编号：2019F0209

哈尔滨工业大学
“大学生创新创业训练计划”
创新训练项目申请书

项目名称： 基于物联网的智能国际象棋对弈系统

申请级别： 校级 **拟申请经费** 10000 **元**

执行时间： ２０１９ **年** １０ **月至** ２０２０ **年** １０ **月**

负 责 人： 郭茁宁 **学 号：** 1183710109

联系电话： 13905082373 **电子邮箱：** gzn00417@foxmail.com

院 系 及 专 业： 计算机科学与技术学院-软件工程专业

指导教师： 聂兰顺 **职 称：** 副教授

联系电话： 13945068547 **电子邮箱：** nls@hit.edu.cn

指导教师： 刘劼 **职 称：** 教授

联系电话： 13701022478 **电子邮箱：** jieliu@hit.edu.cn

院 系 及 专 业： 计算机科学与技术学院-物联网工程

哈尔滨工业大学本科生院制

填表日期： 2019 年 10 月 8 日

一、课题组成员：（包括项目负责人、按顺序）

姓名	性别	所在院	年级	学号	身份证号	本人签名
郭茁宁	男	计算机科学与技术学院	大二	1183710109	350581200004173518	
林亦宁	女	电子信息与工程学院	大二	1180500806	350303200007141028	
方景瑞	男	计算机科学与技术学院	大二	1180301002	429005200101193417	

二、指导教师意见：

签　　名：

年　　月　　日

三、专家组意见：

1．是否达到中期目标（在□内打√）：

□达到中期目标　　□基本达到中期目标　　□未达到中期目标

2．成绩评定（在□内打√）：

□合格

□不合格，项目实施意见：
- □改进后可继续执行
- □提出警告、观察后再定继续执行或中止
- □中止实施

3．其它意见和建议：

组长签名：　　（ 盖　章 ）

年　　月　　日

四、项目研究中期报告

1. 项目简介

我们小组想要设计一套基于物联网的智能国际象棋对弈系统，能够提升国际象棋棋手的下棋体验感并对其水平提升起到积极效果：通过人机对弈，棋手可以调整 AI 水平与自己匹敌，产生良好的训练效果，并且可以通过复盘功能了解自己的行棋缺陷；同时也可以复盘比赛棋局，AI 会给出特定局面相应的评价和建议，棋手可以总结经验提升水平。我们设想在电脑上设计国际象棋 AI 程序，并且制作基于单片机的智能棋盘记录行棋过程，与电脑通信的过程中可以将 AI 分析的局势和最优走法体现在棋盘上，此外设计手机 app 支持棋手和观众随时随地了解棋局状态。在设计的过程中，设计国际象棋 AI 程序、制作智能棋盘、实现物联网通信、制作手机 APP 等将是我们要面临的困难，在攻克这一系列问题之后，我们预期能完成这一套系统，能够在实际操作中顺利使用。这一套系统将会为国际象棋比赛和棋手们平时训练和学习带来极大的便利，在传统的棋盘上展现了人工智能的力量，实现了电脑、手机、棋盘的多端互联，将国际象棋这一领域带上科技时代的潮流。

2. 立项背景

国际象棋是国际通行棋种，也是一项全球性智力竞技运动。而目前，许多训练基地大多采用大班教学的指导方法，缺乏对棋手进行个性化指导；想随时找到水平相当的对手也是有困难的。随着科学技术的发展，与人机对战成为了一种学习提升的有效途径，但市面上大多数 APP 都不智能，很难替代一对一面对面指导的方式，同时单调的手指在屏幕上滑动，缺少真实感，也免不了乏味感的产生。

基于物联网的智能国际象棋对弈系统，能够提升国际象棋棋手的下棋体验感并对其水平提升起到积极效果：通过人机对弈，棋手可以调整 AI 水平与自己匹敌，产生良好的训练效果，并且可以通过复盘功能了解自己的行棋缺陷；同时也可以复盘比赛棋局，AI 会给出特定局面相应的评价和建议，棋手可以总结经验提升水平。

目前市面上的人机对弈国际象棋软件琳琅满目，但智能棋盘却捉襟见肘，更别说存在人机对弈与智能棋盘的结合体。现有的技术编写出强大的 AI 程序已经唾手可得，然而制作出智能棋盘的公司仍然少之甚少，两个方面的技术水平不平衡，影响了国际象棋在智能化方向的发展。所以我们项目的技术设想很可能将会给国际象棋界带来突破性的发展，在国际象棋紧随科技化潮流的步伐上推波助澜，为这一国际化体育竞技项目做出举足轻重的贡献。

3. 项目方案

我们小组将设计一套基于物联网的智能国际象棋对弈系统，系统主要包括实现人机对战的智能棋盘等硬件设施，用于实时数据分析处理的电脑软件，与实时观战、复盘、指导的手机 APP 构成。针对的服务群体主要是尝试国际象棋的新手，与广大爱好国际象棋的棋手。结合物联网、人工智能等近两年科技、产业界的热门话题，旨在将该系统应用于棋手磨练棋技的指导工作中。目前功能设定主要包括：

（1） 当前棋局实时显示
（2） 之前棋局随时复盘
（3） 重大失误点识别
（4） 自定义残局开局
（5） 一对一手把手指导下棋
（6） 多人实时观看棋局

研究大致内容如下：

（1） 制作智能棋盘：棋盘上 64 个格子上装有相同的感应器，不同种类的棋子底部装有不同的芯片，棋面上的 64 个感应器可以识别不同的棋子，在棋局对弈的过程中实时监测并记录棋子的位置变化，

实时更新位置信息反馈至电脑内处理，基于 AI 程序的“思考”能够使电脑反馈给棋盘相应的走法，棋盘格子可以亮灯和语音来体现电脑的应着，棋手又可以依据电脑的走法来思考相应的应对策略。

