多人在线竞技游戏中对随机因素的伪随机修正算法

一、引言

多人在线竞技游戏中通常会引入若干随机因素以增加游戏多样性，如MOBA（Multiplayer Online Battle Arena，即多人在线战术竞技游戏）中常见的“暴击”：攻击时有一定概率造成额外伤害。但随机因素的存在会削弱多人游戏的竞技性，降低玩家水平对胜负的影响。为了在保证玩法多样性的前提下减少其负面影响，通常会用伪随机算法代替纯随机（二项分布）。本文将以MOBA《英雄联盟》中的暴击（以下简称为“暴击”）为例推测其采用的伪随机算法的具体形式。

二、正文

1、纯随机算法（二项分布）的不合理性

对MOBA的一次对拼而言，由于其持续时间通常较短，双方一般没有机会进行足够次数的攻击以使得暴击的分布满足大数定律。若使用传统的二项分布，则有较大概率造成双方触发暴击的概率相同，但在一次对拼中其中一方的暴击触发次数大于另一方。设此事件为，发生此事件的概率可以利用二项分布的公式简单地计算：设一次对拼中双方各攻击次，暴击几率均为，则有

其部分分布如表2.1.1。

表2.1.1 不同下的值

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| 1 | 0.1800 | 0.3200 | 0.4200 | 0.4800 | 0.5000 | 0.4800 | 0.4200 | 0.3200 | 0.1800 |
| 2 | 0.3114 | 0.4864 | 0.5754 | 0.6144 | 0.6250 | 0.6144 | 0.5754 | 0.4864 | 0.3114 |
| 3 | 0.4088 | 0.5811 | 0.6514 | 0.6797 | 0.6875 | 0.6797 | 0.6514 | 0.5811 | 0.4088 |
| 4 | 0.4821 | 0.6402 | 0.6971 | 0.7201 | 0.7266 | 0.7201 | 0.6971 | 0.6402 | 0.4821 |
| 5 | 0.5383 | 0.6802 | 0.7284 | 0.7482 | 0.7539 | 0.7482 | 0.7284 | 0.6802 | 0.5383 |
| 6 | 0.5821 | 0.7093 | 0.7516 | 0.7693 | 0.7744 | 0.7693 | 0.7516 | 0.7093 | 0.5821 |
| 7 | 0.6169 | 0.7316 | 0.7697 | 0.7859 | 0.7905 | 0.7859 | 0.7697 | 0.7316 | 0.6169 |
| 8 | 0.6450 | 0.7493 | 0.7843 | 0.7993 | 0.8036 | 0.7993 | 0.7843 | 0.7493 | 0.6450 |
| 9 | 0.6681 | 0.7639 | 0.7965 | 0.8105 | 0.8145 | 0.8105 | 0.7965 | 0.7639 | 0.6681 |

令暴击次数完全相同可能有些极端，考虑事件：双方触发暴击次数差大于1，则

其部分分布如表2.1.2。

表2.1.2 不同下的值

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| 1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 2 | 0.0162 | 0.0512 | 0.0882 | 0.1152 | 0.125 | 0.1152 | 0.0882 | 0.0512 | 0.0162 |
| 3 | 0.0413 | 0.1126 | 0.1720 | 0.2074 | 0.2188 | 0.2074 | 0.1720 | 0.1126 | 0.0413 |
| 4 | 0.0708 | 0.1709 | 0.2404 | 0.2776 | 0.2891 | 0.2776 | 0.2404 | 0.1709 | 0.0708 |
| 5 | 0.1019 | 0.2224 | 0.2957 | 0.3326 | 0.3438 | 0.3326 | 0.2957 | 0.2224 | 0.1019 |
| 6 | 0.1329 | 0.2671 | 0.3410 | 0.3769 | 0.3877 | 0.3769 | 0.341 | 0.2671 | 0.1329 |
| 7 | 0.1630 | 0.3056 | 0.3787 | 0.4135 | 0.424 | 0.4135 | 0.3787 | 0.3056 | 0.1630 |
| 8 | 0.1916 | 0.3392 | 0.4108 | 0.4445 | 0.4545 | 0.4445 | 0.4108 | 0.3392 | 0.1916 |
| 9 | 0.2185 | 0.3685 | 0.4384 | 0.4710 | 0.4807 | 0.4710 | 0.4384 | 0.3685 | 0.2185 |

