中文图书分类号：TP311
密　　级：公开
UDC：004
学　校　代　码：10005

硕士专业学位论文

PROFESSIONAL MASTER DISSERTATION

论　文　题　目：六子棋中基于路的双评价参数评估函数的研究与应用

论　文　作　者：齐祎霏

专业类别/领　域：软件工程

指　导　教　师：朱青　王瑾

论文提交日期：2018 年 5 月

UDC： 004　　学校代码：10005

中文图书分类号：TP 311　　学　　号：S201525093

密　　级：公开

北京工业大学硕士专业学位论文

（全日制）

题　　目：六子棋中基于路的双评价参数评估函数的研究与应用

英文题目：Double Parameter Evaluation Function Based on Path

论 文 作 者： 齐祎霏

专业类别/领域： 软件工程

研 究 方 向： 数字媒体技术

申 请 学 位： 工程硕士专业学位

指 导 教 师： 朱青 王瑾

所 在 单 位： 软件学院

答 辩 日 期： 2018 年 5 月 31 日

授予学位单位： 北京工业大学

独 创 性 声 明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京工业大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签　　名：齐祎霏

日　　期：2018年5月31日

关于论文使用授权的说明

本人完全了解北京工业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵守此规定）

签　　名：齐祎霏　　日　　期：2018年 5 月 31日

导师签名：朱青　　日　　期：2018年 5 月 31日

摘 要

机器博弈是人工智能领域的重要研究方向，是机器智能、兵棋推演、智能决策系统等人工智能领域的重要科研基础。机器博弈被认为是人工智能领域最具挑战性的研究方向之一。机器博弈常被看作是人工智能的“果蝇”，其研究成果对其他领域也能产生重要影响，因此对机器博弈的研究具有重要的意义。六子棋是机器博弈中的一个新兴棋种，近几年越来越走进大众的视野，吸引了越来越多的研究者行进研究。

评估函数是博弈程序对局面形式进行判断的核心方法，评估函数的准确程度决定了程序对局面形势判断的准确程度，从而程序才能选择正确的应对方案。现有六子棋评估函数大致分为基于棋型和基于路的两种。基于棋型的评估函数对棋型的分割非常复杂和耗时，而传统基于路的评估函数不能很好地应对各种局面。本文重点研究六子棋的评估函数，针对现有基于路的评估函数存在的问题提出了双评价参数评估函数，解决了传统基于路的评估函数不灵活且不能应对不同局面的问题。本文研究了基于全局路的评估函数和基于局部路的评估函数的特点，将二者相结合，使评估函数既可以准确地评估全局，又保证了执行效率。同时使用了两组参数分别应用于我方优势与我方劣势的情况，以应对不同的局面。

本文为验证双评价参数评估函数的合理性，设计了棋局局面倾向实验、估值准确性实验和棋力水平实验。局面倾向性实验证明不同实验参数设置会导致不同倾向的说法。估值准确性实验证明在某些局面下，双评价参数评估函数对比传统基于路的评估函数能够更准确的判断当前局面，能更准确地引导棋局。棋力水平实验证明了使用双评价参数评估函数对比传统基于路的评估函数有着更高的胜率。此外由于评估函数对搜索效率有着很大的影响，设计了搜索效率实验对比两种评估函数的评估效率。本文最后基于所提的双评价参数评估函数设计实现了六子棋程序 Executor，详细介绍了各个功能模块的功能与实现方法，进一步验证了本文方法的有效性。

关键词：机器博弈；六子棋；评估函数；双参数评估函数

Abstract

Computer game is a challenging research field that is full of vigor. It is an important research direction of artificial intelligence. It's the base of machine intelligence war gaming, intelligent decision system etc. Machine game is considered to be one of the most challenging research directions in the field of artificial intelligence. Computer game is often seen as a "fruit fly" of artificial intelligence. Its research results also have an important impact on other fields. Therefore, it has significant meaning on research of computer game. Connect6 is a new kind of chess which has been occurred in recent years. It has become more and more popular in recent years, attracting more and more researchers to study.

Evaluation function is the core method for judging the situation of a position for the game program. The accuracy of evaluation function determines the accuracy of the procedure to judge the situation, so that the program can choose the right response plan. Generally, the evaluation methods can be roughly divided into two categories, based on the evaluation function of the chess type and the evaluation function based on the road. There is no good solution to the segmentation problem of connect6 when use evaluation function based on chess type. The segmentation of chess type is very complicated and costs too much time. The traditional road based evaluation function cannot deal with all kinds of situations well. In this paper, we study the evaluation function based on global path and the evaluation function based on local path. The combination of these two functions is an evaluation function that can accurately evaluate the overall situation and ensure the execution efficiency. At the same time, two groups of parameters are applied to our advantages and our disadvantages, in order to cope with different situations.