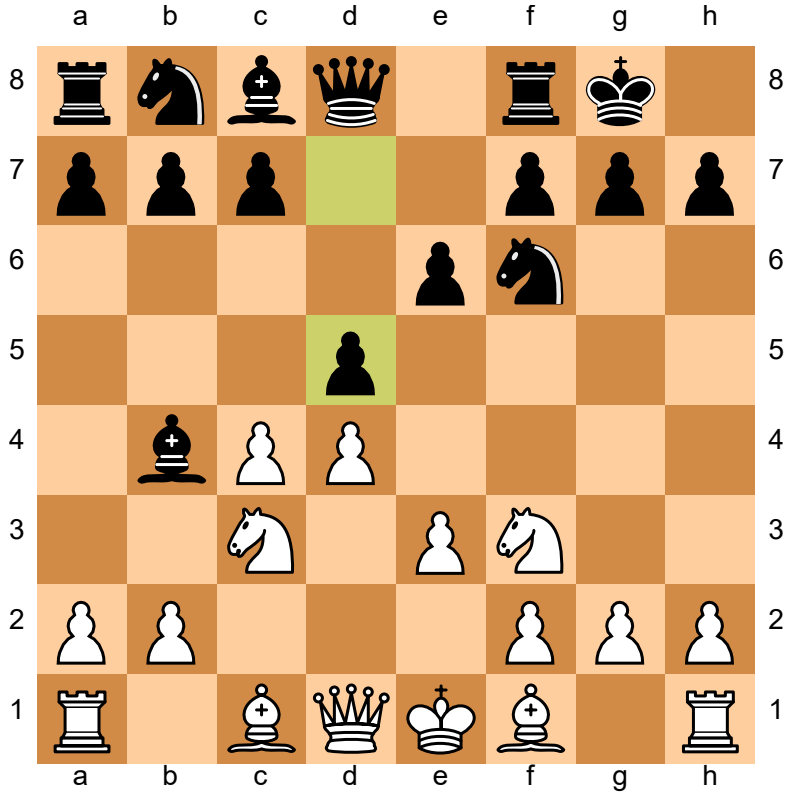
（2） 调试棋盘与电脑连接：电脑搭载智能算法，可以对棋手的走法做出即时相应，可以识别犯规走法，可以识别棋手重大失误，可以记录每一局对战，可以根据算法分析棋手下一步最优解，可以根据棋盘布局记录当前局势

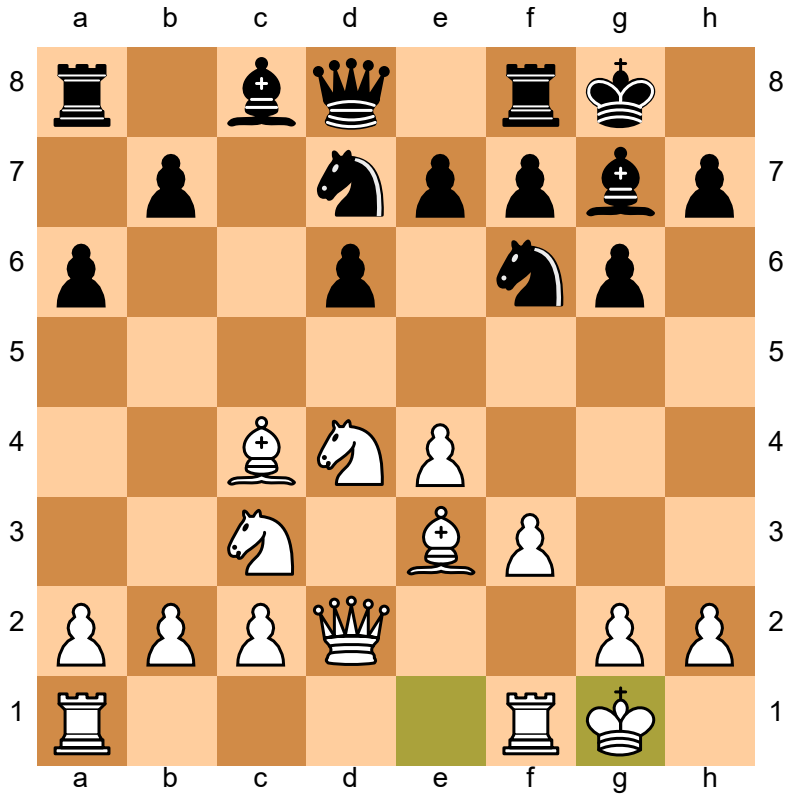
（3） 调试电脑与手机连接：电脑记录的局势通过互联网实时传输给手机，手机里 APP 可以以图像、语音等方式显示当前局势，多人可以通过 APP 进行实时观看学习。

4. 项目实施的进展情况及初步取得的成果

4.1 Chess-AI 程序

计算机博弈作为人机对弈的关键技术的理论基础，为人工智能的发展做出重要贡献。本文基于传统动态博弈算法中的 MINIMAX 搜索、α-β 剪枝和局面评估函数的理论研究以及在国际象棋上的算法推广实现，探索了静态搜索、空着向前裁剪和藐视因子等对于算法的导向型优化，并提出了冒险纵深式搜索（ADS）这一牺牲精度提高侵略性的搜索策略猜想，基于实战的测试数据对这些优化方案做出深入评价。





4.1.1 数字化动态模拟棋局状态

该国际象棋人机对弈系统使用博弈搜索树算法设计，玩家可以选择黑白放以挑战不同先后手的游戏。真人在下一步棋之后，电脑要充当裁判，对局面的可行性和结束与否进行有效判断；并且在一定时间的思考要有最佳的应着来匹配真人的走法；由此再把行棋权交还给真人。

虚拟棋盘人机交互

该人机对弈程序在面向对象的环节采用了朴素的人机交互：开始前可以选择先后手；棋局开始，真人输入走法，电脑产生相应应着并输出当前棋盘局面和局势值，棋局结束后程序显示赢家。

