

# MR. CURIOSITY 档案<sup>\*</sup>

杨润哲

2015.8.18

---

<sup>\*</sup>Final Project Report, Principle and Practice of Computer Algorithms, 2014  
ACM Class, Summer 2015

# Contents

<b>1</b>	<b>简介</b>	<b>3</b>
1.1	规则理解 . . . . .	3
1.2	比赛环境 . . . . .	3
1.3	地图大小和布局 . . . . .	4
<b>2</b>	<b>策略分析</b>	<b>5</b>
2.1	搜索策略 . . . . .	5
2.2	状态评价策略 . . . . .	6
2.3	基于博弈理论的决策 . . . . .	7
2.4	时间性能优化 . . . . .	8
<b>3</b>	<b>结果分析</b>	<b>9</b>
3.1	我的 Bots . . . . .	9
3.1.1	Dumb_Ways_To_Die . . . . .	9
3.1.2	Drag_Me_Down . . . . .	10
3.1.3	Mr_Curiosity . . . . .	10
3.2	对 Mr_Curiosity 的评价 . . . . .	10
3.2.1	优势 . . . . .	10
3.2.2	缺陷 . . . . .	11
<b>4</b>	<b>感谢</b>	<b>12</b>

# 1 简介

今年 PPCA 大作业是一个关于“贪吃蛇”的 AI 设计。和传统的贪吃蛇游戏不同的地方这个贪吃蛇游戏是双人对战的形式，双方每回合同时做出决策。蛇会在一些回合自动伸长。直到有一方蛇死亡，比赛结束，另一方获胜。

我从 2015 年 8 月 3 日开始设计我的 Snake 大作业 Ai，期间对策略进行了两次升级。最终的 Bot Mr\_Curiosity 完成于 8 月 11 日。Ai 设计给 PPCA 的课程任务带来了不少欢乐。谨以此文档纪念这段时光。

```
1 //  
2 // main.cpp  
3 // Snack Proj  
4 //  
5 // Created by Runzhe Yang on 8/3/15.  
6 // Copyright (c) 2015 Runzhe Yang. All rights reserved.  
7 //
```

## 1.1 规则理解

比赛开始时给定  $n \times m$  的网格，网络由  $1 \times 1$  的草地与障碍物构成，玩家在其中通过控制蛇头的朝向 (上、下、左、右) 来操纵自己的一条蛇 (蛇是一系列相邻坐标构成的有限不重复序列，序列中第一个坐标代表蛇头)，蛇以每回合前进 1 格 (前进即为序列头插入蛇头指向方向下一格坐标，删除序列末位坐标)。蛇的初始位置在网格中的左上角  $[1,1]$  与右下角  $[n,m]$ ，初始长度为 1 格。蛇会自动伸长 (即不删除序列末尾坐标)，前 10 回合每回合长度增加 1，从第 11 回合开始，每 3 回合长度增加 1。

蛇头 (序列的第一个坐标)，在网格外、障碍物、自己蛇的身体 (即序列重复)、对方蛇的身体 (即与对方序列有相同坐标)，或非法操作均判定为死亡。任何一条蛇死亡时，游戏结束。若蛇同时死亡，判定为平局，否则先死的一方输，另一方赢。

每回合裁判程序调用两玩家的程序请求决策。所以输入是当前状态，输出是玩家的决策。

## 1.2 比赛环境

这次 PPCA 的大作业应同学们要求提供了可视化测评环境。在北京大学信息科学技术学院，人工智能实验室的 Botzone 上进行测评。

同学们可以在这个平台上提交自己的 Ai Bot 并且可以不断更新升级，历史版本都会保留。每场对战的输入输出 log 都可查看，方便调试大家程序。玩家可以让自己的 Bot 参与天梯排名。