In order to verify the rationality of double parameter evaluation function, this paper design the inclination experiment、the accuracy of estimation experiment and the power level experiment. The inclination experiment verifies that different experimental parameters will lead to different inclination. The estimation accuracy experiment verifies that in some situations, the double parameter evaluation function can more accurately judge the current situation than the traditional road based evaluation function, and can guide the game more accurately. The experiment of power level proves that using the double evaluation parameter evaluation function has a higher winning rate than the traditional road based evaluation function. In addition, due to the great influence of evaluation function on search efficiency, a search efficiency experiment is designed to compare the efficiency of two evaluation functions. At the end of this paper, based on the proposed double parameter evaluation function, we design and implement

a connect6 program called “Executor”. The functions and implementation methods of each function module are introduced in detail, and the effectiveness of the method is further verified.

Keywords： machine game;connect6;evaluation; double parameter evaluation

目 录

摘 要.....I

ABSTRACT.....III

第 1 章 绪论.....1

1.1 论文研究背景与意义.....1

1.2 国内外研究现状.....1

1.2.1 机器博弈研究现状.....1

1.2.2 K 子棋和六子棋研究现状.....2

1.3 主要研究内容.....2

1.4 论文结构.....3

第 2 章 六子棋理论基础.....5

2.1 六子棋基本概念.....5

2.2 六子棋棋盘表示.....6

2.3 博弈树.....7

2.4 评估函数.....7

2.4.1 基于棋型的评估函数.....8

2.4.2 基于棋型的评估函数存在的问题.....9

2.5 搜索技术.....10

2.6 本章小结.....11

第 3 章 基于路的双评价参数评估函数.....13

3.1 基于路的评估函数.....13

3.1.1 “路”的定义.....13

3.1.2 基于路的全局扫描评估函数.....15

3.1.3 基于路的局部扫描评估函数.....18

3.2 基于路的双评价参数估值方式.....21

3.2.1 估值参数对棋局的影响.....21

3.2.2 基础局面估值.....21

3.2.3 基于路的全局扫描与局部扫描相结合.....22

3.3 实验及结果.....24

3.3.1 实验环境.....24

3.3.2 实验构思.....24

3.3.3 棋局局面倾向实验.....25

3.3.4 估值准确性实验.....27

3.3.5 棋力水平实验.....29
3.3.6 搜索效率实验.....32
3.3.7 参数影响实验.....34
3.4 本章小结.....35
第 4 章 基于双参数评估函数六子棋程序的设计与实现.....37
4.1 六子棋体系结构.....37
4.2 人机交互界面.....37
4.3 走法生成器模块.....39
4.4 搜索引擎模块.....40
4.5 HASH 函数模块.....41
4.6 开局库模块.....45
4.7 程序整体流程.....46
4.8 本章小结.....47
结 论.....49
参 考 文 献.....51
攻读硕士学位期间取得的研究成果.....55
致谢.....57

第1章　绪论

1.1　论文研究背景与意义

人工智能是计算机科学的一个分支，该领域的研究包含机器人、语言识别、图像识别和自然语言处理等。随着计算机技术的发展，人工智能慢慢成为一个瞩目的研究热点，其研究成果在多个领域中得到了广泛的应用。1956 年夏季，以麦卡赛[39]、罗切斯特、明斯基和申农等为首的一批年轻科学家聚在一起，共同研究和讨论用机器模拟智能的一些值得注意问题，并第一次提出“人工智能”这一名词，它成为了“人工智能”这门新兴学科正式诞生的标志。

机器博弈是人工智能领域的重要研究方向，是智能决策系统、兵棋推演等领域的重要研究基础。机器博弈被认为是人工智能领域最具挑战性的研究方向之一。机器博弈先由国外发展起来，最早国外的一些研究者先用国际象棋做实验，现在围棋慢慢代替了国际象棋的位置，成为实验的主要项目。如今，机器博弈在国内慢慢发展起来，一些组织和机构慢慢对机器博弈的发展重视起来，组织了一些比赛，吸引了个高校学生和研究人员参与进来。机器博弈常被称为人工智能的“果蝇”，国内外有许多研究者投身于机器博弈的研究中，在此过程中衍生出大量的研究成果，而这些研究成果通常可以应用于其他领域的生产当中。机器博弈可以对其他研究领域产生重要影响，对机器博弈的研究具有重要研究意义。

六子棋是近几年兴起的棋类博弈项目，它最早是由台湾交通大学吴毅成教授于 2005 年发表的。六子棋由五子棋发展而来，而在规则上做了些许的改进，这使六子棋有了更多变化，相较于五子棋更加复杂。六子棋黑棋第一步先落一子，随后黑白双方轮流落两子，率先连成六子在一条直线的一方获胜。由于每一步棋可以落两子，六子棋的棋型组合要比五子棋更多。棋型的复杂度已经被评估为仅次于围棋与日本将棋，远远高于五子棋及西洋棋，与中国象棋、国际象棋相当[10]；而在游戏公平性方面更，由于黑、白各方每次下完一手后，盘面都比对方多一子，这使公平性大为提升。