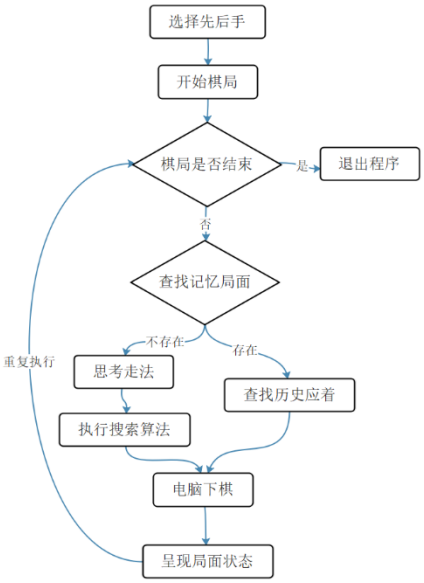


图 1 人机交互运行流程图

基于开局库的历史启发

开局是奠定孰优孰劣的关键时刻。在前人千百年来的基础上，优秀的开局定式已经被广泛认可，若真人依据布局来行棋，计算机可以调用已知的开局走法从而减少思考时间。

对弈角色选择

真人有权在棋局开始前选择先后手，单次输入后新一轮开始前不可更改。先后手确定之后，程序需要有一个标记，贯穿所有的函数，指示搜索。

走法生成

下面以真人执白为例：

棋盘记录真人出招后的局面，若已知棋谱中有该局面，则调用预置走法；否则程序开始进入搜索部分，计算机思考后得出最佳的应着，输出走法，打印棋盘局面。

```
AI_Move=搜索结果，即最佳应着
设立标签
保存该棋盘状态
board.push_san(AI_Move)执行
```

以上步骤循环往复，直至某一方行棋之后电脑判断棋局结束。

合理走法约束

在对弈中，计算机首先应该尊重棋的规则。所以解决如何判断走法的合法性、拓展出静态局面的合理走法等问题，是支持后续局面状态搜索的重要基础。

棋子预期移动落子点

国际象棋一共有 6 种棋子，每一种棋子有自己独特的走法。例如后，能够走“米”字型。若在后的 8 个方向的某一方向上若干格有一个对方棋子，则同一方向与原格子的距离大于（若为对方棋子则大于等于）存在棋子的格的距离与原格子的距离的这些格子都不在该后的合法行棋范围之内。即：

S_{Emp}: *当棋盘为空时指定棋子可到的格子集合*

S_{Nor}: *当一般状态下时指定棋子可到的格子集合*

P_0: *指定棋子的位置*

P_x: *指定棋子的可到的位置*

$$S_{Nor} = \{P_x | P_x \in S_{Emp}, path(P_0, P_x)\ is\ clear.\}$$

棋子中的马是个极为特别的例外，它的走法相比于其他棋子的“移动性走法”，可以说是“跳跃性”的，因为日字形走法堪比“虫洞”，棋盘的其他格子对其合法性不会有贡献，只有目标格子棋子的所属方对此有贡献。所以：

$$S_{Nor}^{N} = \{P_x^N | P_x^N \in S_{Emp}^N, P_x^N\ is\ empty\ or\ P_x^N \in Enemies\}$$

基于以上讨论的情况，可以预存储各种棋子的走法，并设计挑选合理走法的函数。以后为例：

```
def Queen_Legal_Steps():
    移动方向预存储
    for (int i=0;i<8;i++)
    {
        While(下一个格子无棋子在棋盘内
              或有对方棋子)
        {
            添加到可走格子列表
            指针移动到该格子
        }
    }
```

识别标准棋谱单步骤记录

现在的国际象棋界有一套完整的棋谱记录体系规则，在计算机中可以完整沿用这一体系进行单步骤的记录的识别。

	简称	辨别	列	行
兵	/	原所属列	a-h	1-8
车/马	R/N	原所属列/行	a-h	1-8
后/王/象	Q/K/B	/	a-h	1-8
特殊	短易位 O-O；长易位 O-O-O			

表 1 预设棋谱记录规则

人通过输入走法（例如：Nf3 e4 Rad1）来与计算机进行对话，计算机首先需要识别这一走法。以下大致呈现了拆解和分析走法的算法：

```
if 第一位是'O' then 长/短易位
else switch (字符串 strlen)：
        2：兵
        3：识别首位大小写
        4：车/马
从(x1,y1)移动到(x2,y2)
返回坐标移动数据
```

通过这一算法的实现，能够实现流畅的人机交互，并且有力支持了之后的计算机博弈系统的思考。

真人行棋合法性检测

真人下一步棋后，电脑通过合法性检测函数来判断真人走法的合法性，若不合法，则弹出提示，要求真人再次输入走法，直到合法为止。

```
def check_human_move(Human_Move):
    try:
        加入该走法
        判断走法合法性，合法则弹出
    except:
        不合法则返回 False
    return True
```

只有真人走法合法，计算机才能依据此进行后续的思考，否则违反规定的棋局无法继续。

4.1.2 局势值驱动的博弈树启发式搜索

在国际象棋的对弈程序中，传统的启发式搜索包含了局面评估、极小极大搜索以及 Alpha-Beta 剪枝。这 3 方面缺一不可，是 AI 算法的重点基础。

局面评估函数

局面评估是电脑基于搜索的“思考”的核心基础，评估的准确度直接影响了程序对弈能力的高低。因此，设计有效和精准的局面评估函数是很有研究意义的。作者基于该领域专业知识，设计了依赖于“子力和位置价值”、“机动性价值”的局面评估函数，可以计算每个局面的原始局势值$SumValue$，再由一下平衡调节公式使局势值大多数时候稳定在$[-1,1]$，与实际情况较为相符。

$$Color = \begin{cases} 1, & 电脑执白 \\ -1, & 电脑执黑 \end{cases}$$

$$SumValue = \sum_{i=1}^{n} Color(i) \times [PosValue(i) + MobiValue(i)]$$

$Evaluate\_Balance$: *平衡调节值*

$Evaluate\_Range$: *局势值缩小界*

$$AnsV = \frac{SumValue + Evaulate\_Balance}{Evaulate\_Range}$$

子力和位置价值

子力，顾名思义，指棋子的实力。在国际象棋中，不同种类的棋子在价值上各不相同：在人们通常的评估中，后为 9~10 分，车为 5 分，象和马为 3~3.5 分，兵为 1 分，王为无价之宝。所以子力的分值极大程度上影响了局势的倾向性。