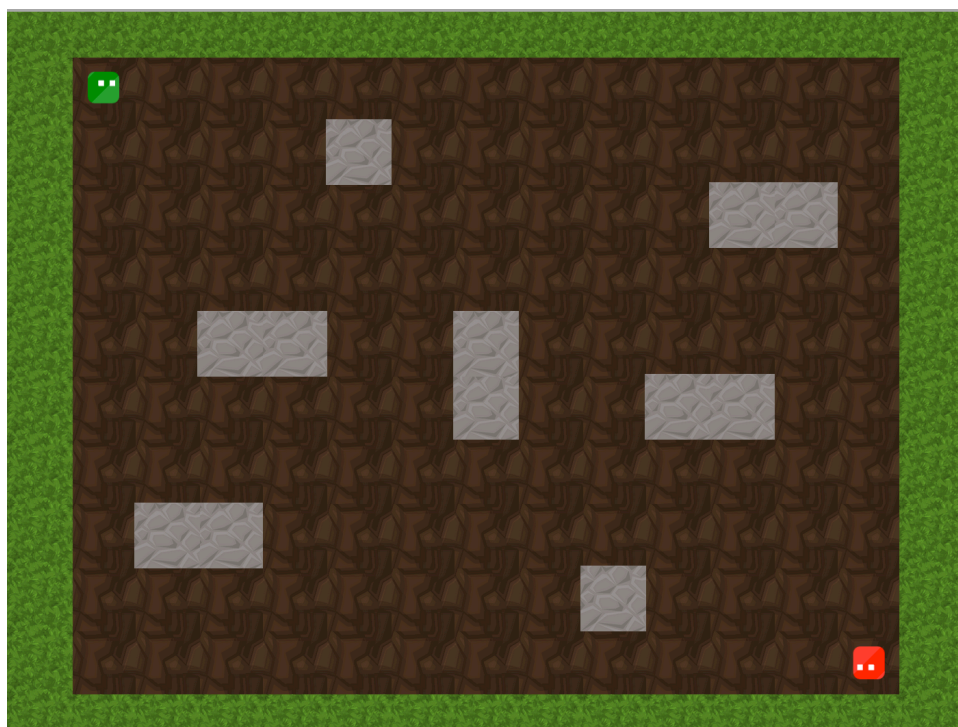
测评双方通过开游戏桌对战的方式决出胜负，胜负结果会在历史对局中显示。



### 1.3 地图大小和布局

棋盘大小设置开始时随机给定的  $n \times m$  的网格，网络由  $1 \times 1$  的草地与障碍物构成。地图边长满足：

- $n \in [10,16]$   $m \in [10,12]$



## 2 策略分析

### 2.1 搜索策略

和一般用深度优先搜索 (dfs) 回溯状态的搜索方法不同, 我的 Ai 采取了广度优先搜索 (bfs) 的搜索方式, 对一开始在 possibleDire[] 中的可能决策进行搜索。

搜索用队列 simuDeci 来维护, 队列元素为由两个记录了历史自己和对手历史决策字符串 mySimuD, eneSimuD 组成的二元组, 每次搜索之前通过队头的决策记录恢复出两条已经搜索到的两条蛇的状态, 用两个 list fakeSnake[2] 保存下来再继续搜索之后的可能状态。

因为需要逐层搜索, 所以在每次搜索前记录下当前层的状态数, 在状态出队时 --ThisLevelSize, 当 ThisLevelSize $\leq$ 0 时当前层搜索完成, 已到达的搜索层数加一。

如果当前状态没有转移了, 也就是其中一条蛇已经死了, 但搜索还没有达到目标层数时, 搜索将会继续进行, 当前的状态也会保留下来, 在状态评价的阶段将会被考虑。实现这一判断只需要对队头出队的元素加一个特判, 即出队元素中我方 (或者对方) 历史决策的步数是否为当前搜索层数少一, 如果不是, 将该出队元素直接加入队尾, 下一元素出队。历史决策的步数由字符串的长度之间表示。当前层数可以由虚拟的回合计数器 fakeTotal 确定。该部分代码实现如下:

```
if (mySimuD.length() <= fakeTotal - startTotal) {
    simuDeci.push(tmp);
    simuDeci.pop();
    if (--ThisLevelSize <= 0)
        break;
    continue;
}
```

记录决策而不是蛇的状态可以减少空间上的开销。但是这样从每一步开始搜索就需要对蛇的状态进行一次复原, 消耗了一定的时间, 而且这个常数乘上状态数会比较大。但事实上, 从程序的表现来看, 搜索并没有耗费太多的时间, 如果仅仅只有搜索对于所有的地图情况都可以搜到 7 层。

