

Quoridor 步步为营

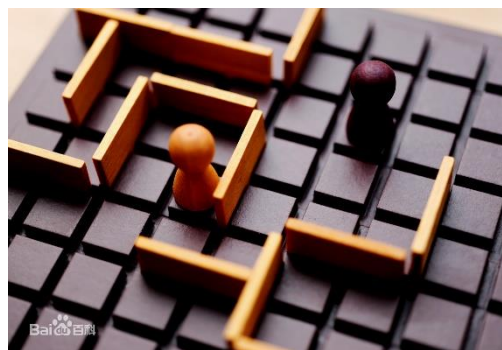
用户手册

计算机科学与技术系

191220008 陈南瞳

目录：

- 一、 产品简介
- 二、 设计思路
- 三、 功能说明
- 四、 策略说明
- 五、 预期效果



一、产品简介

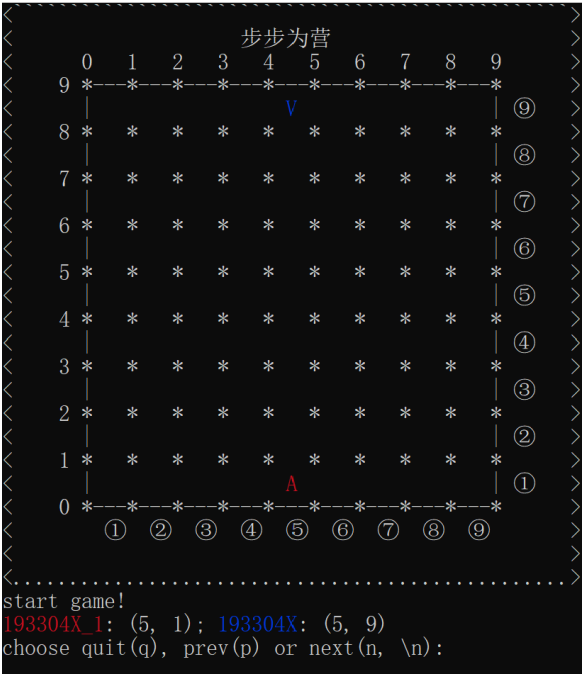
1、 Quoridor

Quoridor（步步为营）是个规则相当简单的游戏，可以两人对战，游戏的目标就是让自己的小人走到对面。当然事情是不会这么一帆风顺的，每个玩家手上都有阻挡去路的木板，每个回合你既可以选择移动自己的小人格，也可以选择摆放一块木板。

2、 AI 走棋

在 Quoridor 现有的游戏框架下, 设计出一个能够针对不同的布局采用一定行棋策略进行自动走棋的 AI 玩家, 并要求尽可能地击败其他 AI 玩家, 获取最终的胜利!

3、 UI 界面



二、设计思路

1、 形式化定义

棋盘：9 × 9 方格

棋子：

小人：1 个/玩家, 开始时在棋盘分别在棋盘两边的中间位置。

胜利条件：己方小人到达对方底边。

- (1) 一次仅允许移动一步或放置一块木板，否则视为违规操作。
- (2) 面前是对方小人时，若其后没有木板，则移动到对方背后，否则视为违规操作。
- (3) 若 10 块木板用尽，则只允许进行移动操作，否则视为违规操作。
- (4) 木板长度为 2，且仅能竖直或水平放置，否则视为违规操作。
- (5) 放置的木板不允许将对方的所有获胜路径堵死。
- (6) 当小人位于对方的唯一路径上时，只允许移动，不允许放置木板，否则视为违规操作。
- (7) 每次必须进行操作，不能无操作，否则视为违规操作。
- (8) 所有操作不允许超出棋盘边界，否则视为违规操作。
- (9) 违规操作 3 次则直接判负。
- (10) 单次决策超过 1s 视为超时，直接判负。

```
graph TD; A[从服务器接收新的棋盘信息] --> B[存储敌方坐标或新放置的木板]; B --> C[通过BFS搜索出当前最佳的移动方向和放置的木板]; C --> D[用评估函数对两种操作进行评分]; D --> E[重置当前数据便于接收下一次棋盘信息]; E --> F[将下一步操作返回给服务器]; F --> G[等待对手下一步操作]; G --> H[从服务器接收新的棋盘信息]; H --> A;
```