1.2　国内外研究现状

1.2.1　机器博弈研究现状

国外的机器博弈发展较早,1977 年的 ICGA(International Computer Game Association)成立。该国际组织为推动该领域的发展做出了巨大贡献。其中的一些研究成果发表在国内外重要的期刊和会议中。最早机器博弈常把国际象棋当做实验的对象，研究者们不但提出了优秀的理论成果,还将博弈软件开放源代码,供其它研究者阅读和使用。在 IBM 的深蓝战胜卡斯帕罗夫之后，近些年围棋的研究

逐渐变为机器博弈的热点,并取代了国际象棋成为新的果蝇。近年来,出现了相当多的围棋开源软件，各大论坛都有着围棋研究者热烈的讨论。

中国大陆的机器博弈研究近年来发展迅速。2008 年中国人工智能学习机器博弈专业委员会正式成立。该委员会意在推广和普及机器博弈知识,提高国内机器博弈的研究水平。该组织每年一度会举办中国机器博弈锦标赛、人机大战以及学术研讨会，吸引了不少国内外学者，研究者和机器博弈爱好者的积极参与。近年来,国内的一些科研院所在机器博弈领域做出了重要贡献,包括:东北大学徐心和团队以及该团队的“棋天大圣[14]”，该团队数次获得国际国内的冠军,还在人机大战中屡次战胜或战和人类大师;此外,东北大学的六子棋、点点连格等棋类项目也在不断的研究中;北京邮电大学的刘知青团队等对围棋做了大量的研究;北京理工大学黄鸿教授的团队对中国象棋、围棋、六子棋、亚马逊棋类的研究也有出色的成果;南京航空航天大学的夏正友、哈尔滨工业大学的王轩、重庆理工大学的张小川、桂林电子科技大学的郝卫东、大连理工大学的高强等也做出了重要贡献。虽然国内对机器博弈的研究起步较晚,但研究水平发展迅速,研究的范围也在逐步扩大,包括四国军棋、兵棋、桥牌等的研究也正在如火如荼地进行。

1.2.2 K 子棋和六子棋研究现状

K 子棋是一组相似的连珠游戏，通常表示为 Connect(m，n，k，p，q)，五子棋与六子棋属于 k 子棋的两种棋种。从组合博弈的研究成果—著名的“策略盗用”原理可以推断,在 k 子棋博弈问题中白方永远不会赢棋,亦即最终的解或者是黑胜,或者是和棋。此外,文献[2,3]列举了一些 k 子棋的解的情况,文献[5-9]研究了基于 Maker-Breaker 规则下的 k 子棋,文献[10]研究并指出了 k 子棋的复杂度。在 k 子棋机器博弈研究中,文献[11]和[12]分别给出了破解 Go-Moku 和 Renju 时所采用的机器博弈的方法和技术。文献[10]研究了 k 子棋公平性、复杂度等问题,并首先提出六子棋。文献[6]研究了六子棋的模式、搜索算法等议题。六子棋远比五子棋复杂,已经成为机器博弈新的挑战性问题。

1.3 主要研究内容

六子棋的评估函数大致分为两类，基于棋型的评估函数与基于路的评估函数，各自有其优缺点。基于棋型的评估函数规定了六子棋中常见的几种棋型，六子棋的每种局面都是六子棋棋型的组合。这样通过把局面分割成棋型就可以评估出当前局面的态势。基于棋型的评估函数可以较为准确的评估局面，但棋型分隔起来比较复杂且耗时，没有一种很好的解决方案。基于路的评估函数把局面拆解成不同的路，拆解起来较为方便，但由于一组路只用一组参数经行评估，不利于应付不同的局面。

因此本文以六子棋为研究对象，研究的重点是六子棋的评估函数，研究的主要内容有：

1) 本文在基于路的基础上提出了一种新的双参数评估函数，提出了不同局面使用不同组参数的新想法，初步解决了应付不同局面的问题。同时把基于全局路的评估函数与基于局部路的评估函数相结合，保证了评估的效率。文章 3.1 节主要阐述基于路的评估函数，详细描述了基于路的全局扫描与局部扫描的方法思路、优缺点以及实现方式。3.2 节本主要阐述双评价参数评估函数的设计思路。包括不同参数设置决定了倾向性的不同、局面的判定方式和全局扫描与局部扫描相结合的方式。3.3 节主要为设计的实验。本文为验证双评价参数评估函数的合理性，设计了棋局局面倾向实验、估值准确性实验和棋力水平实验。局面倾向性实验证明不同实验参数设置会导致不同倾向的说法。估值准确性实验证明在某些局面下，双评价参数评估函数对比传统基于路的评估函数能够更准确的判断当前局面，能更准确地引导棋局。棋力水平实验证明了使用双评价参数评估函数对比传统基于路的评估函数有着更高的胜率。此外由于评估函数对搜索效率有着很大的影响，设计了搜索效率实验对比两种评估函数的评估效率。

