黑白棋 AI

目录

Project Introduction

1.1 开发环境与系统要求

1.1.1 开发环境：

Win10

1.1.2 所用的工具：

本小组后端的 MCTS 分别用 Java 和 C++都实现了一遍，
Java 使用了 IDEA 开发工具
C++使用 VS2013 开发工具
前端 UI 使用了 Sublime+Chrome

1.1.3 开源库：

C++使用了读写 Json 文件的 RapidJson 第三方库

1.1.4 系统运行要求：

前端 UI 运行要求：Chrome 浏览器
后端服务器 C++： VS2013+nodejs+express
Java：目前没有与 UI 结合起来，直接运行即可，是命令行输入和显示

1.2 工作分配简介

Technical Details

2.1 理论知识阐述

2.1.1 MCTS 算法

MCTS 算法是本实验中使用的基础算法，MCTS 是结合了 Monte-Carlo simulation 和 game tree search。Monte-Carlo simulation 即不断随机选择抉择，得到结果后将值返回，而 game tree search 中的 game tree 的每一个节点代表游戏的状态，AI 不断扩展 game tree，最后选择最好的 action。

MCTS 简而言之就是用 Monte-Carlo simulation 去扩展 game tree，首先会优先选择还没有被选中的结点，如果下一层结点已经全部被访问过了，则从里面选出一个 BestChild，评估函数会结合结点选中的次数和最后的总收获做一个平衡。然后继续往下做选择策略，如果所有都是被访问过的，那么直接返回结果，并上升反馈。

当选择一个结点扩展的时候，便会不断随机做决策，直到结束状态，返回 reward，然后将 reward 从被选中扩展的节点逐级上升反馈，注意在本实验中，由于是有 2 个选手身份的，所以 reward 要根据选手身份和结束状态适时变化。来选择一个对自己当前状态最有力的。

然后在一定的时间范围内，不断的重复操作，到最后选择出一个针对根结点最优的动作。

MCTS 本质上是用大量的 simulation 去逐级主动学习出一个最优策略，在学习的过程将访问次数和最终 reward 来综合评估。

2.1.2 散度策略

黑白棋中，在下一步子的时候并不是你吃点的子越多越好，因为这些子待会可以被吃回来，评价一个落子的优劣的标准是下完一步子后，自己可落子的位置减少的数量，对方可落子的位置减少的数量。一个好的落子，应该是让自己可落子的位置越来越多，对方可落子的位置越来越少，这样，到最后，对手可以走的位置越来越少，只能走一些对自己不利的棋。

而在黑白棋中，我们用散度来评估这个数据，散度就是吃掉子周围的空格数。在落子的时候我们需要选择散度小的值。

我们可以看到下方的棋局，现在是绿色的选手下（代表黑棋），我们利用散度策略来评估 A 和 B 的 2 个位置，若下 A, 则 1 会变成黑棋，而 1 的周围有 6 个空格，散度为 6；而若下 B，则 2，3 都会变成黑棋，2，3 的散度为 5，更小。所以根据散度理论，B 的位置比 A 更好。

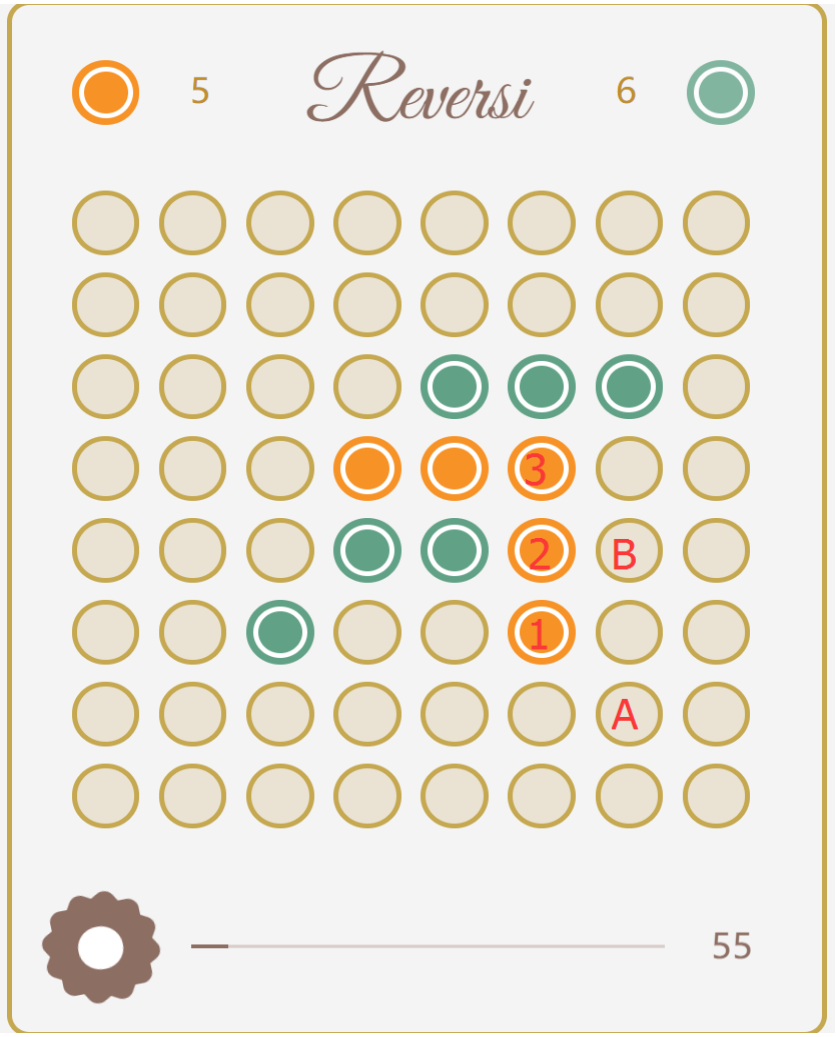

Figure 1 散度计算

2.1.3 棋盘位置的价值理论

在黑白棋中，有一句术语“金角银边草肚皮”，棋盘上每一个子都会对应的价值，其中四边的四个角是很重要的，因为一旦抢占了那个子，对手是不能再把你吃掉的，同时你可以依据这个不断的扩展。自然，能够导致对手抢占到角的星位和靠近角的边的价值就很低。并且边也比较重要，相对来说，边比较稳定。