Type	后	车	象	马	兵	王
Score	900	500	330	320	100	20000

表 2 程序预设子力分数

棋子在不同位置发挥出的价值又大不相同，例如：兵占据中心时能产生更大的价值，王在王城外容易被攻击等。

所以，程序在评估棋局时有必要将子力和位置结合起来考虑贡献给价值。基于这样的设计考虑，程序结合了子力和位置价值，预置了各种棋子的子力线性表、以及棋子位置价值的三维多层矩阵，用于局面的评估。

$$p \in Pieces = \{K, Q, R, B, N, P\}$$
$$PosValue(p) = PieceScore(p) \times Table[p][x, y]$$

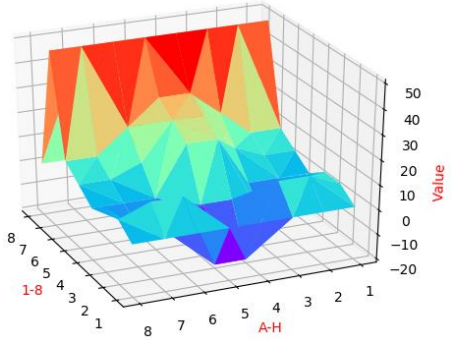


图 2“兵”的位置价值分布图

```
value_table = [
    #元素(x,y)的值代表兵在该格子获得的局势值因子
    [0, 0, 0, 0, 0, 0, 0, 0,
     50, 50, 50, 50, 50, 50, 50, 50,
     10, 10, 20, 30, 30, 20, 10, 10,
     5, 5, 10, 25, 25, 10, 5, 5,
     0, 0, 0, 20, 20, 0, 0, 0,
     5, -5, -10, 0, 0, -10, -5, 5,
     5, 10, 10, -20, -20, 10, 10, 5,
     0, 0, 0, 0, 0, 0, 0, 0],
```

机动性价值

机动性，在国际象棋中指每个棋子潜在的运动空间及威胁性。棋手不能只看到每个棋子下一步能产生的威胁，应该更长远地考虑潜在价值。因此，评估机动性，既要计算每个棋子能达到格子的数量，还要评价这些格子的战略价值、所攻击到的棋子（正在捉子）。

$$MobiValue = \sum_{p \in Pieces} \left[AtkNum(p) \times \sum_{(x,y) \in Atk\_List} Val(x, y) \right]$$

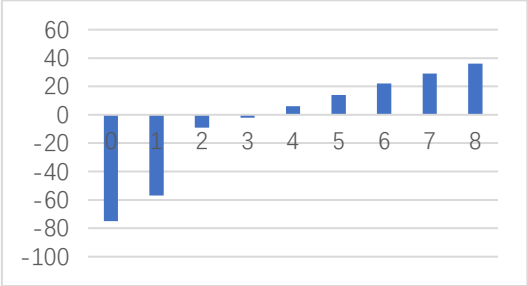


图 3“马”机动性评价参数

```
for i in 后可移动到的格子:if (i != -1):
```

```
QueensMobilityBonus-=该格子机动性和价值贡献
def 求指定棋子的机动性评价：
    枚举可攻击的位置
    attack_list = []
    for i in pcs_position:
            寻找攻击格子并统计数量
            查询对方棋子所在位置
            做交集（即该棋子能攻击到的对方格子）
            attack_list.append(atks_num - ol_num)
    return 攻击序列
```

MINIMAX 搜索

对于每一个局面，人机都有若干种不同的走法，而分别选择这若干种走法之后会产生若干个不同的局面；而这若干个局面又能分别产生若干个局面。如此往复，能生成一颗博弈树。

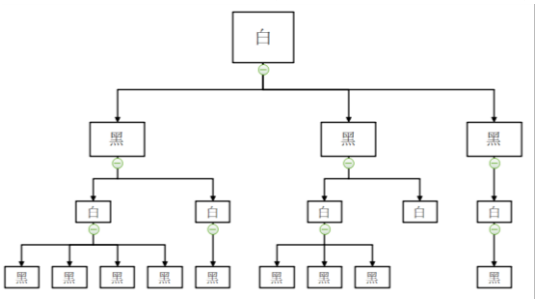


图 4 博弈树

博弈树每一层代表的执棋方是交错的，以

$$白 \rightarrow 黑 \rightarrow 白 \rightarrow 黑 \rightarrow 白 \rightarrow \cdots$$

顺序向下扩展。而在搜索时，电脑只代表单独一方，另一方的走法也要进行预测。在人与人的对弈中，我们往往都会考虑对方最好的走法是什么，即在预测对方走法时我们考虑自己的最坏情况。在最不利的情况下，正在思考的电脑要做出最有利于己方的走法。

在局势评估中我已经能对任意局面的局势进行评估。约定俗成地，我可以认为白方优势时局势值为正，黑方优势时（即白方劣势时）局势值为负。以电脑执白为例，因此在博弈树上搜索时的“预测对方走法、考虑自己最坏情况”就量化为“局势值最小”，而“考虑己方走法、预期最有利情况”则量化为“局势值最大”。