2) 设计实现了 Executor 六子棋程序，程序包括开局库、搜索引擎、走法生成器、评估函数、hash 表等功能模块。详细阐述了模块的构架关系，以及阐述了模块各自的功能的实现细节。

1.4 论文结构

第一章 分析机器博弈与六子棋的研究背景和意义，介绍了机器博弈与六子棋的研究现状，介绍了本文的研究内容。

第二章 基本阐述机器博弈相关的研究内容和理论基础。详细阐述了六子棋的背景与相关理论基础，阐述了六子棋研究的几个研究点，如：走法生成器、评估函数、搜索技术与开局库等。介绍了基础的六子棋现有算法，从而为能更好的理解后面的内容打下基础。

第三章 突出了本文研究的重点，即六子棋的评估函数，大致描述了六子棋评估函数的作用与现有的方法。六子棋目前评估函数大致分为两种，即基于棋型的评估函数和基于路的评估函数。重点描述了基于路的评估函数，展示出现有的方法与不足。在此基础上提出了多参数对应多局面的思想，提出了双参数评估函数。设计、实现了实验并展示出实验数据。

第四章 设计和实现六子棋程序，展示六子棋程序组成模块，展示上述基于路的双评价参数评估函数的应用效果。

结论 总结了本文研究的意义及方法，分析了研究中的不足之处，指出了该课题继续研究的方向。

第2章　六子棋理论基础

本文主要研究的项目为六子棋，重点放在六子棋评估函数的研究，在此之前需要了解六子棋的相关知识，以及现有的研究内容。本章的主要内容为六子棋相关理论基础。

2.1　六子棋基本概念

(1)k 子棋相关概念

k 子棋是一族游戏的统称，由 connect(m，n，k，p，q)表示，其中 m，n 分别表示棋盘的长宽；k 表示在(水平的、垂直的、对角线方向的)任何一条线上，能率先连成 k 子的那一方获胜，如果双方均无法取胜，判定为和棋；p、q 表示第一步棋由黑方先落 q 子，然后双方轮流落 p 子。

六子棋与五子棋同属于 k 子棋，前者由后者发展而来，且比后者更加复杂。六子棋可以用 connect(19,19,6,2,1)来表示。目前，六子棋已经替代了五子棋在 k 子棋中的地位，并逐渐成为一种新的挑战性的机器博弈问题。六子棋有如下显著特点；1）平均分枝因子大。普通的博弈树搜索的深度太浅，在一定程度上抑制了搜索的作用。2）开局、中局、残局的策略几乎没有明显差异。3）相对来说，六子棋的特征比围棋、象棋等棋类更明显，更容易提取出来。

(2)六子棋的基本规则

六子棋产生自五子棋，规则也与没有禁手的五子棋类似，主要内容规定如下：六子棋有黑白双方，黑方先落子。黑棋第一步先落一子，随后黑白双方轮流落两子，率先连成六子在一条直线的一方获胜，如果双方均无法取胜，判定为和棋。

(3)六子棋的复杂度

对于机器博弈来说，一般采用 state-space 复杂度和 game-tree 复杂度来衡量某种博弈游戏的复杂程度。目前采用的是围棋的 19 路棋盘，其 state-space 复杂度可达 10^{172}，与围棋相当；其 game-tree 复杂度（以 30 手计算），可达 $(300*300/2)^{30}$~10^{140}，远大于五子棋，与象棋相当。目前，由于人们对六子棋的逐渐了解和熟练，通常可以行棋到 40 多手，由于六子棋无围棋的提子或象棋的吃子等行为，因此，棋盘上可以达到 160 多颗棋子，此时 game-tree 复杂度可高达 $(300*300/2)^{40}$~10^{188}，这个复杂度就远大于象棋的复杂度。因为象棋的规则比较详细、条款比较多，随着行棋进程，其棋盘棋子数目也越来越少，因此，象棋中局之的 game-tree 复杂度，就远小于六子棋[39]。表 2-1 是各种棋类的 game-tree 复杂度比较表，从表中可以看到，六子棋的复杂度介于日本将棋（一般认定是第二复杂的游戏）和象棋（一般认定第三复杂的游戏）之间，或者保守地说，应该与象棋差不多。