目标层数的搜索全部完成后, 将队列 simuDeci 中的全部元素导入 vector<PSS> waitForJudging 中, 并且按照长度和字符串双关键字由大到小排序。进入状态评价和决策阶段。

## 2.2 状态评价策略

怎么来衡量某一局面的好坏呢？怎样把这种好坏的程度表示出来？一开始的朴素想法是通过计算蛇的可行面积，即在一定步数内蛇头所能达到的位置的总数，来对当前蛇可以生存的局数做一个简单的估计。实践证明，这样的评价方式是十分高效可行的。甚至可以取得好于单纯搜索的效果。

仅仅考虑自己的可行面积是不够的。获取胜利不仅仅是要确保自己有足够大的生存空间，还应该使对手的生存空间相对自己较少。所以马上能够想到一个方式是定义  $\gamma = \text{我方可行面积} / \text{对方可行面积}$ 。这样衡量下的  $\gamma$  越大，局面在这一时刻相对越优。

这样的考虑还是朴素且不精确的。由于双方是同时移动，可行面积不能出现交叠。但是在之前的评价方法中却没有考虑这一情况。我采用了交替进行 floodfill 的方法，先对方扩展一层，再我方扩展一层的方式，计算可行面积。这种方法也会导致一个问题，就是对方由于先扩展所以会限制我方的可行面积。我的解决方法是改变双方能够到达格子的权值，对方的能到达的格子赋权值 ATA，我方能够到达的格子赋值 DEF，若我方到达某一格子时对方已经到达过，那么获得的权值将减去 BAL；若对方不能到达我方已经经过的格子。

这样重新定义的

$$\gamma = \frac{\Sigma(DEF - BAL)}{\Sigma ATA}$$

即作为每种状态的估价。

对方所能到达的格子赋的权值 ATA 越大，对方损失活动范围对  $\gamma$  的影响越大；我方能够到达的格子的赋权值 DEF 越大，我方获得活动范围对  $\gamma$  的影响越大；我方到达对方格子减去的权值 BAL 越大，我方损失活动范围对  $\gamma$  的影响越大。ATA 可以作为估价对进攻性的重要，DEF 与 BAL 也能对我方冒险性起到调节作用，其中 DEF 在小地图时表现为我方寻找并占领空地展开防御的能力，BAL 决定了我方在多少的优势下会选择出击。

现在仍有一个问题没有解决。双方的拓展究竟设计为几层才比较合理呢？如果层数太多，不但会导致估价的效率低下，而且我方可行面积将和地图剩余空白面积差不多，对于那些容易被封死的情况，将无法有效地衡量。我设计了我方拓展层数 MYCONSIDER，对方拓展层数 ENECONSIDER，来控制双方的拓展。

最后确定的上述几个参数如下，能够取得不错的效果：

- const int DEF = 4;
- const int ATA = 6;
- const int BAL = 3;

- `const int MYCONSIDER = 40;`
- `const int ENECONSIDER = 14;`

## 2.3 基于博弈理论的决策

按照上一节给出的评价方法，我对搜索得到的 `waitForJudging` 中的每一种状态给出评价。假设这些评价都是合理客观的，我们仍然不能直接由这些状态判断某一个决策的好坏。但至少我们已经获得了足够多的评价依据。只要用好这些依据，我们就可以做出好的决策。

由于双方是同时做出决策，所以不能通过只模拟对方的决策来做出最佳回应。由于之前已经得到了每种状态的评价，即每种决策带来的收益。这里有一个假设：在上一节给定的评价方式下，我方决策收益和对方决策收益是一致的。即同在该种评价方式下，我方收益越高，对方收益越低。故对于对手来说，较优的选择是我方收益较低的选择。

于是考虑如下的收益矩阵：

$$\begin{pmatrix} C_{1a} & C_{1b} & C_{1c} \\ C_{2a} & C_{2b} & C_{2c} \\ C_{3a} & C_{3b} & C_{3c} \end{pmatrix}$$