在博弈树的每一个节点，都有一个特定的局面（状态不一定唯一，但形成局面的走法一定唯一），每一个局面都可以进行评估，得出一个局势值。想要得到树根（当前局面）的最优走法，就是在搜索一条树的内路，由下向上贡献，使当前局面的局势值最大（白）。因此，我们预先设定搜索深度，得出整棵搜索树，从下往上贡献局势值——白方行棋时选择局势值最大的子节点走法、黑方行棋时则相反。在整个递推的过程结束后，根节点得到了最佳的走法的预测的最优局势值，计算机可以认为自己找到了当前条件下的最优走法，贡献给人机交互界面进行行棋。

```
def 深度优先搜索: #Search_Depth:搜索深度，越大 AI 越强
    初始化
    cur_dep=1
    while 当前深度<最大深度:
        通过函数返回元组（当前最佳走法，最佳局势值）
        if 局势值<alpha:
            alpha =-INFI
            continue
        elif 局势值>=beta:
            beta =INFI
            continue
        else:
            if temp_value > value:
                保存当前最佳走法和最佳局势值
            alpha=value - valWINDOW
            beta=value + valWINDOW
            增加深度
    返回最佳走法
```

α-β剪枝

搜索树是庞大的，尤其是当不断加大搜索深度的时候，从算法复杂度的角度来说，递归搜索的复杂度注定是呈指数级地增长。为了要优化时间，只能通过有效地剪枝来降低时间复杂度的常数。α-β 剪枝正是一个在博弈树搜索中重要的剪枝策略。

α：已评估的己方最优局势值

β：已评估的对方最优（己方最劣）局势值

x：局势值，$对于\forall x有\begin{cases} x>0 \rightarrow 白方优势 \\ x=0 \rightarrow 双方均势 \\ x<0 \rightarrow 黑方优势 \end{cases}$

在自下向上的递推贡献过程中，当某一走法形成的局面局势值小于α时，该走法一定是不优于搜索树浅层的任意走法的，不可能对局面有更好的提升，所以可以省去该局面所有分支的搜索和评估；当某一走法形成的局面局势值大于β时，该局面是对手不愿意遇到的，对手也一定有更好的走法可以到达更好的局面，避免这一差局面（对手角度），所以也可以省去该局面所有分支的搜索和评估。

$$\begin{cases} x<\alpha 剪枝 \\ x>\beta 剪枝 \\ \beta<x<\alpha 搜索 \end{cases}$$

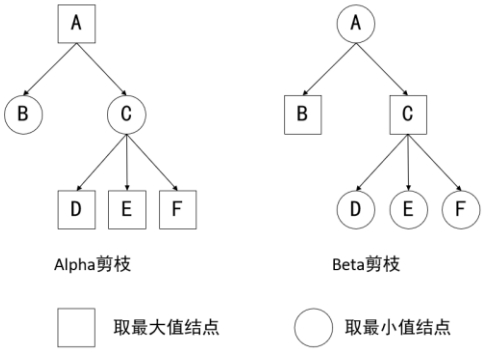


图 5 Alpha-Beta 剪枝示意图

```
if 局部最优局势值<=alpha:
          alpha =-INFI
          continue
elif 局部最优局势值>=beta:
          beta =INFI
          continue
else:
     if temp_value > value:
          更新总最优局势值
          以该局部最优解为总最优解
```

因此，当且仅当着法的局势值$x(\alpha<x<\beta)$时，博弈树会进行向下的拓展和加深深度。

4.1.3 传统零和博弈的导向型优化

普通的零和博弈，例如中国象棋和国际象棋，可以利用局势值驱动的传统启发式搜索进行“思考”。从算法复杂度的角度来说，指数级的算法已经不可避免，在剪枝极为有效的情况下才能尽可能地优化时间常数。

目前世界上的国际象棋程序在优化方面参差不齐，静态搜索是普遍会加入的优化策略，空着向前裁剪还难以在算法中稳定贡献，藐视因子对于根本性的算法能力提升只有调整性作用。

本文提出了冒险性纵深式搜索的大胆猜想，结合了藐视因子、静态搜索和空着向前裁剪各自的特点。这一算法具有极大冒险性，在实战中能有出其不意的走法，俨然一个侵略性极强的棋手。

以下讲逐一阐述算法优化的内涵、方法和具体实现。

静态搜索

在国际象棋中存在很多约定俗成、习以为常的应对。如果对方用象吃我的马，我们经常会吃对方的象。

基于 α - β 剪枝的启发式搜索难以针对强制性走法，而静态搜索就是一个侧重解决这一棘手问题的搜索策略。当 α - β 搜索到达最深层，选择调用静态搜索取代调用评估函数计算局势值，补充潜在动荡局面，规避在有明显应着的情况下决策失误。此外，为了解决典型静态搜索只考虑搜索吃子走法的片面性（局势封闭或者僵持、吃子走法只有大子吃小子，则不应强迫行棋），需要引入更完善的静态搜索思路。

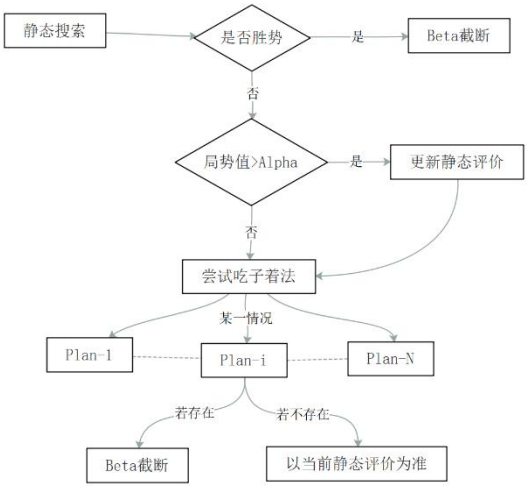


图 6 静态搜索思路流程图

上图描述了静态搜索思路：如果评价好得足以截断而不需要试图吃子时，就马上截断，返回 Beta；如果评价不足以产生截断，但是比 Alpha 好，那么就更新 Alpha 来反映静态评价。然后尝试吃子着法，如果吃子的任意选择中存在之一产生截断，便立刻退出迭代。或许它们没有一个是好的，无伤大雅。这个函数过程有几个可能的结果：