表格 2-1 棋类复杂度表

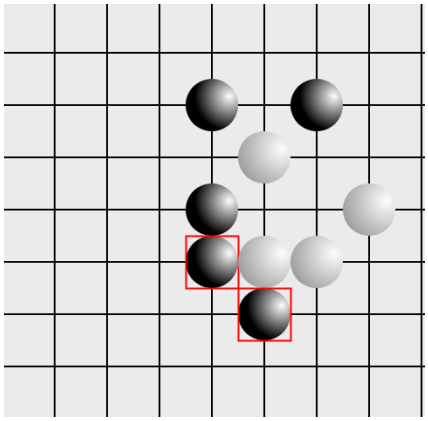
Table 2-1 State-space complexities and game-tree complexities of various games

棋类项目	state-space 复杂度	game-tree 复杂度
国际象棋	10^{46}	10^{132}
中国象棋	10^{48}	10^{150}
围棋(19*19)	10^{172}	10^{360}
围棋(15*15)	10^{105}	10^{70}
五子棋(15*15)	10^{105}	10^{70}
将棋	10^{71}	10^{226}
六子棋	10^{172}	10^{140}-10^{188}

2.2 六子棋棋盘表示

人类在下棋过程中，首先需要对棋盘的局面存储到脑子中，这时人类需要对棋盘上棋子的分布进行解析。六子棋程序需要类似的方法，对棋盘进行表示。六子棋棋盘表示的方法一般有比特棋盘、矩阵棋盘。其中矩阵表示相较于其他方法更能直观的表示棋局的状态，更容易理解，本文使用矩阵方法对棋盘进行表示。

矩阵方法使用二维矩阵表示棋盘，六子棋的棋盘使用 19*19 大小的棋盘，总共 361 个交叉点。矩阵方法使用 19*19 的二维数组对棋盘进行表示，数组每一个位置对应棋盘相应的位置，每个位置分别用 0、1、7 对应棋盘不同的状态。0 代表此位置上无字，1 代表有黑子，7 代表有白子。这里的 0、1、7 可以自行选取对应数值，0 与 1 常用来表示棋盘中的无子与黑子。使用 7 表示的原因是在之后设计基于路的评估函数时、可以简化路的判定。图 2-1 是之一局面下，矩阵方法的棋盘表示。



(a)

(b)

图 2-1 一种局面的矩阵表示

Figure 2-1 A position represented by matrix

2.3 博弈树

几乎所有的棋类问题，都可以用博弈树来描述。在六子棋中每一个局面对应一个节点，每一个局面可以衍生出许多子局面，即每一个节点可以衍生出许多子节点。再将子节点继续推演下去就和形成一颗博弈树，这棵树就反映了博弈双方博弈的所有状态。

图 2-2 是一棵博弈树的图示。假设博弈树一颗节点的子节点数为 n，深度为 h，则有 $1+n+n^2+n^3+\cdots+n^h=(n^h+1-n)/(n-1)$个结点，这个节点的数量是很大的，如果要对所有节点进行搜索，需要花很长的时间。但是人类却可以很快的做出选择。其实，人类棋手在下棋的过程中，在脑子里也生成了一棵博弈树，但不会把所有可能的情况全部生成，人类会很智能的把看起来不可能的节点或对己方很不利的节点直接过滤掉。在机器决策的过程中，这个过程通常通过搜索和剪枝的方法实现。

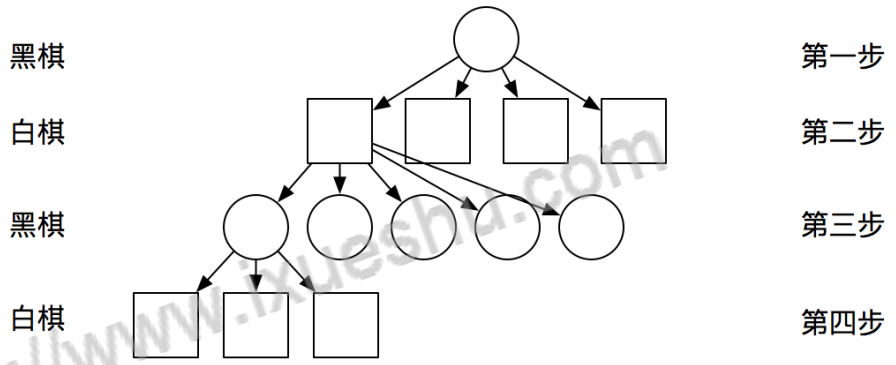


图 2-2 一颗博弈树

Figure 2-2 A game tree

2.4 评估函数

对于人类棋手来说，人们会根据当前棋局的状态，从脑中往下推演，对下面几种可能的情况进行评估，最终选择一条对自己最有利的走法。人们在下棋的过程中，对出现的局面会有一个大致的评估判断，人们会分辨出那种棋局对自己有利，那种棋局对自己不利。同样对于博弈机器人来说，也需要一个这样评估的方法，评估函数就是起着这样的作用。