棋盘上每一个子都有对应的价值，我们为棋盘上 64 个位置都训练生成了对应的价值。

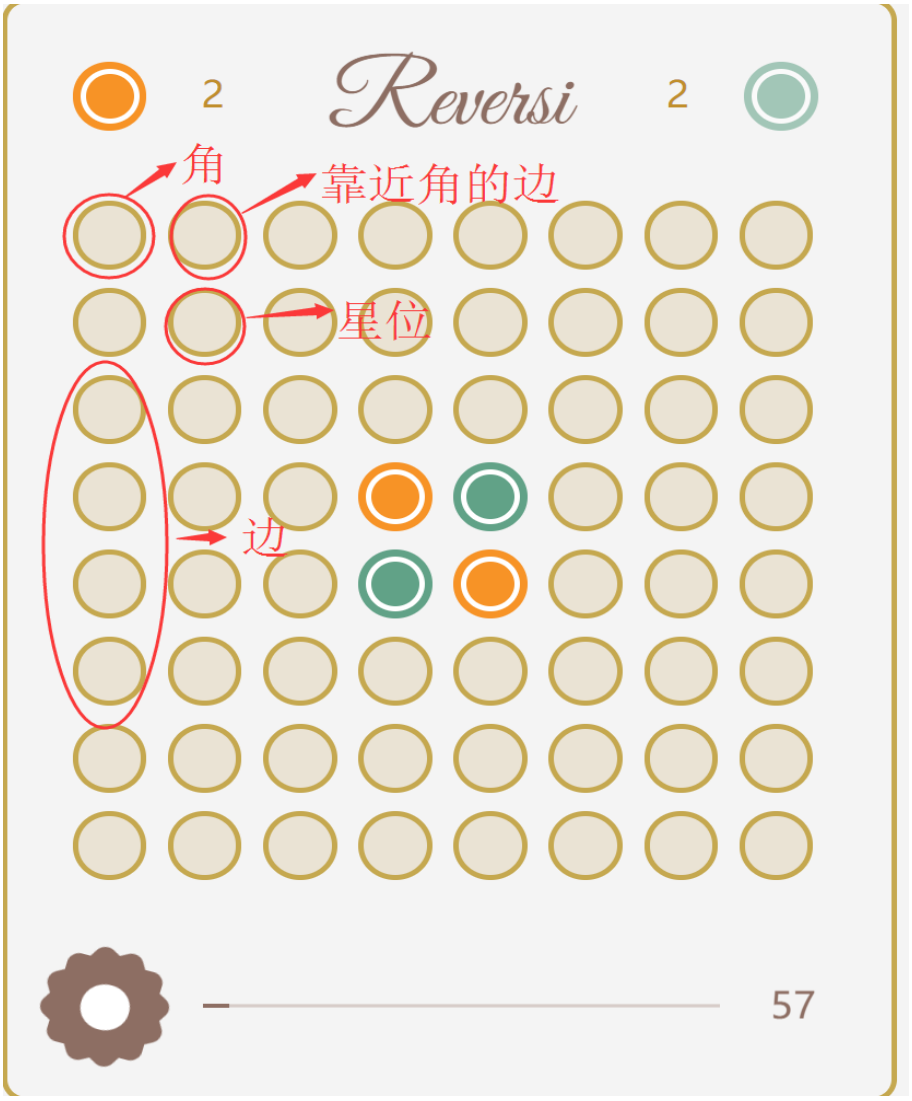

Figure 2 位置价值

2.1.4 中盘之后策略变化因素

散度理论和棋盘位置的价值理论在中盘之前比较重要，因为黑白棋的开局十分关键，同时开局采用对应经典的策略往往可以避免犯一些错误同时尽可能的抢占优势，而在中盘以后，很多策略选择的位置并非是最好的，可落子的地方降低，我们利用 MCTS 模拟计算往往可以准确的计算出所有的结果，锁定胜局。

所以在中盘以后，我们降低散度理论和位置价值的权重，增加 MCTS 的权重。

2.1.5 不同语言执行速度

C++的运行速度普遍上要比 Java 快上 2-3 倍，而由于我们使用了 MCTS，而对 MCTS 而言计算性能是非常重要的，在起初我们使用 Java 实现了一遍，后来为了提升性能，我们实现了 C++版本的 AI。

2.1.6 前后端基本知识

为了性能，最终的 AI 我们是采用 C++, 而 UI 使用了 js，采用了 nodejs 做服务器。为了 2 者能够正常交互通信，我们花费了很多时间，尝试使用了很多的方法。在这中间，我们需要注意的是以下几个点。

首先是前端没有权限修改文件，任何修改文件的操作都应在 nodejs 后端进行；任何程序都可以通过对同一份文件的读写来进行通信，在本工程中，C++与 nodejs 即通过对相同的文件的读写来完成通信；js 与 nodejs 中的 socket 是单线程的，并且 2 者的资源发送是异步加载的，在具体前后端实现中最好将任务分割，可以采用类似握手协议，逐步逐步完成任务。