1. 可能静态局势值极高，使静态搜索通过 β 截断马上返回；
2. 可能存在吃子走法产生 β 截断；
3. 可能静态局势值较差，所有吃子走法都不能有效贡献；
4. 可能任何吃子都不够好，但是静态评价只比 α 高一些；

```
def 静态搜索:
    获得当前局部最优值
    if value >= beta:
        return beta 产生β截断
    if value > alpha:
        alpha = value 任何吃子都不够好
    静态搜索杀棋可能
    for i in capture_san_list:
        拷贝棋盘状态
        局面状态标记
        尝试静态走法并保存
        value = 深搜最优解
        回溯并更新最优值
    return alpha
```

空着向前裁剪

空着向前裁剪，即己方不行棋而让对手连续走 2 次。存在可走可不走的状态，空着确实是思考时不错的选择。

图 7 空着向前裁剪典型局面

在搜索中让程序尝试向前的空着，能有效加快速度和提升精准度。例如：

如果此时局面对电脑来讲是胜势，规定：

$$N: 搜索层数$$

$$Br: 分枝因子$$

则空着代替了整棵搜索树，搜索空着就等价于只搜索了一个$N-1$层的分枝。依据统计，中局阶段平均来说：

$$Br \approx 35$$

因此空着启发搜索仅仅使用了原有的全部搜索占用的3%的空间，理论上节约20%~75%的搜索时间。

```
if board_info['null_move_pace']<2:
    null_move_san=board.san(chess.Move.null())
    加入空着并记录局面
    求得当前状态最优值
    if value >= beta:
        发现潜在优势局面
        return beta
    else:
        取消空着尝试
```

藐视因子

在博弈论中，定义：

$$Cf: 藐视因子$$

藐视因子反映国际象棋 AI 下棋的风格，如保守，均衡或冒进。程序中可以依据对局的状态选择不同的藐视值。如果是很焦灼的中局、程序有些劣势，经常是在电脑执黑时，可以调大藐视因子，程序就可以为胜利而战。在本研究中，我方的评估价值的基础上乘以

$$Cf = 0.98$$

的参数，确定了该 AI 程序相对均衡的下棋风格。

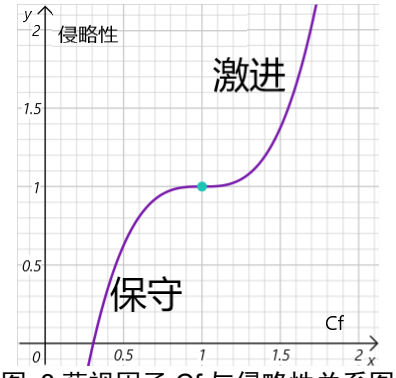


图 8 藐视因子 Cf 与侵略性关系图

冒险纵深式搜索猜想（ADS）

相同情况下，程序的博弈能力与搜索深度呈正相关。某一局面下的许多走法对于局面贡献微乎其微，因此我们可以集中精力对“主要变例”进行纵深搜索，即加深某一重要走法尝试的搜索深度，可以对这一走法优劣的了解程度更加深刻和准确。极小更优忽略不计，通过牺牲搜索精度，提高搜索速度和搜索深度。这非常符合人类在思考时，顾虑于时间因素，对于一些边缘走法不予理睬，只会着重考虑关键走法的未来情况。

然而，这样的搜索策略是存在显而易见的风险的：某些对于局势值贡献不大的走法，可能为之后的厚积薄发埋下伏笔，忽略它们可能会错失一些取得优势的渠道。

因此，作者认为这样的纵深式搜索仅仅是一种猜想，更保守来说，它应该是一个程序下棋风格的重要体现。程序可以基于有局势值大波动的情况进行纵深，也可以重点考虑吃子情况，还可以主要搜索平淡的走法……不同的风格注入到程序里，对于不同类型的棋手，风格可以各不相同。

以下是侵略型程序的冒险纵深式搜索：

```
while 搜索深度范围以内:
        (temp_value, temp_final_san) = 最优解
        if 搜索局势值对于当前影响<10%:
            则放弃这一走法的纵深
            continue
        elif 此走法对于对方局势侵略性不大:
            则放弃这一走法的纵深
            continue
        else:
            if 总最优局势值可更新:
                更新局势值
                保存最“侵略性”走法
            更新 Alpha 和 Beta
            纵深
```

4.2 手机 APP

最终 APP 可以实现：当前棋局实时显示；之前棋局随时复盘；重大失误点识别、自定义残局开局、一对一手把手指导下棋、多人实时观看棋局。共分为三个页面：

①主页：包括棋盘可视化、AI 智能指导、走棋历史等等。

②功能页：包括查看历史记录、复盘之前棋局、自定义残局开局等等。

③设置页：包括调整字体大小、恢复默认设置、适配手机等等

目前已经可以实现棋盘可视化，手动操作棋子，调整字体大小，恢复默认设置，适配不同类型的手机

①棋盘可视化

效果（如图 1）：实时展现棋盘、棋子位置，全屏视角，给观棋者提供完美的观看体验

实现思路：主要通过图像控件的布局，将已有的棋子、棋盘图像显示在手机上。对棋子的操作后，进行图片的平移、平移组合、消失，促使棋子做出相同反应

图 1-虚拟机棋盘展示

②手动操作棋子

效果（如图 2）：可通过特定指令，操作棋子，包括棋子移动、棋子吃子、特殊操作（王车易位、禁卫军的晋升），通过流畅的动画表现，带给观棋者完美的体验。将来结题时，可以通过智能棋盘传输的指令来实现自动操作棋子