行表示我方决策 {1, 2, 3}，列表示对方决策 {a,b,c}， $C_{ij}$  为相应决策下对应状态的  $\gamma$  值。因为是同时做出选择，对于每一种我方决策，对方都有可能选择好的对策和坏的对策；同样对于对方的每一种决策，我方都有可能做出好的或者坏的决策。在该决策阶段，我将尽可能地选择好的那种决策。

首先对于我的每种决策，假设对方总能选择到最优的对应策略，即得到在对方总是最优情况下的我的决策的评价，

$$P_i = \min\{C_{ia}, C_{ib}, C_{ic}\}, (i \in \{1, 2, 3\})$$

再从这三个决策中选出最大者作为最终的决定：

$$Final\_Deci = \arg \max\{P_1, P_2, P_3\}$$

也就是对于收益矩阵先按列取最小值，再按照行取最大值，所得元素的行即为“最坏情况下的最优决定”。如果对手不没有按照假定的那样走最优解，我方的决策只会更好。

通过这种方式从决策树底层向上计算出每一阶段的最佳决策，直到计算出当前需要做出决策的这一层，返回我方的决策。这是我的 `Ai` 的核心部分，需要考虑较多的情况以及实现的细节（决策断层？用 `Map` 来实现？），占据了大约 200 行的代码量。

## 2.4 时间性能优化

基于以上的考虑，Ai 的策略已经基本完备，但是在实现的时候，发现效率并非尽如人意，在一般状态下仅能经行五步搜索，需要进行全面的时间性能优化。

在删除一些冗余计算后，决策阶段只需要在开始时进行一次排序，后面的排序都是不必要的，程序大概快了 100ms，可见原来的程序非常地浪费资源，但是这样的提升还是非常不令人满意的。

在决策阶段用到的大量的插入删除和查询，原来用 STL 的 Map 来实现，但 Map 维护的有序性在这里不是必要的，同时也浪费了不少的时间（有一部分是写法上的），改用 Unordered\_Map 后，程序效率有了大幅的提升，效率提高了将近 30%。可是多出来的时间并不能使得程序多搜一层。

反复调试，发现最耗时的部分出现在状态评估阶段，sample 中实现的 insnake 需要遍历两个 list，效率十分低下。于是新开一个二维数组 visMap[] 来记录该时刻这个位置是否有蛇的身体。每次搜索的起始状态更新后更新 visMap[]，这样减少了 3/4 的对 list 的遍历。除了评估阶段对 floodfill 进行了这样优化，搜索阶段也进行了这样的优化。改进后效率提升明显，耗时仅仅为原来的 1/5。搜索层数已经可以稳定地提升到 6 层。

另外由于浮点数计算较慢，对  $\gamma$  的计算全部改为 long long，计算方式为乘上一个较大的整数再做整除。当然对于对方为死路的情况，避免除零做了特判。

$$\gamma = \Sigma(DEF - BAL) \ll 18 / \Sigma ATA$$

这样一来时间性能相当可观了，为了充分利用时间，尤其是在小地图的时候能够尽量搜的多一点，必须动态调整搜索的层数。

```
SearchDepth = deepRegion(total);
```

由于单纯搜索耗时其实很少，我采取了通过先进行一轮搜索，预处理出最大搜索层数的办法。保险起见，设定最多处理的状态数不超过 STATUS\_SUP，而且最大的搜索层数不超过 SearchDepthSup 层。

经过模拟最坏情况，确定这两个参数为：

- const int SearchDepthSup = 12;
- const int STATUS\_SUP = 40000;

优化后，我的 Ai 运行非常流畅，比赛有很强的观赏性，而且在任何情况下都不会 TLE。我的最终版本 Ai，Mr\_Curiosity 正是采取了上述策略。



### 3 结果分析

#### 3.1 我的 Bots

我一共创作了 3 个 Bots,分别是 Dumb\_Ways\_To\_Die,Drag\_Me\_Down,Mr\_Curiosity, 这三个 Ai 在他们对应的时期都有着良好的表现。