由表2.1.1与表2.1.2可见，事件与发生概率随的增大而显著增大，且大多大于0.5，趋向0.5。采用二项分布无疑会显著增加对拼中运气成分的占比，是不合理的。

2、伪随机算法的基本形式及性质

考虑双方暴击触发次数不同时的具体情况，可以想到大多为其中一方连续攻击了若干次但是均没有触发暴击。基于此，可以猜测修正后某次攻击触发暴击的概率应随这次攻击前连续未触发暴击的攻击次数的增加而增加，同时保证暴击次数在总攻击次数中的占比期望不变。

定义为连续-1次攻击未暴击后第次攻击触发暴击的概率。

0代表一次未暴击的攻击，1代表一次暴击，则一个连续攻击次的样本可以表示为一个长度为的01串。

定义长度为的暴击序列为-1个连续的0与1个1组成的01串，其中第一个0视为某一次暴击后的第一次攻击，则长度为的暴击序列出现概率

任何长度为的01串都可以分割为若干暴击序列与一个长度为的全0串的拼接，其中第个暴击序列的长度为，有

其中为暴击序列个数。，，。

依此我们可以计算出连续攻击n次的样本中暴击次数的期望

其中为所有长度为的01串构成的集合，为串中包含的暴击序列的个数。

利用该式计算的时间复杂度为指数级，在后文将介绍其优化算法。

对于暴击几率的场合，需满足

3、样本收集

在《英雄联盟》中，令暴击率为0.2、0.3、0.4、0.5、0.6，记录产生300次暴击的样本。

基于上述猜测，可以认为任意两个暴击序列是相互独立的。因此交换一个样本中任意两个暴击序列的位置不会改变该样本的理论出现概率。故对一个样本的记录可以由01串变为不同长度的暴击序列的出现次数。

对某个样本，定义为长度为的暴击序列的出现次数。由于长度大于10的暴击序列出现概率极小，故不做记录，但仍会计入暴击序列总数。样本见表2.3.1。

表2.3.1 不同暴击率下连续攻击1000次的样本

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 0.2 | 13 | 34 | 47 | 52 | 50 | 30 | 24 | 13 | 9 | 14 | 300 |
| 0.3 | 34 | 52 | 82 | 71 | 36 | 17 | 5 | 3 | 0 | 0 | 300 |
| 0.4 | 70 | 88 | 80 | 41 | 16 | 3 | 0 | 0 | 0 | 0 | 300 |
| 0.5 | 82 | 137 | 70 | 10 | 1 | 0 | 0 | 0 | 0 | 0 | 300 |
| 0.6 | 120 | 152 | 26 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 300 |

4、样本分析

由于出现了长度大于的暴击序列等价于第次攻击未暴击，又任意两个暴击序列相互独立，故可以得出的统计量公式

其值见表2.4.1。

表2.4.1 不同样本中统计量的值

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| 0.2 | 0.043 | 0.118 | 0.186 | 0.252 | 0.325 | 0.288 | 0.324 | 0.260 | 0.243 | 0.500 |
| 0.3 | 0.113 | 0.195 | 0.383 | 0.538 | 0.590 | 0.680 | 0.625 | 1.000 | 0 | 0 |
| 0.4 | 0.233 | 0.383 | 0.563 | 0.661 | 0.762 | 0.600 | 0.000 | 0.000 | 0.000 | 0.000 |
| 0.5 | 0.273 | 0.628 | 0.864 | 0.909 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 0.6 | 0.400 | 0.844 | 0.929 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