从服务器接收新的棋盘信息

存储敌方坐标或新放置的木板

通过BFS搜索出当前最佳的移动方向和放置的木板

用评估函数对两种操作进行评分

重置当前数据便于接收下一次棋盘信息

将下一步操作返回给服务器

等待对手下一步操作

从服务器接收新的棋盘信息

三、功能说明

1、 变量说明

int targetY = 0; // 目标方向

QuoridorUtils::Location min_Loc; // 最佳移动的坐标

QuoridorUtils::BlockBar min_block_set; // 最佳的放置木板的坐标

QuoridorUtils::Location next_Loc; // 下一步移动的坐标

QuoridorUtils::BlockBar next_block_set; // 下一步放置木板的坐标

int path_status = -1; // 是否移动

int block_status = -1; // 是否放置木板

int my_path_min_step_number = 0; // 己方最短的获胜步数

int enemy_path_min_step_number = 0; // 敌方最短的获胜步数

int my_block_min_step_number = 0; // 放置木板后己方最短的获胜步数

int enemy_block_min_step_number = 0; // 放置木板后敌方最短的获胜步数

std::vector<QuoridorUtils::BlockBar> blocks; // 存储所有的木板

int head = 0, tail = 0; // 节点队列的首尾下标

Node queue[100], path[10000]; // 节点和路径的结构

int board[9][9] = { 0 }; // 棋盘数组，用于标记寻路时到过的位置

int my_block_left = 10; // 己方剩余木板数

int enemy_block_left = 10; // 敌方剩余木板数

```
int illegal = 0; // 是否犯规
```

```
int my_illegal_times = 0; // 犯规次数
```

2、 函数说明

```
void my_strategy(const QuoridorUtils::Location& myLoc, const QuoridorUtils::Location& enemyLoc);
```

```
//总走棋策略函数
```

```
void direction_check(bool directions[], const QuoridorUtils::Location& myLoc, const QuoridorUtils::Location& enemyLoc);
```

```
// 判断移动是否合法
```

```
int path_seek(const QuoridorUtils::Location& myLoc, const QuoridorUtils::Location& enemyLoc);
```

```
// 寻找最短路径
```

```
bool block_check(const QuoridorUtils::BlockBar& block_set, const QuoridorUtils::Location& myLoc, const QuoridorUtils::Location& enemyLoc);
```

```
// 判断放置的木板是否合法
```

```
int block_seek(const QuoridorUtils::Location& myLoc, const QuoridorUtils::Location& enemyLoc);
```

```
// 寻找最佳的放置的木板
```

```
void queue_initialize();
```

```
// 节点队列初始化
```

```
void board_initialize();
```

```
// 涂图棋盘初始化
```

```
Node* pop();
```

```
// 弹出队列
```

```
void push(QuoridorUtils::Location Loc, Node* next);
```

```
// 加入队列
```

```
double evaluation(double my_path_number, double my_block_number, double enemy_path_number, double enemy_block_number);
```

```
// 评估函数（已废弃，现使用另一个）
```

3、 其他功能

(1) 计时

计算单次决策所用的时长，单位为秒（如：0.014s），使用 clock 函数，便于检查是否超时。

四、策略说明

1、移动方向优先性

绝大多数人,在最短步数相同时,会优先选择上下移动,因此我选择在最短步数相同时,优先左右移动,可能会一定程度打乱对手的决策效果。

2、放板方向随机性

在放板后的收益相同时,随机选择不同方向的木板,这可能导致第一局和第三局的结果不同,不过这给比赛带来了更多的未知性,而并非完全固定。

3、堵路处理

经过大量实战后发现:当己方的小人位于对方唯一获胜路径上时,下一步不允许放置木板,只允许移动。

针对这种情况,当出现犯规时,记录下来,默认下一步不放置木板,只移动。

或者犯规次数达到两次时,默认下一步不放置木板,只移动。

但缺点明显,会白白浪费犯规次数和移动的机会,并且犯规次数达到两次后不能放置木板将很难获胜。在现有的框架下很难实现直接避免堵路或更好的处理方式,因此属于可提升的空间。