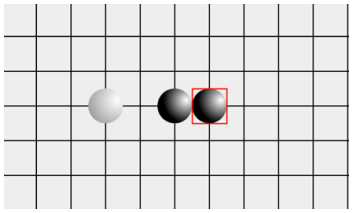
博弈机器人在下棋的过程中，也是经过比较才得到较好的落子位置。评估函数就是对棋局的状态给出一个评分，给出具体的数值从而根据数值的高低决策取舍。估值大小决定落子的位置，是产生着法的基础，其准确性将直接决定决策系统的“棋力”。

六子棋的评估函数大致分为两类，基于棋型的评估函数与基于路的评估函数，各自有其优缺点。基于棋型的评估函数规定了六子棋中常见的几种棋型，六子棋

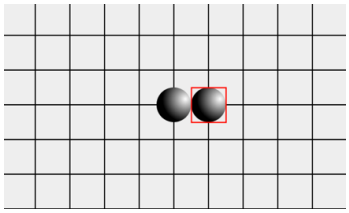
的每种局面都是六子棋棋型的组合。这样通过把局面分割成棋型就可以评估出当前局面的态势。基于棋型的评估函数可以较为准确的评估局面，但棋型分隔起来比较复杂且耗时，没有一种很好的解决方案。基于路的评估函数把局面拆解成不同的路，拆解起来较为方便，但由于一组路只用一组参数经行评估，不利于应付不同的局面。

2.4.1 基于棋型的评估函数

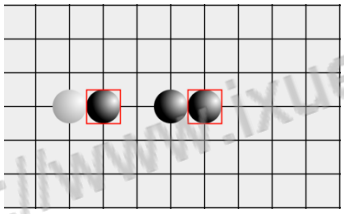
基于棋型的评估函数的整体思路为，把当前棋局分割为几种“棋型”，通过对“棋型”评分来对整个棋局进行评分。所谓“棋型”就是通过分析棋子间的关联关系判断出的相应模型，在六子棋中常见的棋型如图 2-3。