2.2 算法描述

2.2.1 整体构架

整体构架分成三个部分，分别是 UI，Node 服务器和 C++AI 程序。用户在 AI 上输入，将输入传递给 Node 服务器，而 Node 服务器又与 C++AI 程序通信，获得当前 AI 落子和棋盘情况等，再将这些数据传递 UI 显示给用户，其中服务器与 AI 程序的通信采用对同一 json 文件的读写来做

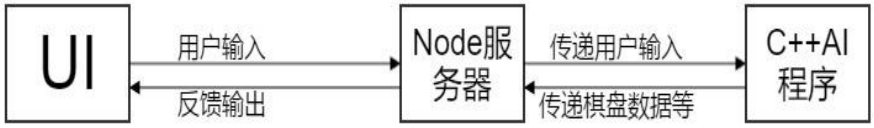

Figure 3 整体构架

2.2.2 C++AI 程序

我们以黑棋作为 AI 为例，在程序运行的初始阶段，首先会生成 2 个初始化文件（SendToUI.json 和 FromUI.json）用来初始化通信，接着 AI 会根据自己的 AI 策略进行选择落子，然后将棋盘更新，写入 SendToUI.json 文件，然后等待 UI 传递落子数据给客户端，并且更新棋盘给 UI。

其中，所有的数据交互都是通过采用了对 SendToUI.json 和 FromUI.json 文件的读写来进行的。AI 策略结合了 MCTS，散度理论和位置价值函数。具体的执行过程在后续会有简单介绍。

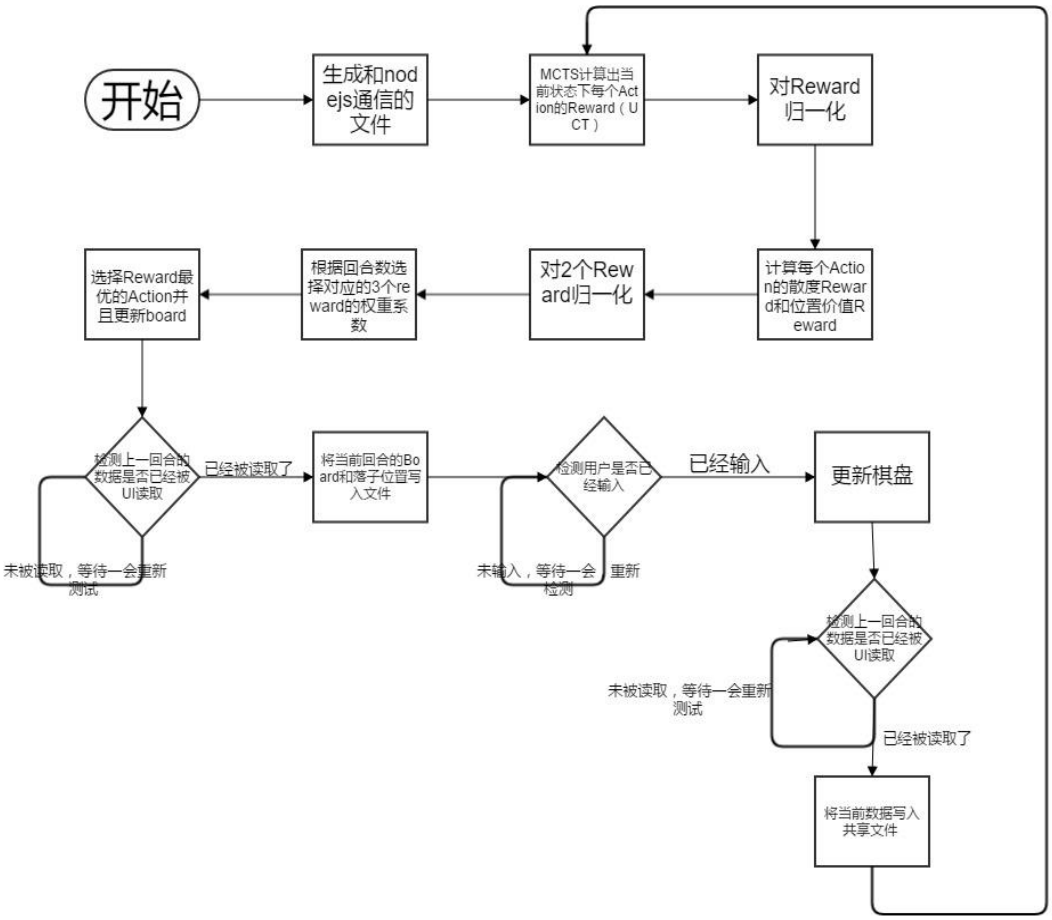

Figure 4 AI 流程

2.2.3 AI 选择策略（结合 MCTS，散度和价值网络）

AI 选择的策略主要分成 3 个部分考虑，首先我们利用 MCTS 进行模拟得到当前结点下的每一个 Action 的 UCT reward，然后进行归一化操作，并且对每一个 Action 计算他们的散度值和位置价值（根据预先训练出的数据），然后根据回合数，选择对应权重来计算出最后的 reward，最后选择 reward 最高的 action 输出。

我们采用下列公式来评估一个 Action,每次我们都会选择 Reward 最大的 Action

$$Reward = w_1 * Reward_{MCTS} + w_2 * Reward_{Divergence} + w_3 * Reward_{PostionValue}$$

而

$Reward_{MCTS} = Q(V')/N(V') + C * \sqrt{2 * ln(N(v))/N(v')}$，其中 v'是 V 的子结点，Q（V）是 V simulation 的返回值，N（V）是其被访问的次数

而$Reward_{Divergence}$是被翻转的棋子的相邻空格数之和，而$Reward_{PostionValue}$，w_1, w_2, w_3是最初训练得到的，并且都会随着回合数的变化而变化

