Kalorian Game Jam Progression

An Algorithm to distribute Upgrade Cards across Matches

Vincent Kreuziger (Zeel)

7. Mai 2020

Inhaltsverzeichnis

1 Overview

Both players receive cards throughout a match and upon ending it. Amount and type of those cards are based on their performance throughout the match.

2 The Algorithm

T: Set of different towers
$u_1^t, \ldots, u_n^t$: Set of Standard upgrade cards for tower $t \in T$ $v_1^t, \ldots, v_n^t$: Set of Special upgrade cards for tower $t \in T$

Function evaluate(x) for $x \in \mathbb{R}$ assigning each x a unique u_i or v_i so all $u, v \in U \cup V$ get assigned to one x[1]. The function will give out poor cards if the return value is close to 0 and progressively better cards the greater the absolute return value is.

[1] Not the other way round: Each potential x is mapped to a specific card.

Examples:

$$\begin{aligned} evaluate(x) &= 1\ for\ 0 < x \leq 0.5 \\ &= 2\ for\ 0.5 < x \leq 1 \\ &= 3\ for\ 1 < x \leq 1.5 \\ &= 4\ for\ 1.5 < x \leq 2 \\ &= \vdots \\ &= n\ for\ 3 < x \\ &= -1\ for\ 0 \geq x < -0.5 \\ &= -2\ for\ 0.5 \geq x < -1 \\ &= -3\ for\ -1 \geq x < -1.5 \\ &= -4\ for\ -1.5 \geq x < -2 \\ &= \vdots \\ &= -m\ forx < -3 \end{aligned}$$

Changing those intervals changes the return values from the Gaussian distribution, meaning a card becomes more likely if its interval's absolute moves closer towards zero and vice versa.

Function assign(i,t) assigning every integer within the range of evaluate(x)[2]

Number of cards received upon end of match:

$$n = (int(c_1/a_1) + max(int(a_2/e^{c_1}), a_3) + a_4$$

For each card received its variance is driven by the following function:

$$s = (c_1 + b_1) \cdot b_2$$

Var	Meaning
a_{1-4}	constants
$b_{1,2}$	constants
c_1	waves survived by the players
c_2	Cards left in the inventory
a_1	every a_1 waves the players receive a card for reward
a_2	weighting of remaining cards, high means more fresh cards while the player is running low
a_3	covers the maximum so wasting all your cards doesn't yield 500 new draws
a_4	minimum of cards the player draws
e	Euler's number, taken to the power of tower types for higher weighting
s	Variance of quality, higher values mean for better upgrade cards
b_1	Baseline applied to b_2 without considering the wave reached
	Guarantees a minimum of card variance even if the first wave isn't survived
b_2	Weight of each wave, higher values make the waves more significant

We now roll a weighted die with T sides[3] n times. As a result we get the set $\{t_1, \ldots, t_n\}$ of towers that we receive cards for, a subset of all the towers available in game. Next we draw from a Gaussian Distribution N(0,s)[4], which provides a sample $\{x_1, \ldots, x_n\}$. As a result we get a set of n towers $\{n_1, \ldots, t_n\}$ and a set of n gaussian distributed numbers $\{x_1, \ldots, x_n\}$. Our upgrades are now $assign(evaluate(x_i), t_i)$ for all i.

[2] Meaning: $-m-1, n \quad \rightarrow$ see 2.
[3] Each side represents one type of tower.
[4] Mean 0, standard deviation s.

3 Pseudocode

4 Notes