实现思路：对传入的指令进行分析，为了统一格式，指令的长度永远为 4。一共有 4 种类型的指令

移动时（如图 2）：前两位表示棋子的起始位置，后两位表示棋子的目标位置，图像移动（如 b1c3 表示将 b1 的白色骑士移动到 c3）

吃子时（如图 3）：前两位表示行动方棋子的位置，后两位表示被吃棋子的位置，图像移动，被吃棋子消失（如 a8c3 表示用 a8 的黑色战车吃掉 c3 的白色骑士）

王车长/短易位（如图 4/5）：国王向战车的方向移动 2 格子，战车向相反方向越过国王，移动到其旁边（如 w000 表示白方长易位/0w00 表示白方短易位）

禁卫军的晋升（如图 6）：禁卫军移动到对方棋盘的最后一行时，可以发生晋升行为，变为其他类型棋子（除了国王和禁卫军），图像被替换为晋升后的棋子

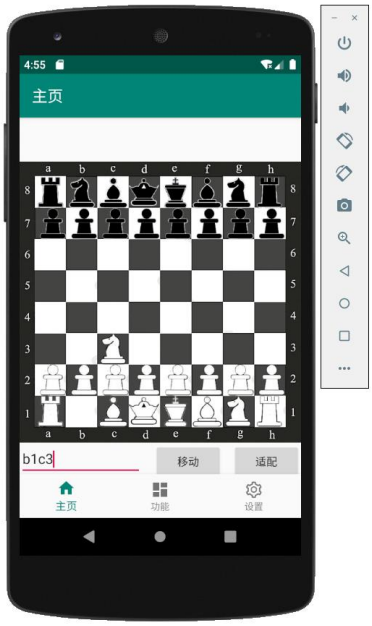


图 2-移动棋子（虚拟机）

图 3-吃子（虚拟机）

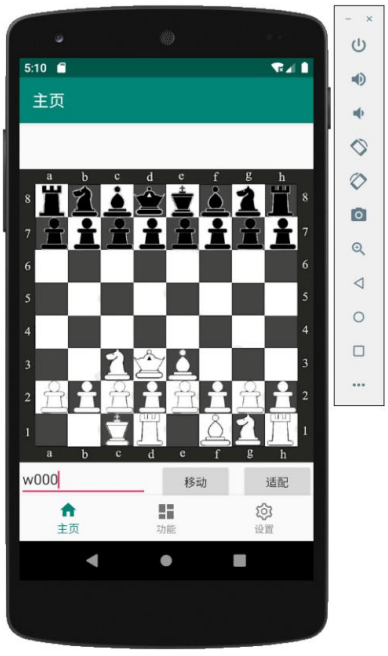


图 4-王车长易位（虚拟机）

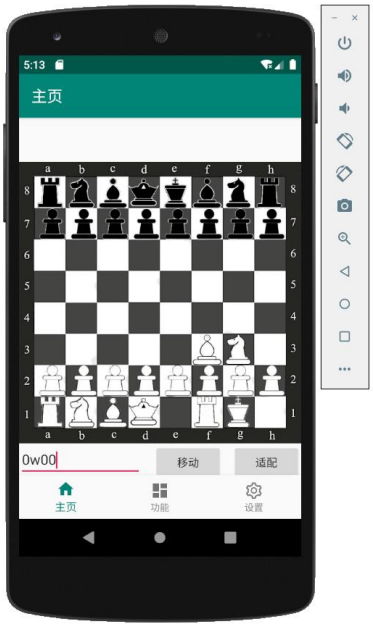


图 5-王车短易位（虚拟机）

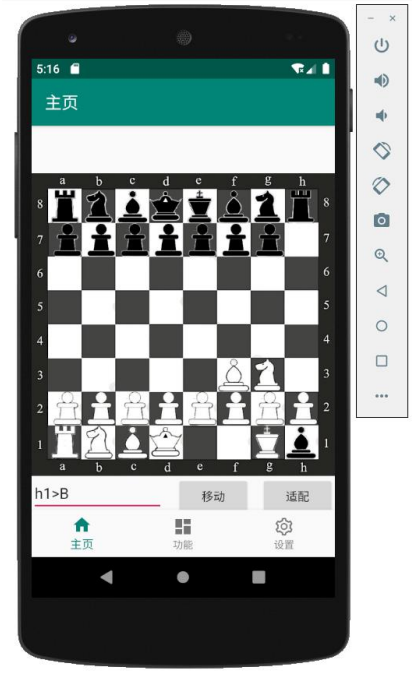


图 6-禁卫军晋升（虚拟机）

③调整字体大小

用户可以根据自己的喜好调整字体大小，提升用户使用体验

实现思路：通过按钮“放大”或者“缩小”来改变全局字体的大小（如图 7）已经设置阈值，防止控件字体过大，超出控件范围，导致显示异常

图 7-缩小字体大小（虚拟机）

④恢复默认设置

可以清理棋盘上的历史记录、适配、字体大小等等数据，恢复默认设置

实现思路：将所有变量进行初始化，变为原来的值，给出提示（如图 8），随后重启 APP 后生效

图 8-恢复默认设置（虚拟机）

⑤适配不同类型的手机

由于 Android 手机屏幕大小，长宽比多种多样，适配不同类型的手机，扩大用户面，是必不可少的。为了测试阶段方便，目前设计了适配按钮，在操作棋子的之前，需要进行手机适配，适配后棋子移动的距离刚好是棋盘上一个格子的长度。将来将做成提示框，在用户移动之前给予提示。

实现思路：将图像移动的距离从像素改为密度无关像素（dip：Density Independent Pixels），当用户按下适配按钮时，将会根据用户手机的屏幕尺寸，调整图像移动的距离。换言之，棋子将可以移动一格（如图 9）。真机演示上，若未适配，可能导致棋子移动异常（如图 10）；反之，显示正常（如图 11）