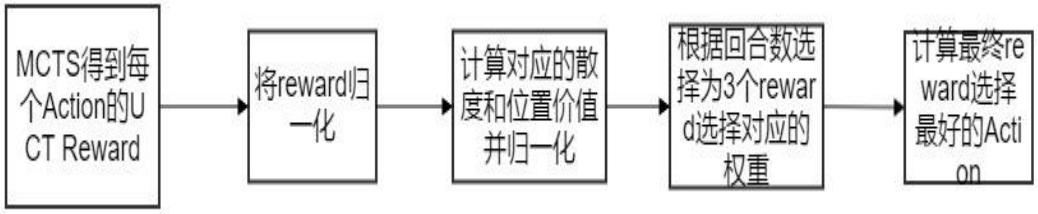

Figure 5 AI 选择策略

2.2.4 MCTS 流程

伪代码介绍（主要参照 ppt 上的算法）：
其中对 child 的评估采用了 UCT 策略
我们选择 children 结点中让 Q(v')/N(v')+c*sqrt(2*ln(N(v)))/N(v')最大的结点

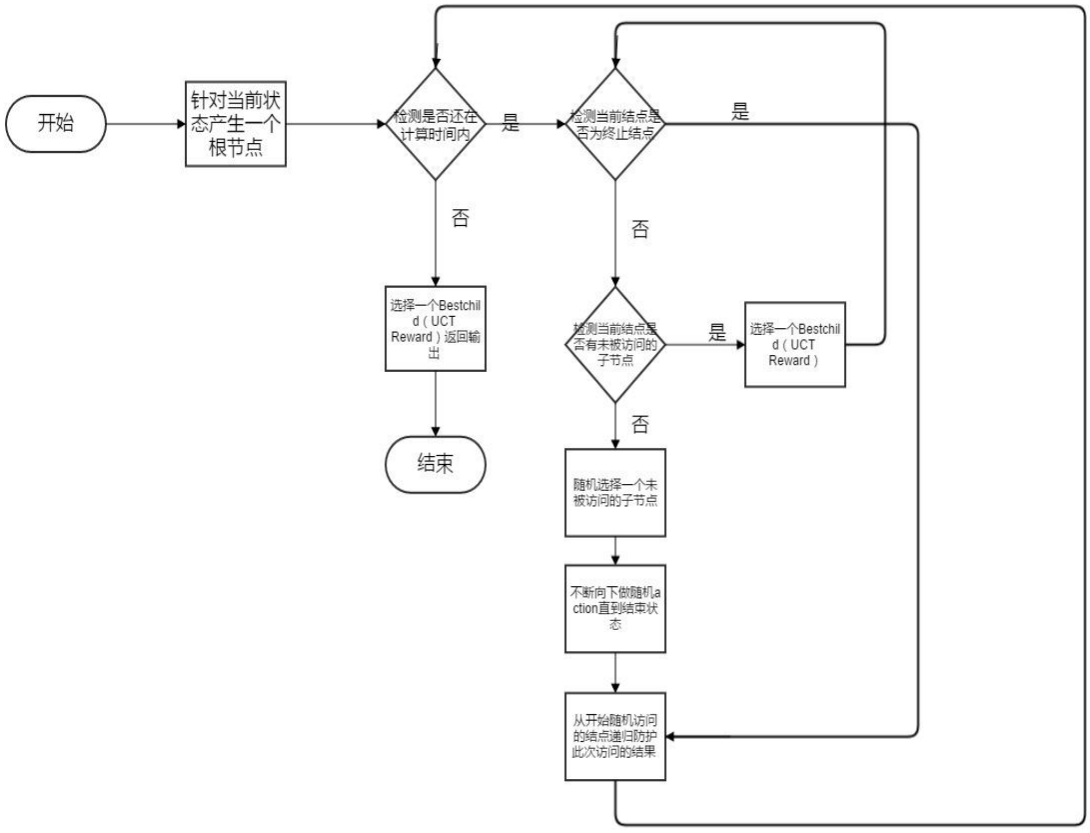

Figure 6 MCTS 流程

2.2.5 散度 reward 和位置价值计算

我们为每个 action 计算他们翻转棋子的空给数得到散度，然后根据回合数，选择原先训练的位置价值网络，得到每个 action 的价值

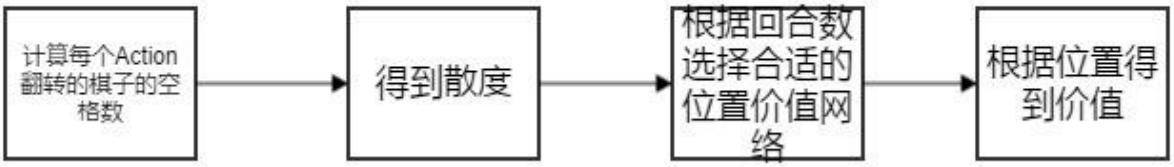

Figure 7 散度和位置价值计算

2.2.6 参数训练模型

在前面的策略决策中，有很多参数是需要我们训练的，我们初始设置参数，采用不同参数的 AI 对战。我们选择胜利次数多的参数来更新我们的策略，然后不断重复过程，最后得到我们的最终参数设置

