

Internship report

Nathan Boyer

July 10, 2024

Abstract

It's possible to distribute the Internet to users via drones. However, this raises the question of how to place the drones around the users, and how to distribute the bandwidth between the different users. A reinforcement AI has already been designed to address this problem. However, in this article, we will see how learning and optimization can be combined to further improve performance.

1 Problem presentation

We have m users, classified into 3 categories, each class having its own bandwidth demand for a drone. We then want to place n drones, and for each drone, decide how much of its bandwidth it gives to each class of drone, so that as many drones as possible are satisfied, i.e. the bandwidth available to them is greater than or equal to their demand.

2 Problem solving by constrained optimization

2.1 Rigorous problem definition

This problem can be solved by constrained optimization.

First, we have to define the equation that computes how well a user will receive a base station's connection depending on where it is.

We have the following Equations for the drones's SINRs, which is a factor that represents how much data is effectively received by the user, in relation to the data sent by the base station to the user:

$$\text{SINR}_{i,j}^t = \frac{pc\mu(y_j, u_i^t) \left((\|y_j - u_i^t\|)^2 + (h)^2 \right)^{-\alpha/2}}{\sum_{k \in \mathcal{U} \setminus i} pc\mu(y_j, u_k^t) \left((\|y_j - u_k^t\|)^2 + (h)^2 \right)^{-\alpha/2} + \sigma^2}$$

To evaluate a configuration we will thus proceed as follows:

You first need to associate each user with its nearest base station.

You then share the bandwidth of a base station dedicated to a given class between all users of this class associated to this base station.

Then, you compute each user SINR and decide for the user is satisfied or not.

The percentage of satisfied user is thus what you want to maximize.

2.2 Optimization problem

This problem can be resolved by as an optimization problem, I will show you how to resolve it for 2 base stations.

Let's first define u a matrix of dimensions $N * N$, with N being the number of positions possible for a base station. $u_{i,j}$ is a binary variable that is equal to one if and only if the first base station is in position i and the second is position j .

bw is a $3 * 2$ matrix that represents the bandwidth each base station allocates to each user class.

δ is a vector that represents for every whether it is satisfied or not.

The function to optimize is then: $\max_{u,bw} \sum_{g \in \mathcal{G}} \delta_g$

Under the following constraints:

$$1. \sum_{b \in |\mathcal{U}|} \sum_{i,j \in \mathcal{U}} u_{i,j} \times bw_{\text{slice}(g),b} \times \text{bps}_g \frac{100}{G_{\text{conn}}(\text{slice}(g),b)} \geq th_g \times \delta_g \quad \forall g \in \mathcal{G}$$

$$2. bw_{em,b} + bw_{ur,b} + bw_{mm,b} = 1 \quad \forall b \in |\mathcal{U}|$$

$$3. bw_{em,b} \geq 0; bw_{ur,b} \geq 0; bw_{mm,b} \geq 0 \quad \forall b \in |\mathcal{U}|$$

$$4. \sum_{i,j \in \mathcal{U}} u_{i,j} = 1$$

$$5. \delta_g \in \{0, 1\} \quad \forall g \in \mathcal{G}$$

$$6. u_{i,j} \in \{0, 1\} \quad \forall i, j \in \{0, 1, \dots, |\mathcal{U}| - 1\}$$

The resolution of this optimization problem can be implemented in Python with the help of a module such as pulp.

2.3 Time Optimizations

To speed up the resolution of this problem, I implemented a few optimizations:

First, only positions in the convex envelope of the users can actually be candidates for optimal placements of the base stations. We can thus reduce by a lot the size of the matrix u .

Moreover, it is possible to go further but not without losing precision. Indeed, you can divide the users into k clusters and take the union of this convex envelope of this clusters. The graph of performance and time in function of the number of clusters can be found below.

However, since the computation times are pretty reasonable for 2 base stations, I decided to go with only 1 cluster.

Finally, multithreading can be used to further accelerate the computations.

3 Machine learning to solve the problem

However, a major issue is that this method of solving the problem by optimization is still pretty long. This can be problematic, especially if the users are moving, and it thus becomes necessary to recalculate the optimal position often.

Thus, a solution using machine learning was considered. Indeed, solving an instance of the problem would only require a forward pass of the neural network, which is negligible in time.

3.1 Reinforcement learning

This is why a reinforcement learning solution was imagined and implemented by Lorenzo Bellone.

Here I maybe present his results, and do I say they don't really work?

3.2 Supervised learning from the optimal solution

But, an other idea is to use supervised learning. By using an optimal agent, the neural network can learn to replicate the optimal solutions.

The neural network is then divided in 2 neural networks:

1. A neural network that takes as inputs the positions of the users and learns to output the best positions for the 2 base stations.
2. A neural network that takes as inputs the positions of the users and the positions of the base stations and learns to output the best slicing of bandwidths between the 3 classes of users.

As a side note, the type of each users is represented by 3 one-hots, one for each of the class.

Here, I present my results with a lot of different graphs

3.3 Mixer les 2

3.4 Les graph neural network

4 Ouverture: Utiliser la distance à l'optimum comme erreur pour l'apprentissage

5 Conclusion