AI 设计——双人贪吃蛇

柯嵩宇
上海交通大学

2015 年 8 月 18 日

Contents

1 游戏介绍

1.1 游戏规则

与传统单人贪吃蛇不同的是，本贪吃蛇为双人对战，每回合玩家同时做出决策控制自己的蛇。

玩家在 $n * m$ 的网格中操纵一条蛇 (蛇是一系列坐标构成的有限不重复有顺序的序列，序列中相邻坐标均相邻，即两坐标的 x 轴坐标或 y 轴坐标相同，序列中第一个坐标代表蛇头)，玩家只能控制蛇头的朝向 (东、南、西和北) 来控制蛇。蛇以恒定的速度前进 (前进即为序列头插入蛇头指向方向下一格坐标，并删除序列末尾坐标)。蛇的初始位置在网格中的左上角 (地图位置 $(1,1)$) 与右下角 (地图位置 (n,m))，初始长度为 1 格。与传统贪吃蛇不同，本游戏在网格中并没有豆子，但蛇会自动长大 (长大即为不删除序列末尾坐标的前进)，**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。**

地图为 $n * m$ 的网格，由 $1 * 1$ 的草地与障碍物构成。

蛇头在网格外、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。

其中 n 和 m 为随机参数，n 为 $[10,16]$ 间的整数，m 为 $[10,12]$ 间的整数，选手的程序应该对任意 n,m 均有效。

1.2 程序交互

程序收到初始化信息：

```
{
    "width": Number, // 地图的宽度
    "height": Number,  // 地图的高度
    "0":{"x":Number,"y":Number}, //0号玩家初始位置的x,y坐标
    "1":{"x":Number,"y":Number}, //1号玩家初始位置的x,y坐标
    "obstacle":[{"x":Number,"y":Number},{"x":Number,"y":Number}]
    //地图中障碍物位置的x,y坐标，注意，障碍物的数目不固定
}
```

每回合 bot 收到一个 JSON 对象，表示对方蛇的移动方向。具体格式如下：

```
{
    "direction": Number //
    0表示北(x-1,y)、1表示东(x,y+1)、2表示南(x+1,y)、3表示西(x,y-1)
}
```

bot 输出的回应也是一个 JSON 对象，格式同上，表示自己蛇的移动方向。[1]

2 AI 策略

2.1 经典贪吃蛇问题

考虑只有一只蛇的情况，且有苹果的情况：

[1]程序接收到的实际上是单行文本描述的的 JSON 对象，而为了增加可读性而格式化后的 JSON 对象。

保证贪吃蛇存活的基本策略是：在能到达尾部的前提下吃到食物。这样考虑的原因有二：一来贪吃蛇不至于盲目追逐食物将自己引进死路；二来尾部不断前进为蛇的移动腾出空间，因此追着尾部是一条安全的路径。基于这个策略，AI 优先保证蛇头到蛇尾的连通性，若前瞻到自己吃到食物后仍然可以到达尾部，那么就认为这条通往食物的路径是安全路径，否则就先追着尾部移动，伺机继续寻找通往食物的安全路径。同理，若出现某种蛇头无法到达食物（比如食物落在被一部分身体围起来的孔中），AI 也先追着尾部移动，等待能安全到达食物的时机。

“追逐尾部”这一策略可以保证贪吃蛇的“不死”，但是是否能吃到食物就另当别论了。[2]由于双人贪吃蛇游戏中不存在食物，追逐尾部就不失为一个很好的策略。

2.2 双人贪吃蛇问题

我的 AI 策略在尝试中不断的修改，大致有这几个策略：

2.2.1 当前局面估价

首先考虑，如果目前蛇头的位置比较和谐，无论是走什么方向都不会影响地图的连通性，那么随机走一个方向，否则选择选择估价值较高的方向前进，如果有多个相同估价值的方向则随机。

估价值定义为蛇走了这个方向可以存活的最长时间（即最长路的长度）[3]。

对于这个策略有一个很严重的问题，就是当三个方向都在一个联通块的时候会出现很多不确定的情况，但是实际上，相对较优的策略一般就只有一种。同时由于这个是基于当前局面的估价，因此，这个 AI 几乎没有大局观。

2.2.2 “追尾巴”

判断在目前情况下，蛇能不能通过移动来到达目前自己尾巴的位置，如果可以，那么从自己的尾巴开始进行 BFS，计算每个点到蛇尾巴的距离，然后选择和蛇头相邻的格子里距离最大的那个方向移动，如果有多个相同距离的话就随机选择一个方向。

如果自己的尾巴在当前情况下是不可见的，那么选择追逐对方的尾巴，如果仍不可见则试图以自己或者对方蛇的身体最早消失的那一段作为目标移动。

这个策略在实际使用的时候其实效果还是不错的，可以和对方长时间周旋[4]。当时对于一些有侵略意识的 AI 表现有些不佳。因为当对方开局直接把地图一分为二，然后开始蚕食我方的生存空间时，我的 AI 还会沉浸在追逐自己尾巴的乐趣中，直到陷入危机。

[2]有可能绕着尾巴跑了千万步吃不到东西

[3]而实际实现中，由于我不会写插头 DP 来求最长路，我就只好使用联通块大小来作为估价值（即，可能生存的最大步数）

[4]如果对方采用的是相似的生存型策略的话

2.2.3 枚举局面并估价

由于“追尾巴”的策略会让我的 AI 无法预知及时预知到危险然后突然陷入难以决策对策环境，紧接着突然死亡。为了增加 AI 全局观，我开始思考枚举局面并估价的策略。

双方的蛇各走 6 步，然后考虑 6 步之后的情况，选择一个平均估价值最高的方向前进。[5]。

但是由于运行时间的限制，搜索的步数也是有限的，因此，这个 AI 的大局观也是有限的，有的时候并不能做出正确的决断。[6]

2.2.4 “追尾巴”的改良（最终版本）

考虑到这个游戏的特点，生存是最大的目标，因此我决定放弃进攻性策略，思考如何利用有限的资源进行更长时间的生存。

提到“生存”，我还是想起了单人贪吃蛇的“追尾巴”的不死策略。如何让蛇在有限的空间中存活尽量的久，答案是——盘旋。同时“每次选择蛇尾到蛇头邻接的格子的最短路距离最长的方向前进”这一策略可以让蛇自然地盘旋起来。而且很容易证明对于任意一个格子，我们都可以这么干，这样就可以自然的找到最长的路，生存最久的时间。因此我决定改良“追尾巴”的策略，不再是但是为了能够让蛇不会因为追自己的尾巴而陷入危险，我把目标设定为我方蛇在当前状态下可以到达的对方蛇的最早消失的一段，然后使用**“最短路最长”**的策略来选择方向。

如果我方的蛇在当前局面下不能到达任何一个对方蛇的部分，那么 bot 将以当前情况下能够遍历到的最远 的格子为目标执行**“最短路最长”**的决策。

[5]估价的方法同“当前局面估价”

[6]而且由于实现的时候出现了一些 BUG，在极少数的时候会选择错误的方向前进。