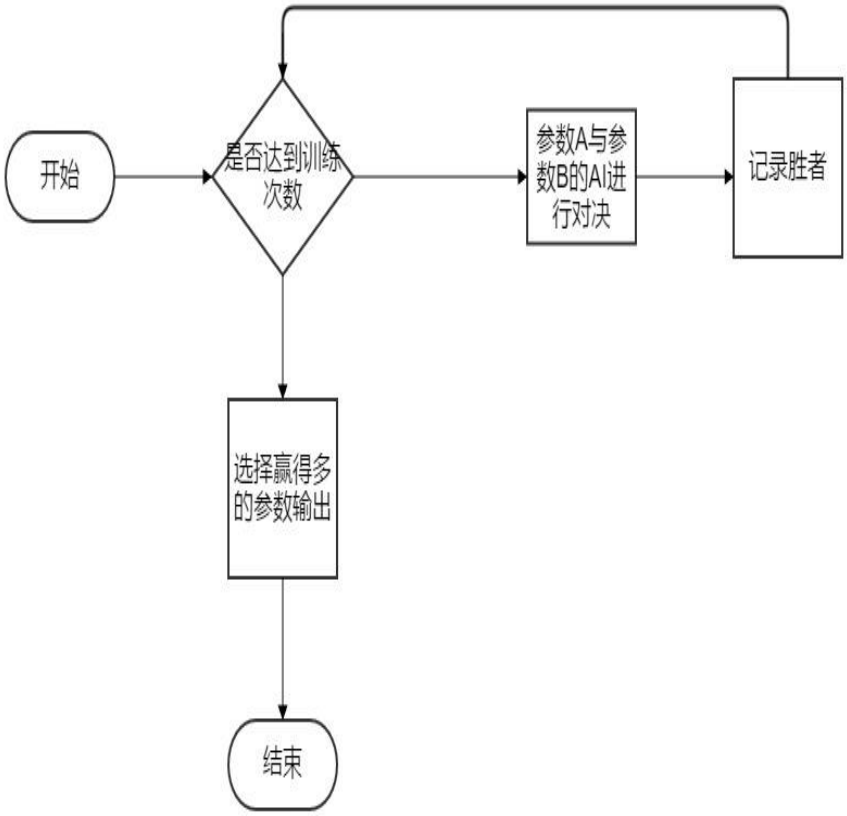

Figure 8 训练模型

2.2.7 C++与服务器交互

C++与服务器的交互采用 2 者读写同一的文件来进行通信。在本 AI 中将通信数据存放在 data 文件中，分别为 FromUI.json 文件和 SendUI 文件，其中 C++对 json 文件的读写使用了一个 RapidJson 的第三方库。

2.2.8 前后端交互

前端只负责显示，所有对数据文件的操作和读写都是在服务器进行的，服务器和 C++ Ai 的通信是通过对相同文件的读写来进行的

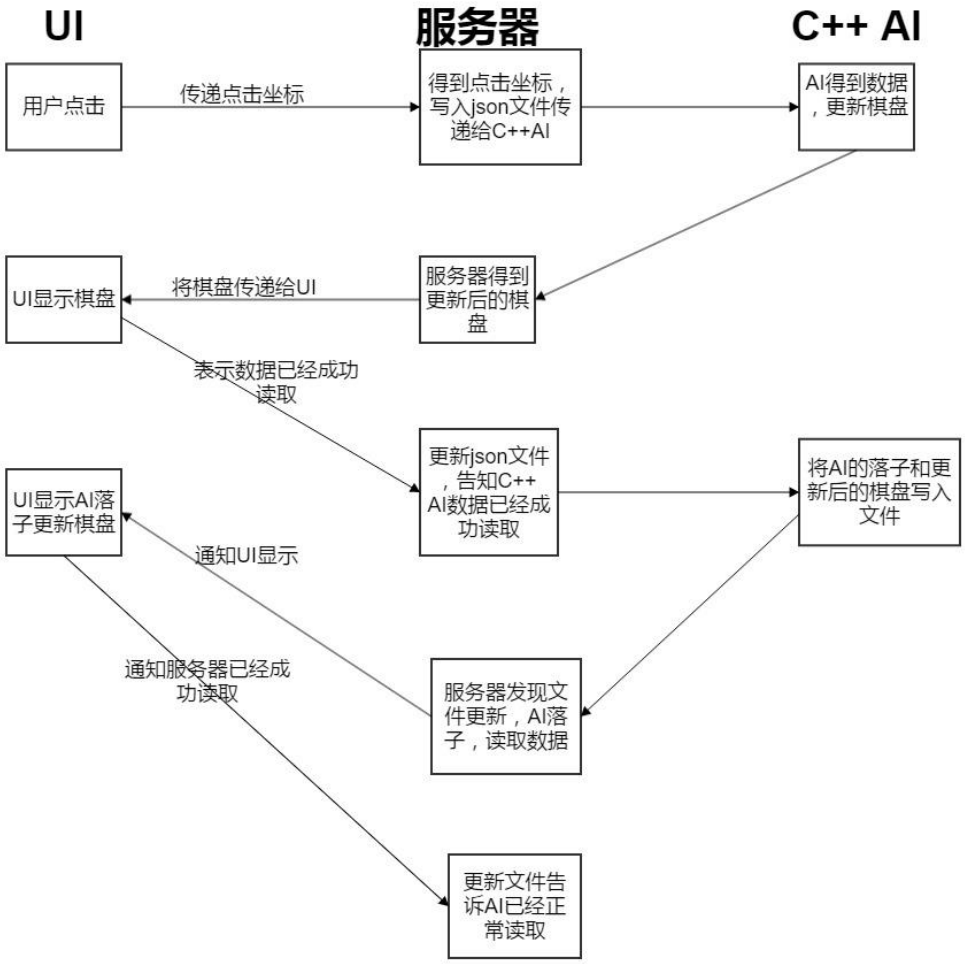

Figure 9 前后端交互

2.3 技术细节描述

在本工程中，所有函数，除了 C++读写 json 文件使用了第三方库，其他所有的函数都是自己独立实现的。
需要注意的技术细节：

2.3.1 AI 的 MCTS 搜索

实现了 MCTS 的搜索，具体的算法过程在上一部分已经描述。由于 MCTS 对性能要求比较高，所以在实现过程中需要特别注意，进行一些底层优化。

首先，起初的时候，采用了 set 来存储数据，但是这对性能影响以较大，后来将其修改为 vector 来存储数据。

接着，在 MCTS 的中，我们需要随机选择一个没有被访问的位置来扩展，我们将可供选择的数据都是存放在 vector 中，我们采用的方法是随机选择一个位置，将他读取以后，再将其和最后一个元素进行交换，然后再将最后一个元素 pop 出去来实现随机选择一个元素后并