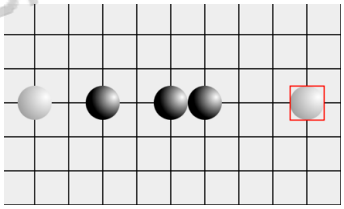
(a)眠 2



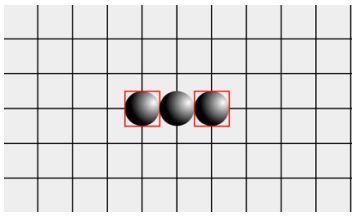
(b)活 2



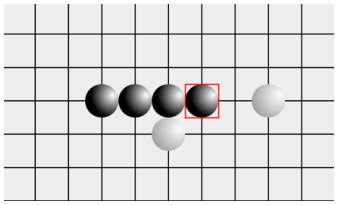
(c)眠 3



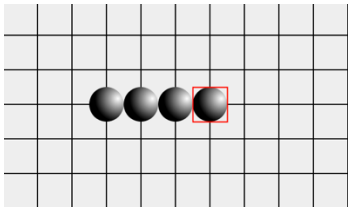
(d)朦胧 3



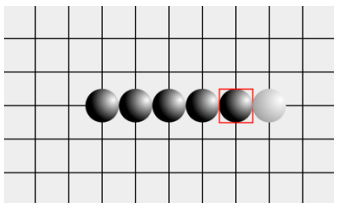
(e)活 3



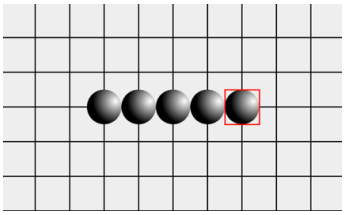
(f)眠 4



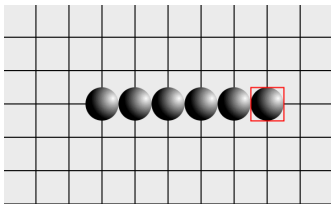
(g)活 4



(h)眠 5



(i)活 5



(g)6 连

图 2-3 六子棋棋型示例

Figure 2-3 Model of connect6

对棋型进行分割时，需要以走法的两个位置为中心，向 8 个方向进行扫描，扫描的距离为包括中心点的六个位置，如图 2-4。

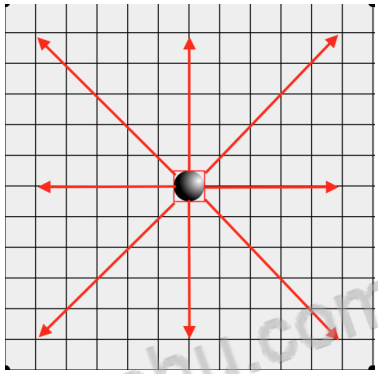


图 2-4 棋型扫描示意图

Figure 2-4 Scanning way of model

2.4.2 基于棋型的评估函数存在的问题

上述只列出了六子棋的部分棋型，六子棋中常见的棋形就有 19 种[37]，各种棋形又有多种情况。另外，当几种棋形交叉组合的时候，往往不好分割。如图 2-5。因此，分割和判断棋形就成为计算机系统最为头疼的难点。计算机要判定棋形，通常要从所支持的数据库、知识库中寻求模板。如果模板设置错误或棋形统计不完全，将严重的影响博弈系统的棋力。采用基于“路“的评估思想，计算机决策就不需要对复杂的棋型进行统计和区别，只需要根据”“路”的数量进行估值，大大简化了估值的复杂程度。

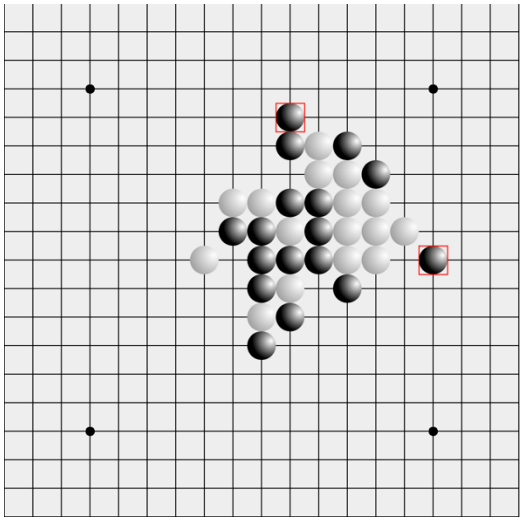


图 2-5 复杂局面

Figure 2-5 Complicated situation

2.5 搜索技术

当前节点下会有许多的子节点，即当前局面下会有多种走法。评估函数会对当前局面下多种走法进行评估得到走法的估值，然后通过估值对走法进行排序，再对子节点做相同操作，走法生成与估值。这个递归的过程就是搜索的过程。图 2-6 是搜索的示意图。每个节点下会有多个子节点，虽然需要对每个节点进行估值，但不是每个节点都有搜索的价值，所以在搜索中会指定估值靠前的一些节点进行接下来的搜索，这个数量就叫搜索宽度用 w 表示。树的高度表示搜索的高度，用 h 表示。此外之前的研究者研究出许多剪枝算法，提高搜索的效率。评估函数会对每个生成的节点进行评估，评估函数的效率会对搜索效率有着非常大的影响。

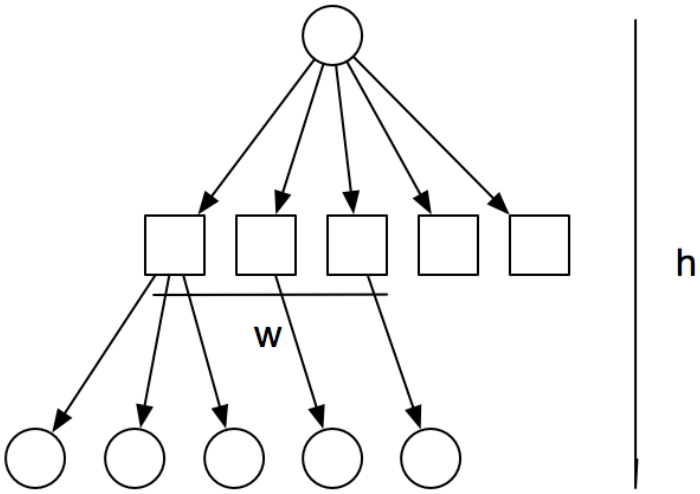


图 2-6 搜索示意图

Figure 2-6 Process of searching

2.6 本章小结

本章介绍了六子棋的基本理论，具体介绍了六子棋的规则，数据表示，以及博弈树。在此基础上介绍了评估函数，阐述了现有评估函数的缺点和不足。最后阐述了六子棋基础的搜索技术。