4、评估函数参数设置

最初设想时通过博弈树、 α - β 剪枝、minimax、蒙特卡洛、A*中的方法来实现,但最终

并未采用。一是因为尽管看了许多资料，但对其了解仍然不够深，想要用代码实现，可能时间不允许；二是因为通过了解和自己大量的实践，发现决定胜负的关键因素其实在于评估函数，使自己的贪心算法有智慧地贪心，可以让自己的 AI 看起来更“智能”。

我设立了两个评估函数 (score_1 和 socre_2)，一个适用于对移动的评估，一个适用于对木板放置的评估。以及针对特定情况的加分处理。

经过上千次的参数调整和打翻重构形式，最终选择出表现较好的参数和公式形式。

```
double score_1 = 0;
double score_2 = 0;
score_1 = 22.0 * (double)enemy_path_min_step_number + 1.0 * (double)enemy_block_left - 5.0 * (double)my_path_min_step_number - 3.0 * (double)my_block_left;
score_2 = 2.0 * (double)enemy_block_min_step_number - 1.0 * (double)enemy_block_left - 5.0 * (double)my_block_min_step_number + 4.0 * ((double)my_block_left - 1)
    + 40.0 * ((double)enemy_block_min_step_number - (double)enemy_path_min_step_number);
if (my_block_min_step_number - my_path_min_step_number > enemy_block_min_step_number - enemy_path_min_step_number)
    score_1 = score_1 + 50;
if (my_path_min_step_number < 4 && my_path_min_step_number < enemy_path_min_step_number)
    score_1 = score_1 + 10000;
if (enemy_path_min_step_number < 4 && my_path_min_step_number >= enemy_path_min_step_number)
    score_2 = score_2 + 10000;
if (my_path_min_step_number > enemy_path_min_step_number + 1)
    score_2 = score_2 + 10;
else if (my_path_min_step_number > enemy_path_min_step_number)
    score_2 = score_2 + 15;
if (my_path_min_step_number > enemy_path_min_step_number + 3 && my_block_min_step_number - my_path_min_step_number <= enemy_block_min_step_number - enemy_path_min_step_number)
    score_2 = score_2 + 25;
if (my_block_min_step_number - my_path_min_step_number + 1 < enemy_block_min_step_number - enemy_path_min_step_number)
    score_2 = score_2 + 35;
if (my_block_min_step_number - my_path_min_step_number <= enemy_block_min_step_number - enemy_path_min_step_number && my_path_min_step_number > enemy_path_min_step_number + 2)
    score_2 = score_2 + 50;
```