将其清除的功能。

在 MCTS 实现中需要特别注意的是 2 个函数，分别是每一步落子后的更新重置和一次 simulation 结束后的反馈。

```
void Reversi::reset_location(int type)
```

该函数实现的功能是在每一次落子结束后，即时的更新下一个选手可以下的位置。我们采用了遍历的方法，对 64 个位置每一个位置进行检测，其中在正式检测之前，我们会先简单判断一下，比如是否可以落子，以及四周是否有字进行剪枝处理。当都符合情况的时候，我们会对其 8 个方向进行逐一检测，只需其中一个方向是合法的，便将其加入我们的结果中。

这个函数其实是非常重要的，在进行性能优化的时候，我们发现大部分的运算时间都花在这个函数上面，因为每次 simulation 的落子都需要调用这个函数重置可以落子的地方。

```
void BP(Node* v, int winner);
```

该函数实现的功能是即时更新结点的值

需要特别注意的是游戏结束的时候，我们需要为向上为结点更新值，而在这个游戏是有 2 个游戏身份的，我们记录了最后的游戏结果和当前结点的身份，如果是胜利，便加 1，如果失败的话，便减 1.

2.3.2 前后端交互

前后端交互中 socket 通信是单线程的，并且数据是异步加载的。所以我们采用的类似握手协议反复通信确认，具体涉及到的函数有

```
getBoardFromBackEnd(Data)
UpdateToUI(Data)
AskAINextRoud(Data)
AckPos(Data)
AskUserBoard(Data)
AskAIBoard(Data)
UpdateUserBoard(Data)
UpdateAIBoard(Data)
```

这些函数构成了一个通信模型。具体的流程为用户点击了界面，UI 会将数据发送给后端，后端得到数据，将数据写入 json 文件，然后待用 Ackpos(Data)告诉 UI 已经成功得到数据，然后 UI 调用 AskUserBoard(Data)向客户端请求落子后更新的棋盘，接着服务器调用 getBoardFromBackEnd(Data)，从 json 文件中得到 C++写入的数据，并将数据传递给 UI，UI 调用更新棋盘，并且调用 UpdateToUI(Data)告诉服务器已经成功显示，服务器更改 json 文件中的内容来告知 C++ AI 文件已经被读取，可以正常写入了。

接着服务器调用 AskAINextRoud(Data)，通知 UI 可以来向服务器发起 AskAIBoard(Data)请求，要求得到 AI 的落子。然后后 AI 不断读取 json 文件的内容，直到文件内容被 C++ AI 更新，得到 AI 的落子和棋盘，告知 UI，UI 得到数据后正常显示，告知服务器已经显示成功，服务器再次修改 json 文件表示已经正常读取，一次用户和 AI 的落子便结束了。

