Optimal PUBG

Maxim Pasko
Optimization Class Project. MIPT

Introduction

If you have ever played PUBG, you know that the key point of this game is to staying alive as long as you can. Also, it is very important to have a good loot (rifles, ammunition, heavy armor etc.). But there is a question: where should I jump off the plane to maximize my lifetime and to get good loot? So, in this work I'm trying to answer this question.

Loot map

I use data from the [1]. For each small square calculated efficiency score:

$$\text{min spawns} = 10, \text{penalty} = \min\left(1, \frac{\text{loot spawns count}}{\text{min spawns}}\right)$$
$$\text{weighted loot spawns} = \text{penalty} \cdot \text{loot spawns count}$$
$$\text{score} = \frac{\text{weighted loot spawns}}{\text{time to loot}}$$

As a result, we get this

Kills statistics

In work [2] we can see that most kills occur with distance between killer and victim less that 50 meters (which is equivalent half of small square), so we can calculate the probability of death for each small square separately.
In order to do it, I use dataset from [3]

Using this, also using the previous paragraph, we get a map with possible "good" points for the jump, not taking into account the trajectory

Function

Flight paths of an airplane are much more likely to touch the small squares in the center of the map than at the edges. Then it will not be fair to consider probability as just the number of deaths in a given square. Suppose that the beginning of the trajectory is chosen on one of the four sides of the square, and the end is a normal random variable whose mathematical expectation is a point diametrically opposite to the beginning. This leads to the fact that in the rounds trajectories are more often obtained, covering more locations. In this case, in order to nevilate the frequency of the trajectories, the probability of death in each square will be considered as $\ln(\text{deaths count})$.

Also, assume that players' distibution regarding the trajectory is a normal random value. It means that players is more likely to select a point closer to the trajectory.

So, let's mininmize this function:

$$f(x,y) = \frac{\ln(1 + \text{deaths count}) \cdot F(dist(x,y))}{\text{efficiency score of } (x,y)},$$

where F is a distribution function of normal random value, $dist(x,y)$ - distance between trajectory and (x,y).

Algorithm

To minimize this function, I used three different algoritms. Let's give a brief explanation of each of them (full explanation can be seen at [4], [5]):

1. Differential evolution: Generation of vectors is initialized. On the next iteration for every vector v_{old} in the last generation mutant vector is created using formula:
$$v_{mut} = v_1 + m_{rate} \cdot (v_2 - v_3),$$
where v_1, v_2, v_3 - random vectors from previous generation. Then v_{mut} and v_{old} crossover, that gives new vector v_{new}, and if $f(v_{new}) < f(v_{old})$, then v_{old} is replaced by v_{new}
2. Brute: A discrete grid is created, for each sect cell, the algorithm considers the value of the function in it.
3. Dual annealing: This algorithm is based on simulated annealing algorithm. It uses a distorted Cauchy-Lorentz visiting distribution, with its shape controlled by the parameter q_v:
$$g_{q_v}(\triangle x(t)) \propto \frac{\left[T_{q_v}(t)\right]^{-\frac{D}{3-q_v}}}{\left[1 + (q_v - 1)\frac{(\triangle x(t))^2}{[T_{q_v}(t)]^{\frac{2}{3-q_v}}}\right]^{\frac{1}{q_v-1}+\frac{D-1}{2}}},$$
which is used to generate a trial jump distance $\triangle x(t)$ of variable $x(t)$. The jump is accepted if it decreases function. Otherwise, it can be accepted with the probability $[1-(1-q_v)\beta \triangle f]^{\frac{1}{1-q_v}}$. The temperature decrease according to $T_{q_v}(t) = T_{q_v}(1)\frac{2^{q_v-1}-1}{(1+t)^{q_v-1}-1}$

Results

In order to check the quality of the program's work, I used another dataset in which I examined the early deaths of the players and calculated how many of them went to "bad" squares (that is, those that our program definitely would not have chosen). It turned out that 83% of all these deaths occurred at such points.
Code available at [6]. You can also use my telegram bot.

References

[1] Pubg interactive map. https://pubgg.com/maps/erangel.

[2] Pubg kill analysis. https://www.kaggle.com/etsc9287/pubg-kills-analysis, 2020.

[3] Pubg kills dataset. https://www.kaggle.com/skihikingkevin/pubg-match-deaths.

[4] Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. https://link.springer.com/article/10.1023/A:1008202821328.

[5] Brian Suomela Yang Xiang, Sylvain Gubian and Julia Hoeng. Generalized simulated annealing for global optimization: The gensa package. https://journal.r-project.org/archive/2013/RJ-2013-002/RJ-2013-002.pdf, 2013.

[6] Code. https://colab.research.google.com/drive/1gGeC5JpHFazB_QUmfVahIYH9htjTifcY.