以为横坐标，为纵坐标绘制折线图，得图2.4.2。

图2.4.2 不同样本中统计量的分布

从图2.4.2中可以得出，当较小时，大致随线性增加；而当较大时，由于在中占比很小，较较小时的线性性质有明显偏移，故舍弃不用。

考虑到伪随机算法的初衷是保证每若干次攻击必定出现一次暴击，故的最大值应近似满足，同时结合图像线性性质进行微调。

故0.2、0.3、0.4取，0.5取，0.6取，作线性拟合，得表2.4.3。

表2.4.3 不同值下线性拟合结果

|  |  |  |
| --- | --- | --- |
|  | 线性拟合公式 |  |
| 0.2 |  | 0.9974 |
| 0.3 |  | 0.9933 |
| 0.4 |  | 0.9896 |
| 0.5 |  | 0.9981 |
| 0.6 |  | 0.9996 |

以为横坐标，拟合斜率为纵坐标做折线图，得图2.4.4。

图2.4.4 与拟合斜率的关系

对与作线性拟合，得，。

考虑到暴击率0时必定不暴击，暴击率1时必定暴击，即，，与线性拟合结果不符，考虑作多项式拟合，二次函数拟合得

，效果明显优于线性拟合。

则有结论如下

5、结果正确性验证

首先介绍前文提到的计算的优化算法。

对一个攻击次数为的样本，定义第次攻击为暴击的概率为，由期望定义有

关于的求法，考虑使用递推，考虑枚举第次攻击前连续未暴击的攻击次数，则有

其中。

考虑到大部分情况下不会特别小，即当增大到一定值时，有

该值通常小于10，故该递推算法复杂度为线性。

依此算法计算不同值下时的，得表2.5.1。

表2.5.1 伪随机算法下不同值对应的占比

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|  | 0.0842 | 0.0968 | 0.2897 | 0.3881 | 0.4897 | 0.5918 | 0.6922 | 0.7807 | 0.9124 |
| 误差 | 0.1579 | 0.5161 | 0.0345 | 0.0297 | 0.0206 | 0.0136 | 0.0111 | 0.0241 | 0.0138 |

由表可知当时理论结果与期望结果误差很小。

较小时误差较大可能是因为用C++程序求解过程中浮点数导致的精度损失。

6、结果合理性验证

回到最初的问题，伪随机算法的初衷是为了减少对拼过程中运气成分的影响，下面使用求得的伪随机分布重新计算，采用枚举01串的方式计算，结果见表2.6.1与表2.6.2。

表2.6.1 伪随机算法下的

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| 1 | 0.0316 | 0.1008 | 0.1981 | 0.3088 | 0.4128 | 0.4848 | 0.4939 | 0.4039 | 0.1736 |
| 2 | 0.0908 | 0.2614 | 0.4348 | 0.5337 | 0.5218 | 0.4317 | 0.4267 | 0.4994 | 0.2988 |
| 3 | 0.1698 | 0.4185 | 0.5516 | 0.5333 | 0.5014 | 0.5604 | 0.5334 | 0.5538 | 0.3907 |
| 4 | 0.2586 | 0.5236 | 0.5582 | 0.5460 | 0.5930 | 0.5763 | 0.5791 | 0.5997 | 0.4596 |
| 5 | 0.3467 | 0.5673 | 0.5585 | 0.6043 | 0.6075 | 0.6225 | 0.6067 | 0.6339 | 0.5123 |
| 6 | 0.4252 | 0.5734 | 0.5914 | 0.6276 | 0.6400 | 0.6444 | 0.6377 | 0.6603 | 0.5536 |
| 7 | 0.4882 | 0.5733 | 0.6258 | 0.6452 | 0.6606 | 0.6682 | 0.6592 | 0.6817 | 0.5867 |
| 8 | 0.5333 | 0.5842 | 0.6441 | 0.6670 | 0.6789 | 0.6859 | 0.6777 | 0.6995 | 0.6136 |
| 9 | 0.5614 | 0.6054 | 0.6569 | 0.6825 | 0.6950 | 0.7017 | 0.6938 | 0.7147 | 0.6360 |