Experiment Results

3.1 系统界面

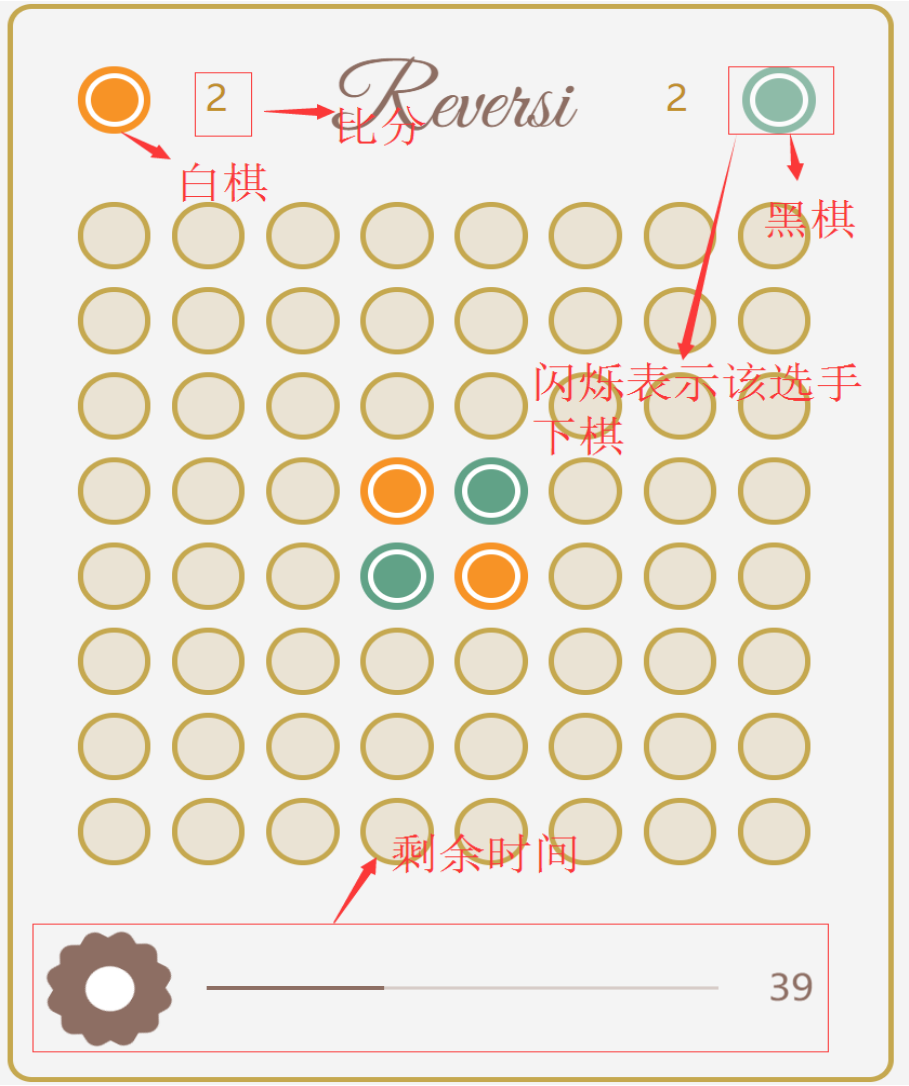

Figure 10 系统界面

3.2 操作说明

首先运行 C++程序，接着运行开启服务器，运行 DisplayUI 目录下的 test.js 脚本，然后在浏览器中输入 http://localhost:85/ 便可以进行我们的游戏了。用户可以通过鼠标点击对应的位置来落子。