图 9-适配（虚拟机）

图 10-未适配（真机）

图 11-适配后（真机）

4.3 电子棋盘

项目的硬件部分是一种集多种传感器的智能电子棋盘,包括:主体、内部电路、计算机终端。

主体部分包括:棋子、棋盘面板、内部固定框、棋盘底板,所述棋子底部粘贴有 NFC 电子标签,所述内部固定框安装在棋盘面板与棋盘底板中间,所述棋盘面板表面设有若干落子点,所述棋盘底板底面四角均设有防滑脚垫。

所述内部电路安装在内部固定框内,包括:RFID 感应电路板、棋盘控制电路板,所述 RFID 感应电路板安装在棋盘面板与棋盘底板之间,RFID 感应电路板上设有与落子点一一对应的 FID 识别芯片,每个 RFID 识别芯片上设有天线并构成 RFID 感应电路,所述棋盘控制电路板上设有译码电路、MCU 控制电路、通讯电路,FID 感应电路通过译码电路与 MCU 控制电路连接,MCU 控制电路通过通讯电路与计算机终端通信。

现有的电子棋盘采用的是光电感应技术，每个棋格底部放置有-一个光电传感器，当棋子落下时，光电传感器上方受到棋子的遮挡后,周围的光线亮度产生变化从而产生电流信号传送至控制芯片 MCU 进行处理。因此该技术方案能识别棋子是否已落下以及落下的位置,适用于只有黑、白两种棋子的围棋,围棋执黑先走,因此通过软件判定第一步黑子 ,第二步白子,依次类推每--步棋子的颜色。

但以国际象棋为例，棋子分为黑方和白方,且每方都有 8 个兵、2 个车、2 个马、2 个象、1 个王后、1 个国王,共计 32 个棋子,种类较多,现有的光电感应技术仅能识别棋格.上是否有棋子落下,无法识别当前所落下的棋子是哪-一个棋子。

5. 特色与创新

- 当前棋局实时显示：智能棋盘上的棋子变化实时显示在 app 上，app 上的棋子移位实时更新在智能棋盘上，方便棋手随时观察棋局与人机对战；
- 对局记录实时完成：由电脑记录每一局的每一步，省去在对局记录纸上的操作；
- 之前棋局随时复盘：电脑记录每一次对局，可随时复盘任意一局对战，方便反复研究琢磨对局，方便棋手制定个性化对局策略，提高棋手技艺；
- 重大失误点识别：可智能识别任意一局对战中重大失误点，前车之鉴后车之覆，棋手之路平步青云；
- 自定义残局开局：通过在棋盘上自定义摆放棋子，可实现残局开局，与人机反复磨练，巧解残局；
- 一对一手把手指导下棋：可智能提示下一步的最优解，为想要提高技术的棋手提供全方位的指导；
- 多人实时观看棋局：给有共同爱好的棋手们提供一个交流学习的方式，一起交流，共同进步；
- 真实人机对战：摆脱传统手机 app 上滑动屏幕时几乎为 0 的反馈感，真实棋盘，享受人机对战；
- 恰逢哈尔滨工业大学建校 100 周年，重现 1958 年全国第一台会下棋、会说话的计算机，迎接百年校庆；

6. 项目实施过程中的收获与体会

6.1 成员 1

我主要负责项目的 AI 程序部分。在 2019 年 10 月开始学习计算机博弈的相关原理，并尝试编写 Python 程序。在学习原理的过程中，可谓非常艰难，枯燥的证明和繁琐的操作，加上实现的艰难，给程序的编写带来极大的挑战。此外，对于 python-chess 底层库的理解和运用也花费了一定精力。

随着工作进展，程序不断被完善。在去年年底时完成了控制台的对弈，在今年寒假，又基于 Jupyter Notebook 完成了可视化渲染的客户端搭建。最后完成了实战测试，并发布在了 GitHub 和 CSDN 上。目前该 AI 程序可以在计算机上运行，接下来的任务是将其移植。

6.2 成员 2

我主要负责硬件设计部分。2019 年 11 月开始尝试设计电路，搭建模块。起初，我采用压力传感器对国际象棋的棋子进行定位，但由于象棋不比围棋，棋子种类较多无法识别当前所落下的棋子是哪-一个棋子，之后我采用 NFC 传感器给每个棋子赋予唯一的 ID,每个棋格下方有独立的感应电路,棋子类型的识别以及每个棋子的位置就被唯一确定下来。

棋盘设计以及部分功能模块确定之后，人工搭建电路带来了较大的困难。一是手工带来的误差，二是未采用集成电路带来性能上的不稳定性。此外，硬件部分尚未与软件部分连接，这也是我们团队后期将想方法改进的部分。

6.3 成员 3

我主要负责项目的 APP 部分，最初开始于今年二月份，那时候还不会使用 JAVA 语言，对 Android 开发也是一知半解，甚至不知道开发工具以及该如何使用。开发的初期，可以说是，困难重重。当然，随后在 mooc 上学习了 JAVA，在博客上学习了 Android 开发，稍微克服了部分困难。

然而，随着工作的逐步进展，涉及的面越来越广，遇到的困难也自然越来越多。比如涉及到棋子的图片、棋盘的图片，没有现成的，只能自学 PS，利用网上模糊的图片，做出了图片素材；比如涉及到棋子移动的动画效果，便自学了动画效果、图片像素等等相关的知识；比如涉及到适配手机，又在博客、github 上查阅学习相关的知识。这可以说是，从 0 到 1，从无到有的典型。当然，目前我已经可以掌握 Android 开发的基本知识，在后续的项目进展中，助团队进展突飞猛进。

7. 经费使用情况

使用物件	开销
RFID 射频感应模块	163.75
野火挑战者 STM32F767 开发板	814.00
Arduino UNO 开发板	30.00