并保留了表现最好的一些参数，部分列举如下：

评估函数 - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

评估式:

```
if(my_path_min_step_number > 0 && ((double)pow(my_path_min_step_number,9) <= 2 * (double)pow(enemy_block_min_step_number, my_block_left * 0.7)
    || my_block_min_step_number - my_path_min_step_number >= enemy_block_min_step_number - enemy_path_min_step_number)
    && enemy_path_min_step_number > 3)

3
score_1 = 12.7 * (double)enemy_path_min_step_number + 2.2 * (double)enemy_block_left - 5.7 * (double)my_path_min_step_number - 3.0 * (double)my_block_left;
score_2 = 7.7 * (double)enemy_block_min_step_number - 2.2 * (double)enemy_block_left - 5.7 * (double)my_block_min_step_number + 4.0 * ((double)my_block_left - 1)

1
score_1 = 16.7 * (double)enemy_path_min_step_number + 1.2 * (double)enemy_block_left - 5.7 * (double)my_path_min_step_number - 3.0 * (double)my_block_left;
score_2 = 7.7 * (double)enemy_block_min_step_number - 2.2 * (double)enemy_block_left - 5.7 * (double)my_block_min_step_number + 4.0 * ((double)my_block_left - 1)

2
score_1 = 18.7 * (double)enemy_path_min_step_number + 1.2 * (double)enemy_block_left - 5.7 * (double)my_path_min_step_number - 3.0 * (double)my_block_left;
score_2 = 7.7 * (double)enemy_block_min_step_number - 2.2 * (double)enemy_block_left - 5.7 * (double)my_block_min_step_number + 4.0 * ((double)my_block_left - 1)

1.5
score_1 = 16.2 * (double)enemy_path_min_step_number + 1.2 * (double)enemy_block_left - 5.7 * (double)my_path_min_step_number - 3.0 * (double)my_block_left;
score_2 = 8.9 * (double)enemy_block_min_step_number - 1.2 * (double)enemy_block_left - 5.7 * (double)my_block_min_step_number + 4.0 * ((double)my_block_left - 1)

0.9
score_1 = 17.2 * (double)enemy_path_min_step_number + 1.2 * (double)enemy_block_left - 5.7 * (double)my_path_min_step_number - 3.0 * (double)my_block_left;
score_2 = 9.9 * (double)enemy_block_min_step_number - 1.2 * (double)enemy_block_left - 5.7 * (double)my_block_min_step_number + 4.0 * ((double)my_block_left - 1)

1.4
score_1 = 30.7 * (double)enemy_path_min_step_number + 0.7 * (double)enemy_block_left - 5.7 * (double)my_path_min_step_number - 3.0 * (double)my_block_left;
score_2 = 8.0 * (double)enemy_block_min_step_number - 0.7 * (double)enemy_block_left - 2.7 * (double)my_block_min_step_number + 4.0 * ((double)my_block_left - 1)
if (my_block_min_step_number - my_path_min_step_number > enemy_block_min_step_number - enemy_path_min_step_number)
    score_1 = score_1 + 100;
if (my_path_min_step_number < 3 && my_path_min_step_number < enemy_path_min_step_number)
    score_1 = score_1 + 10000;
if (my_path_min_step_number > enemy_path_min_step_number + 1)
```

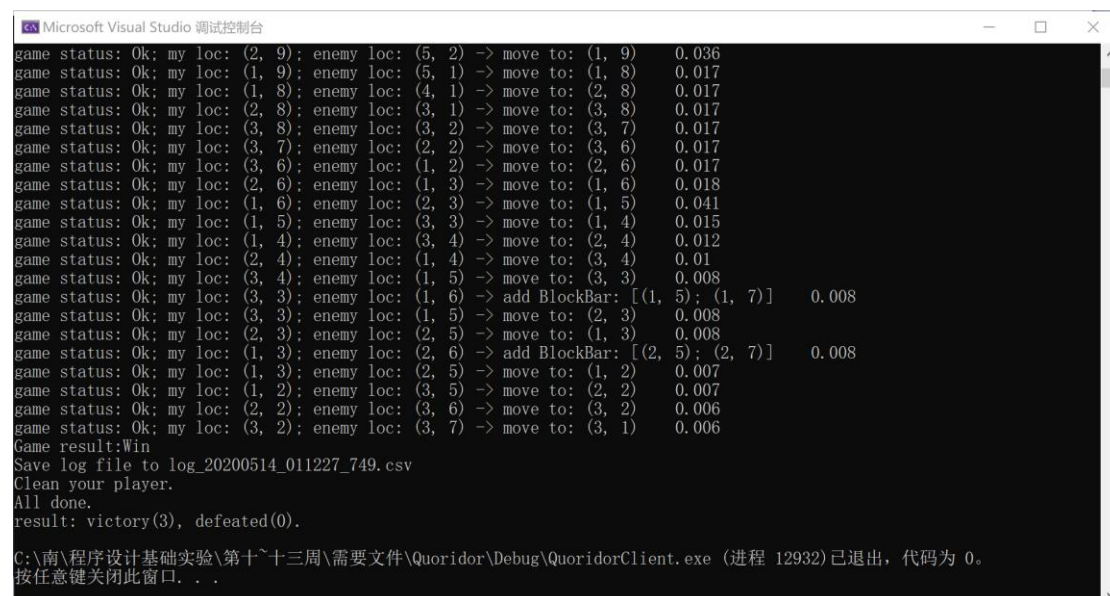
第 127 行, 第 56 列 100% Windows (CRLF) UTF-8