具体的过程可以观看我们的 deemo 视频。

3.3 运行结果

C++ AI 程序

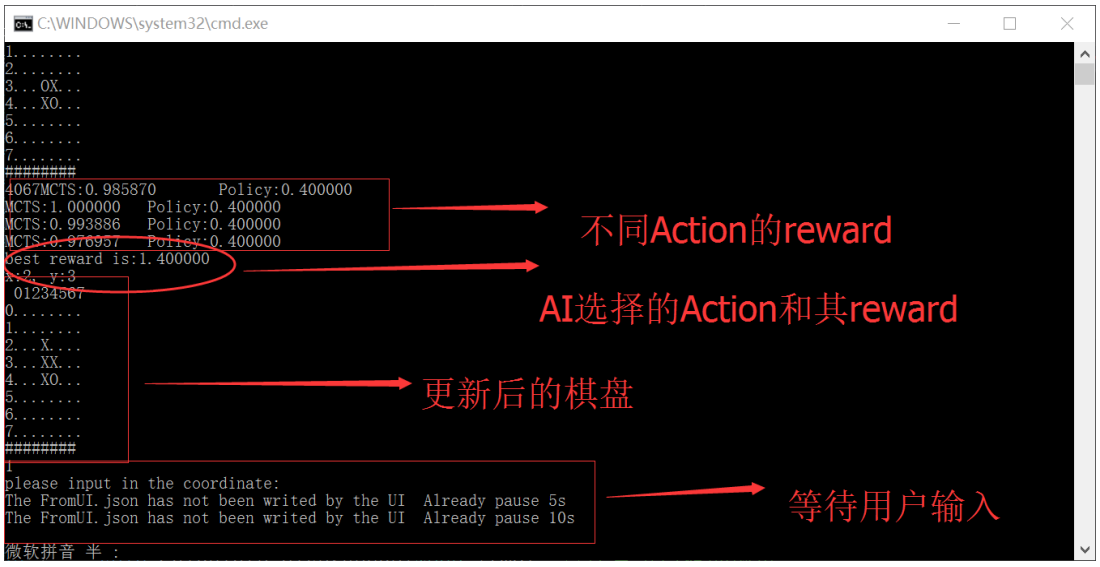

Figure 11 C++AI 运行

Node 服务器

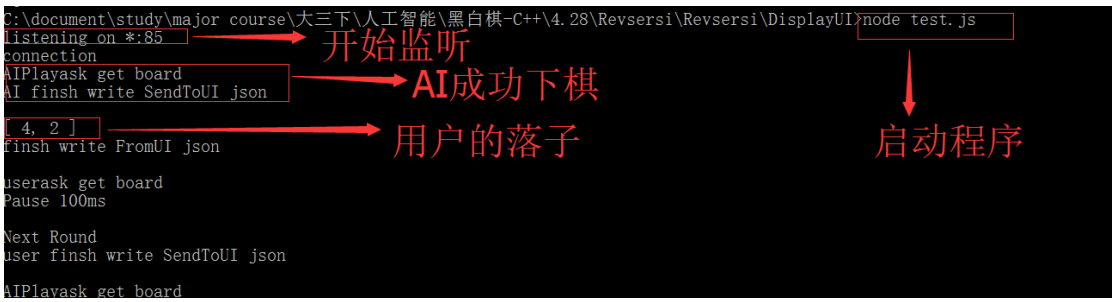

Figure 12 服务器运行

UI

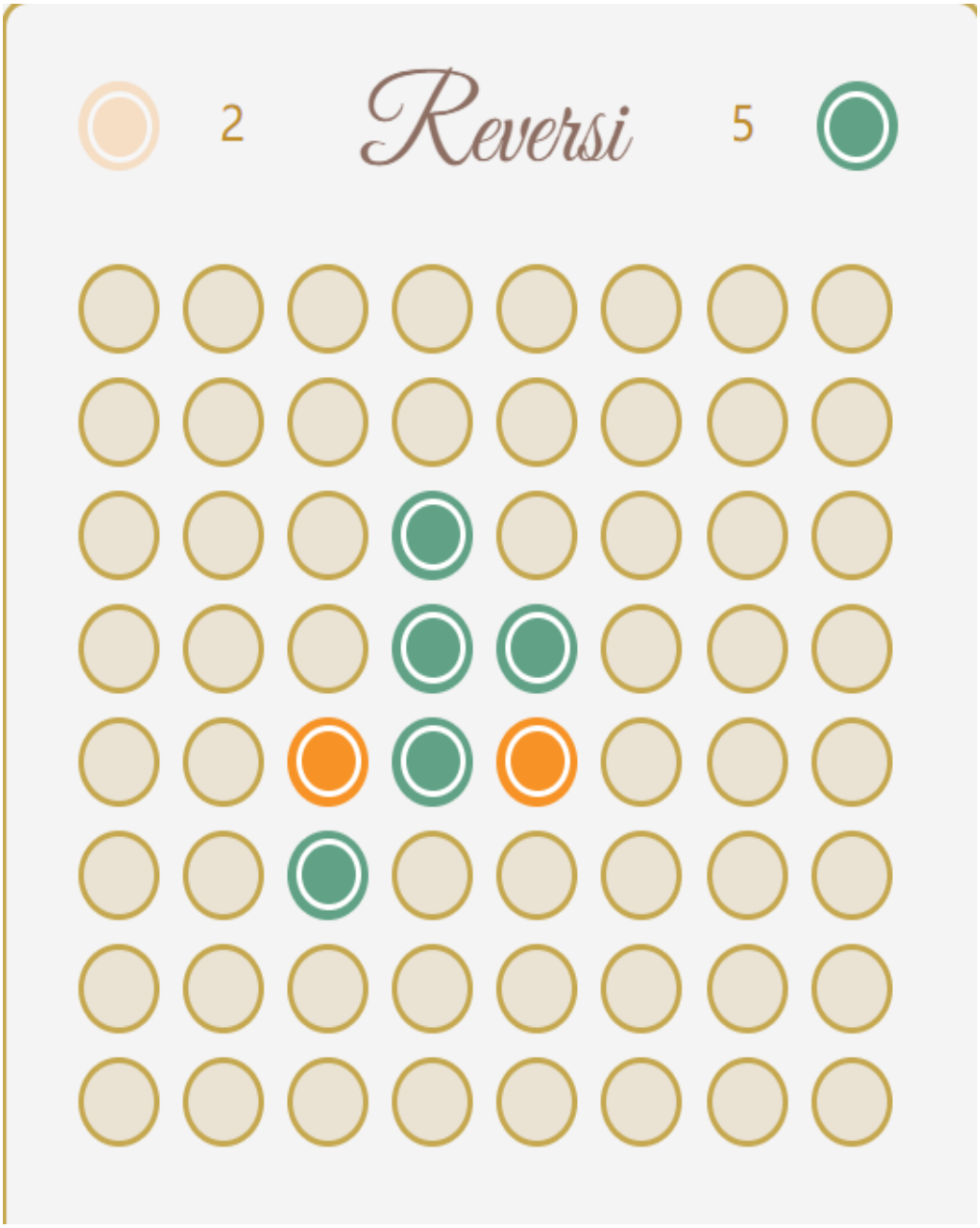

Figure 13 UI 变化

3.4 Java AI

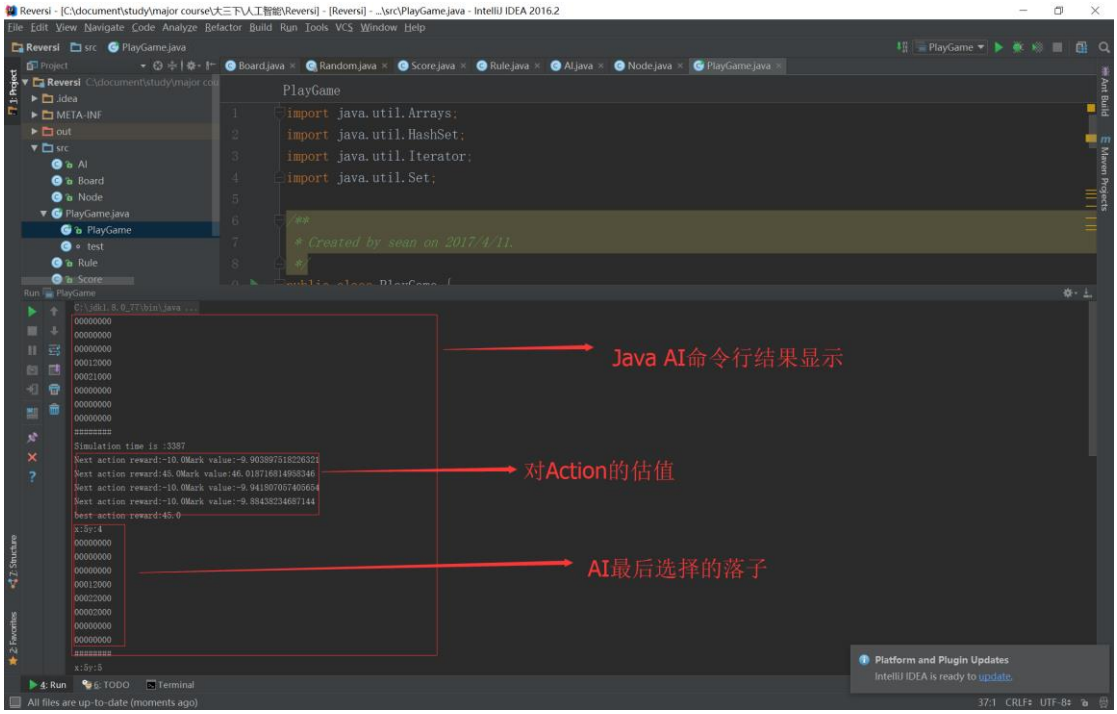

Figure 14 JAVA AI

References:

1. 吴老师提供的课件
2. 黑白棋新手速成虐菜教程（http://video.tudou.com/v/XMTc5MDE0OTU4NA==.html）
3. 黑白棋新手攻略（https://tieba.baidu.com/p/2853547344）
4. 黑白棋指南（电子书籍）（https://tieba.baidu.com/p/2281920455）