Bots 施工中 在 Ai 设计阶段起到了不小的作用, 用这个 Bot 调出了不少错误, 是另外三个 Ai 的原型。

Snake	▼
Dumb_Ways_To_Die	
最新版本: 15	
Move Towards Death!	
★ 排名分: 1028.93	+ 𠵿
Mr_Curiosity	
最新版本: 1	
Is it true what they've been saying about you?	
★ 排名分: 1365.89	+ 𠵿
Drag_Me_Down	
最新版本: 1	
With out your love nobody can drag me down.	
★ 排名分: 1288.19	+ 𠵿
__施工中	
最新版本: 61	
施工Bot	
★ 排名分: 1000.00	+ 𠵿

(a) 我的所有 BOTS

59	??
58	?
57	NO TLE!
56	NO TLE!!
55	NO TLE! NO RE!
54	没有TLE的可能性。
53	RE好了没有哇?!!
52	RE好了没有哇
51	花式RE

(b) 用来调试的 施工中

##### 3.1.1 Dumb\_Ways\_To\_Die

只有状态评价函数, 对 possibleDire 中的决策对应的状态进行评估, 已经优于 Bot 榜上大量的 Ai, 而且甚至优于一些单纯搜索, 估价并不完善的 Ai。Dumb\_Ways\_To\_Die 展现出很强的进攻积极性, 而且代码十分简洁, 只用了 floodfill 进行面积的计算。

修改了 15 个版本, 最终觉得提升空间较小, 对战有搜索有估价的 Ai 时, 明显处于劣势, 决定加上搜索。

### 3.1.2 Drag\_Me\_Down

Drag\_Me\_Down 是相对成熟的 Ai，参与了第一次瑞士轮，赢 4 场输 1 场，排名第 3。Drag\_Me\_Down 结合了搜索与 Dumb\_Ways\_To\_Die 的估价，在各种地图下都有较好的表现，但是优于搜索层数只有固定的五层，效率较低，在面对可以搜索 8、9 层的对手时，显得较为弱小。

尤其在小地图上，表现得进攻力不足。最终决定对效率进行优化，增加搜索层数。

### 3.1.3 Mr\_Curiosity

最终版本的 Ai，效率是 Drag\_Me\_Down 的 5 倍，不但最坏情况下多搜了一层，而且动态调整搜索层数。平均可以搜索 78 层，在对战中有较好的表现。对所现在有强度的 Ai 均可以保证一定的胜率。

参加了第二第三场瑞士轮，由于出师不利，遭遇劲敌仇伟和陆一洲皆败，一次 3 胜 2 负排第 7 名，一次 4 胜一负排第 4 名。最终轮排第四名。

在 Bot 天梯中 Mr\_Curiosity 稳定发挥，登上了榜单的前十名。

## 3.2 对 Mr\_Curiosity 的评价

Mr\_Curiosity 是我最终的 Ai。虽然在对战中由于运气的原因未能够正常发挥，没有遇到恰当的对手导致排名变化的路线出乎意料，但最终的排名还是合理的。

对于特定的几个 Ai，Mr\_Curiosity 会展现出明显的劣势，但是对于个别比较强的 Ai，Mr\_Curiosity 又会表现得十分出色。总体而言 Mr\_Curiosity 攻守平衡，没有太明显的 BUG，是一款比较满意的作品。

我的这个最终 Ai，Mr\_Curiosity 正是采取了这篇文章所介绍的全部策略。但同时正因为采取了这些策略，所以 Mr\_Curiosity 还有许多需要改进的地方和不小的提升空间。最后我想对策略的优劣做进一步的分析与评价。

### 3.2.1 优势

经过分析，我认为采用了本文策略的 Ai，Mr\_Curiosity 有如下优势。凭借着这些优势，Mr\_Curiosity 才有了出色的表现。

**合理估价** Mr\_Curiosity 的估价十分细致，尤其是 MYCONSIDER、ENECONSIDER、ATA、DEF、BAL 这五个参数起到的进攻能力与防御能力的调节，在实际测试中有着非常良好的稳定的表现。用改进版的 floodfill 进行拓展

计算面积，其实是在原先搜索步数的基础上，进行了“不太精确”的进一步搜索，这样的搜索可以穷尽到结局。

**动态搜索** 动态的搜索过程使得搜索步数根据局面变化，对于大部分的局面可以有 78 步的搜索深度，有时候可以达到 111 步的深度，在最坏情况下保证了 6 步的搜索深度。估价难免会有不准确的地方，但是搜索的结果是保证精确的。加深搜索的深度一定可以给程序带来质的变化。