但经过大量的实践后，发现也有许多只靠参数解决不了的问题：比如一些变量可能会突变，导致优劣势直接颠倒。可以通过对未来 2~3 步一起进行评估来预判这种情况的发生，这属于可提升的空间。

五、预期效果

虽然发现运气占比较大，但仍可通过策略将获胜概率一定幅度地提高。

所以预期效果即为尽可能战胜更多的 AI。



```
Microsoft Visual Studio 调试控制台
game status: Ok; my loc: (2, 9); enemy loc: (5, 2) -> move to: (1, 9) 0.036
game status: Ok; my loc: (1, 9); enemy loc: (5, 1) -> move to: (1, 8) 0.017
game status: Ok; my loc: (1, 8); enemy loc: (4, 1) -> move to: (2, 8) 0.017
game status: Ok; my loc: (2, 8); enemy loc: (3, 1) -> move to: (3, 8) 0.017
game status: Ok; my loc: (3, 8); enemy loc: (3, 2) -> move to: (3, 7) 0.017
game status: Ok; my loc: (3, 7); enemy loc: (2, 2) -> move to: (3, 6) 0.017
game status: Ok; my loc: (3, 6); enemy loc: (1, 2) -> move to: (2, 6) 0.017
game status: Ok; my loc: (2, 6); enemy loc: (1, 3) -> move to: (1, 6) 0.018
game status: Ok; my loc: (1, 6); enemy loc: (2, 3) -> move to: (1, 5) 0.041
game status: Ok; my loc: (1, 5); enemy loc: (3, 3) -> move to: (1, 4) 0.015
game status: Ok; my loc: (1, 4); enemy loc: (3, 4) -> move to: (2, 4) 0.012
game status: Ok; my loc: (2, 4); enemy loc: (1, 4) -> move to: (3, 4) 0.01
game status: Ok; my loc: (3, 4); enemy loc: (1, 5) -> move to: (3, 3) 0.008
game status: Ok; my loc: (3, 3); enemy loc: (1, 6) -> add BlockBar: [(1, 5); (1, 7)] 0.008
game status: Ok; my loc: (3, 3); enemy loc: (1, 5) -> move to: (2, 3) 0.008
game status: Ok; my loc: (2, 3); enemy loc: (2, 5) -> move to: (1, 3) 0.008
game status: Ok; my loc: (1, 3); enemy loc: (2, 6) -> add BlockBar: [(2, 5); (2, 7)] 0.008
game status: Ok; my loc: (1, 3); enemy loc: (2, 5) -> move to: (1, 2) 0.007
game status: Ok; my loc: (1, 2); enemy loc: (3, 5) -> move to: (2, 2) 0.007
game status: Ok; my loc: (2, 2); enemy loc: (3, 6) -> move to: (3, 2) 0.006
game status: Ok; my loc: (3, 2); enemy loc: (3, 7) -> move to: (3, 1) 0.006
Game result: Win
Save log file to log_20200514_011227_749.csv
Clean your player.
All done.
result: victory(3), defeated(0).

C:\南\程序设计基础实验\第十~十三周\需要文件\Quoridor\Debug\QuoridorClient.exe (进程 12932) 已退出，代码为 0。
按任意键关闭此窗口。 . . .
```



```
C:\南\程序设计基础实验\第十~十三周\需要文件\Proj3_server_1.051>java -jar server.jar -p 19330 -n 2 -r 3
Server Version: 1.1
Args: [-p, 19330, -n, 2, -r, 3]

-----
Waiter is waiting for the players to sit down.
Waiter's work is done.
Contest start at: 2020-05-14 01:12:24
-----
*****
*****
Knockout Round 1 :
*****
[3 : 0] 191220008 defeat-->
-----
CHAMPION: QuoridorPlayer {name='191220008'}
-----
Contest end at: 2020-05-14 01:12:27
```