表2.6.2 伪随机算法下的

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| 1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 2 | 0.0005 | 0.0048 | 0.0172 | 0.0364 | 0.0502 | 0.0349 | 0.0000 | 0.0000 | 0.0000 |
| 3 | 0.0023 | 0.0194 | 0.0533 | 0.0748 | 0.0558 | 0.0604 | 0.0677 | 0.0587 | 0.0136 |
| 4 | 0.0064 | 0.0434 | 0.0855 | 0.0852 | 0.0992 | 0.1031 | 0.0794 | 0.1029 | 0.0346 |
| 5 | 0.0133 | 0.0707 | 0.1015 | 0.1133 | 0.1344 | 0.1343 | 0.1244 | 0.1429 | 0.0592 |
| 6 | 0.0234 | 0.0937 | 0.1161 | 0.1520 | 0.1589 | 0.1681 | 0.1533 | 0.1796 | 0.0851 |
| 7 | 0.0363 | 0.1095 | 0.1411 | 0.1745 | 0.1911 | 0.1976 | 0.1832 | 0.2126 | 0.1112 |
| 8 | 0.0510 | 0.1209 | 0.1687 | 0.1976 | 0.2159 | 0.2252 | 0.2109 | 0.2422 | 0.1366 |
| 9 | 0.0664 | 0.1331 | 0.1906 | 0.2225 | 0.2406 | 0.2504 | 0.2358 | 0.2689 | 0.1610 |

对比表2.1.1与表2.6.1，表2.1.2与表2.6.2，可以看出与显著减小，具体优化效果见表2.6.3与2.6.4，其中优化量

表2.6.1 优化量

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| 1 | 0.8244 | 0.6850 | 0.5283 | 0.3567 | 0.1744 | -0.0100 | -0.1760 | -0.2622 | 0.0356 |
| 2 | 0.7084 | 0.4626 | 0.2444 | 0.1313 | 0.1651 | 0.2974 | 0.2584 | -0.0267 | 0.0405 |
| 3 | 0.5846 | 0.2798 | 0.1532 | 0.2154 | 0.2707 | 0.1755 | 0.1811 | 0.0470 | 0.0443 |
| 4 | 0.4636 | 0.1821 | 0.1993 | 0.2418 | 0.1839 | 0.1997 | 0.1693 | 0.0633 | 0.0467 |
| 5 | 0.3559 | 0.1660 | 0.2333 | 0.1923 | 0.1942 | 0.1680 | 0.1671 | 0.0681 | 0.0483 |
| 6 | 0.2695 | 0.1916 | 0.2131 | 0.1842 | 0.1736 | 0.1624 | 0.1515 | 0.0691 | 0.0490 |
| 7 | 0.2086 | 0.2164 | 0.1870 | 0.1790 | 0.1643 | 0.1498 | 0.1436 | 0.0682 | 0.0490 |
| 8 | 0.1732 | 0.2203 | 0.1788 | 0.1655 | 0.1552 | 0.1419 | 0.1359 | 0.0665 | 0.0487 |
| 9 | 0.1597 | 0.2075 | 0.1753 | 0.1579 | 0.1467 | 0.1342 | 0.1289 | 0.0644 | 0.0480 |

表2.6.2 优化量

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| 1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 2 | 0.9691 | 0.9063 | 0.8050 | 0.6840 | 0.5984 | 0.6970 | 1.0000 | 1.0000 | 1.0000 |
| 3 | 0.9443 | 0.8277 | 0.6901 | 0.6393 | 0.7450 | 0.7088 | 0.6064 | 0.4787 | 0.6707 |
| 4 | 0.9096 | 0.7461 | 0.6443 | 0.6931 | 0.6569 | 0.6286 | 0.6697 | 0.3979 | 0.5113 |
| 5 | 0.8695 | 0.6821 | 0.6567 | 0.6594 | 0.6091 | 0.5962 | 0.5793 | 0.3575 | 0.4190 |
| 6 | 0.8239 | 0.6492 | 0.6595 | 0.5967 | 0.5901 | 0.5540 | 0.5504 | 0.3276 | 0.3597 |
| 7 | 0.7773 | 0.6417 | 0.6274 | 0.5780 | 0.5493 | 0.5221 | 0.5162 | 0.3043 | 0.3178 |
| 8 | 0.7338 | 0.6436 | 0.5893 | 0.5555 | 0.5250 | 0.4934 | 0.4866 | 0.2860 | 0.2871 |
| 9 | 0.6961 | 0.6388 | 0.5652 | 0.5276 | 0.4995 | 0.4684 | 0.4621 | 0.2703 | 0.2632 |