation is *mutability* because the system will adapt and change the algorithm depending on a number of factors, from wind to age of information thresholds, from the shape of the point cloud to the number of points. We opted to use two popular techniques when dealing with the famous **Travelling Salesman Problem**: a **Greedy** approach and the **Ant Colony** Optimization algorithm.

2 Drones

The drones in the simulation are quad-copters with a full range of possible directions and mostly constant velocity. Originally the drones would:

, a, a, reach a full stop and turn towards the next point.

1. Turn towards the target
2. Accelerate forward then
3. Touch it
4. Reach a full stop
5. Turn towards the next point

2.1 Stopping Process

The drones have a camera but since the task in analysis doesn't involve reporting information, they don't have to stop at the targets, only reach them. For this reason we removed the *"stop and rotate"* phase from the drone's behaviour. In this way we managed to obtain faster patrol routes overall. This led to a problem regarding the high angular velocity which over time resulted in the drones being unable to touch the targets and instead orbiting around them.

2.2 Rotating Process

To solve this last issue we decided to remove the rotation on the z-axis from the drone. Since we don't require the drones to *look* at the target but only to *touch* it, it shouldn't be a problem in these scenarios if the drone gets to its destination *backwards* or sideways.

2.3 Wind

We thought of a few approaches to deal with the presence of wind in the simulation, the most straightforward solution is to reduce the drone's ϵ parameter that dictates how close the drone must get to the point before considering it *touched*.
The reason why this seems to work is that with the default ϵ the drones start decelerating too soon, resulting in the wind pushing them back before they can touch the point. This happened in a few scenarios.
By decreasing the ϵ from 0.5 to 0.3, only when wind is present, the drones start decelerating when closer to the targets and they are able to touch the points more reliably.

3 Clustering

3.1 K-Means

At the beginning we wanted to try and use a different method called the *Spectral Clustering*. By looking at various examples it seemed that this particular method produces better suited clusters for the drone patrolling tasks. Alas those clusters seemed to only work when dealing with a huge number of points. Our testing on the provided simulation files (and more simulation tailor made by us) showed some odd behaviors and general worst performance than the simpler but generally effective *K-Means* clustering method that we ended up using.

3.2 Reordering

Our tests showed that it may happen that drones would crash into each other at the very beginning of the simulation, when they are trying to reach their respective clusters.
To prevent this from happening we rearrange clusters according to the y component of their centroids and then assign them to the closest drone.

3.3 Special Case

There is a very specific and unlikely case in which we either have more drones than targets or as many drones as we have targets.
In those situations the K-Means would return an error and we simply chose to manually assign a drone to every target while keeping the excess drones idle.

4 Task Assigner

This is where the biggest effort was made. We discussed many possible approaches and we settled on a hybrid approach. In this way we are able to confront different scenarios in a more elastic manner and deal with each problem accordingly.

4.1 Greedy Approach

The greedy solution at every step evaluates the targets' priorities in the cluster and points the drone towards the most urgent one based on our dynamic distance metric. This technique is good for environments that can radically change at any given step, for example when the targets have very different thresholds for the age of information requiring them to be visited more frequently than others.

4.2 Ant-Colony Approach

This algorithm send out a number of ants to different paths, being guided by the distance metric, and then returns the best one found. More specifically the ants choose probabilistically the next step of the path and then performs backward updates to said probability by adding the inverse of the path's distance. After a number of steps, the best path found by the ants will be the one with the highest probability of being chosen.

4.3 How to choose

An important aspect of our solution is the way we choose which algorithm to use.
Our tests indicate that the **greedy** approach may perform better in a those cases when:

- The wind is too strong: ant colony isn't able to take in consideration the change in direction caused by the wind.
- The cluster is small: ant colony simply requires more computations and the problem is small enough that greedy does well.
- The thresholds of the targets in the cluster are very different: ant colony computes the path without a good understanding of the growth in the age of information during a patrolling route but only after completing a patrol and computing the next route.

On the other hand, the **ant-colony** approach seems to perform better than the greedy approach when:

- The cluster has outliers far away from the center of mass: the greedy approach gives little to no attention to far away nodes and often bounces between neighbor nodes, this is bad for fairness.
- The thresholds of the targets in the cluster are similar: the path computed minimizes the overall distance traveled resulting in faster completion times.
- The cluster is big: greedy may take very inadequate decisions based on short term gains that will have bad long term consequences.

With these considerations, *cluster by cluster* we choose what approach is the best, resulting in some drones using different methods depending strictly on the cluster they have been assigned to.

4.4 Distance Metric

Our distance metric D is made of two parts:

1. d: euclidean distance between points.
2. B: *bonus* decided by the current state of the destination target.

So, given two scaling hyper-parameters α and β we will have that:

$$D = d\alpha - B\beta \tag{1}$$

4.4.1 Euclidean Distance

Given two points p_1 and p_2 we compute the euclidean distance d between them:

$$d = \sqrt{\left(p_{1,x} - p_{2,x}\right)^2 + \left(p_{1,y} - p_{2,y}\right)^2 + \left(p_{1,z} - p_{2,z}\right)^2} \tag{2}$$

4.4.2 Bonus

The *bonus* B is itself made of two parts:

1. A: Age of Information bonus
2. V: Violation bonus

Given:

- p_2: destination
- a: current age of information of p_2
- t: threshold of p_2

- w_a: age of information weight set by the simulation
- w_v: violation weight set by the simulation

then:

$$A = a^2 w_a \tag{3}$$

$$V = (a - t)\, w_v \tag{4}$$

Note that $V > 0$ if the threshold has been violated. We chose to square a as the distance's impact was generally too strong.

We compute the final bonus B by *reducing* A if there has been a violation, *increasing* it otherwise:

$$B = \begin{cases} \frac{A}{|V|}, & \text{if } V < 0. \\ A \cdot V, & \text{if } V > 0. \end{cases} \tag{5}$$

4.4.3 Special Cases

For the special cases in which either the age of information bonus A or the violation bonus V are equal to *zero*, we use the the one that isn't *zero*.
If both A and V are equal to *zero*, we just will just use the euclidean distance d.
If $|V| < 1$ we will swap the operations of the two cases.

$$B = \begin{cases} A \cdot V, & \text{if } V < 0 \wedge |V| < 1. \\ \frac{A}{|V|}, & \text{if } V > 0 \wedge |V| < 1. \end{cases} \tag{6}$$

5 Future Developments

Here we list a number of future improvements and ideas that we would have made with more time in the future.

- A smarter way of dealing with the wind by modifying the drone behaviour.
- More tests in order to iron the metric with better weights α and β.
- A more precise *Ant Colony* Optimization that considers the time component of each target and the drone movement over time.
- A better clustering that doesn't take the coordinates of the point as the only metric but takes into account the thresholds as well.
- Fixing the rotation of the drone over time.
- Fuzzy Clustering with shared points among clusters
- Collision detection and avoidance