**精细决策** 决策过程是 Mr\_Curiosity 的一大亮点。基于博弈理论的这套决策机制，自底向上返回最优决策，保证了决策的正确性。这里说的正确性，即是在有限搜索层数下，假定估价合理的当前最坏情况下的最好决策，并且保证了如果不是最坏情况，该决策也要比最坏情况下的同种决策要优（但不能保证比非最坏情况下的其它决策优）。

### 3.2.2 缺陷































需要承认的是，Mr\_Curiosity 仍然存在一些缺陷。最需要说的两点分别是它的时间浪费以及对绝杀和逃生机会的浪费。




**时间浪费** 从 Ai 的对局表现看来，Mr\_Curiosity 都是迅速做出决策，时间性能最然很好，但是耗时都非常尴尬，平均大概在 400ms 左右，如果再多搜索一层，则有超时的危险。而平台的时限为 1500ms，有超过 2/3 的时间没有转化为决策资源。这是一个可以继续优化改进的地方。




**机会浪费** 随着搜索层数的增加以及估价的精细程度，Mr\_Curiosity 也暴露出一些奇怪的问题。比如在同时有多种获胜方式的情况下，它会选择  $\gamma$  值最大的那种决策，而  $\gamma$  并不能反应到达转态需要的步数，仅仅只能表达两条蛇局面的差距。所以在这种情况下 Mr\_Curiosity 会放弃一些几步就能绝杀的机会而选择一种更长的方式。但这不影响比赛结果。可是若是陷入对方最优决策下必输的处境时，它同样会选择  $\gamma$  值最大的那种决策。 $\gamma$  最大并不意味着可以延长存活局数，而是反映了局面差距。有时候在局面差距不大但是已经搜索到自己必输的处境中，Ai 会选择自杀来保证“死得最漂亮”，即比赛结束时双方的“局面差距”最小。

## 4 感谢

感谢 PPCA 大助教，尤其是高剑飞对这次大作业的辛勤付出；感谢羽绒、陆二和我对策略的讨论，使我们的 Ai 都得到了一定的提升；感谢一起写大作业在平台上对战的同学们，是你们让这次 Ai 设计又充满了欢乐。

Snake 的 Bot 排行榜						
排名	Bot 名	作者	排名分	Bot 描述	最新版本号	
1	baeTlink_Lick	 baeTlink	1482.64	萌萌哒	62	 source.cpp  ID
2	晨光映彩霞	 Vcmagnolite	1452.84	与光同行	1	 source.cpp  ID
3	虐待小动物	 grids	1447.72	Mapleyou	2	 source.cpp  ID
4	Snake_By_QY	 wzc1995	1418.38	一个字：弱	56	 source.cpp  ID
5	然而并没有什么卵用	 woooking	1417.38	然而并没有什么卵用	8	 source.cpp  ID
6	信科大傻_	 CF_EECS1stBigSB	1412.29	BigWaterMan	19	 source.cpp  ID
7	Ezreal	 ArsenHo	1383.58	是时候表演真正的技术了	38	 source.cpp  ID
8	Snake_Test	 Mike_Smith	1380.19	测试版本	20	 source.cpp  ID
9	长_者无敌	 pkutzy	1378.12	蛤	13	 source.cpp  ID
10	Mr_Curiosity	 o0_0o	1368.67	Is it true what they've ..	1	 source.cpp  ID

2015-8-3 20:58:57	Snake	已完成	 o0_0o <sup>2</sup> Dumb_Ways_To_Die	 o0_0o <sup>0</sup> Dumb_Ways_To_Die	 回放	0
-------------------	-------	-----	--	--	--	---

2015-8-19 20:04:34	Snake	已完成	 Mr_Curiosity <sup>2</sup>	 Mike_Smith <sup>0</sup> Snake_Test	 回放	0
--------------------	-------	-----	---	---	--	---

再见啦 PPCA.  
再见啦大作业.  
再见啦我的 Bots.  
再见啦 Mr. Curiosity.  
你们都好精彩.