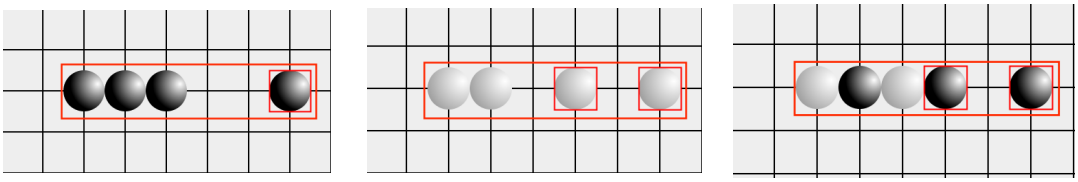
第3章　基于路的双评价参数评估函数

人们在下棋的过程中，对可能出现的局面在脑子里有一个评价，哪种局面对自己有利，哪种局面对自己不利。六子棋中评估函数具有相同的作用，可以对当前的局面进行估值，通过估值对当前局面有一个初步的评价。现有六子棋评估函数大致有两种：基于棋型的评估函数和基于路的评估函数。基于棋型的评估函数把当前局面拆解为各种棋型，每个棋型都有一个估值，通过统计各种棋型的数量，把各种棋型的估值乘以数量求和，就可以得到当前局面的估值。但棋型种类复杂，分割起来效率低下，没有一种很好的方法对棋型进行分割。传统基于路的评估函数，把局面拆解成各种路型。路型只有六种，拆解起来非常简单，但对路的参数设置较为单一，不能应对多种复杂局面。为解决不能应对多种局面的问题，本章在基于路的评估函数基础上，提出了双评价参数评估函数。

3.1　基于路的评估函数

3.1.1　“路”的定义

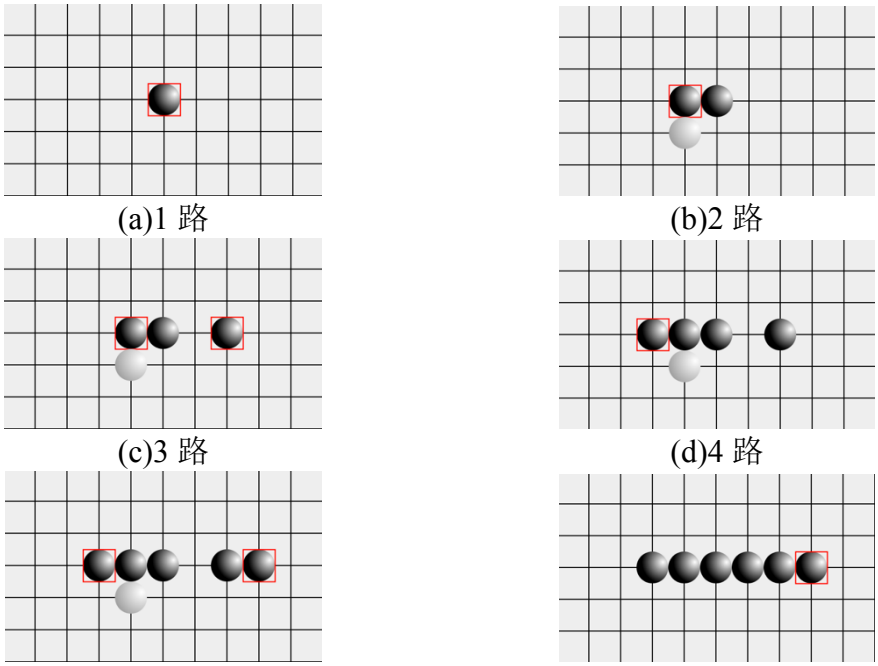
本文采用二维数组的方式对棋局的状态进行描述。在 19*19 的棋盘上会形成 361 个交叉点，每个交叉点分别有 3 种状态，无落子、有黑子、有白子，分别用 0，1，7 来表示。在棋盘上连续 6 个点能够连成一条直线，则称为路。如果路上所有棋子都为黑子，则这条路的颜色是黑色；如果该路上所有棋子都为白色，则这条路的颜色为白色；如果该路上没有棋子或同时含有 2 种颜色的棋子，则这条路没有颜色。在有色路上已经含有棋子的个数称为路的长度，并成为这条有色路的“几路”。图 3-1 为几种路的示例，其中(a)为黑棋的 4 路，(b)为白棋的 4 路，(c)为无颜色的路。图 3-2 为 6 种路型的示例。



(a)黑棋 4 路　　(b)白棋 4 路　　(c)无颜色的路

图 3-1 几种路的示例

Figure 3-1 Kind of roads



(a)1 路　(b)2 路

(c)3 路　(d)4 路

(e)5 路　(f)6 路

图 3-2 路型示例

Figure 3-2 Model of road

按照横向、纵向、左斜、右斜 4 个方向的特点和 “路”的定义，可以通过计算得到不同方向的路数目如下：①横向上，19 行 × (19 － 6 + 1)路/行 = 266 路；②纵向上，19 列 × (19 － 6+1)路/列 =266 路；③左斜方向上，14 行 × (19 － 6+1)路/行 =196 路。如图 2；④右斜方向上，14 列 × (19 － 6 + 1)路/列 = 196 路。因此，在 19×19 围棋棋盘上，总共有 266 × 2 + 196 × 2 = 924(路)。图 3-3 为横向扫描的示例图，图 3-4 为四个方向扫描的示例图。

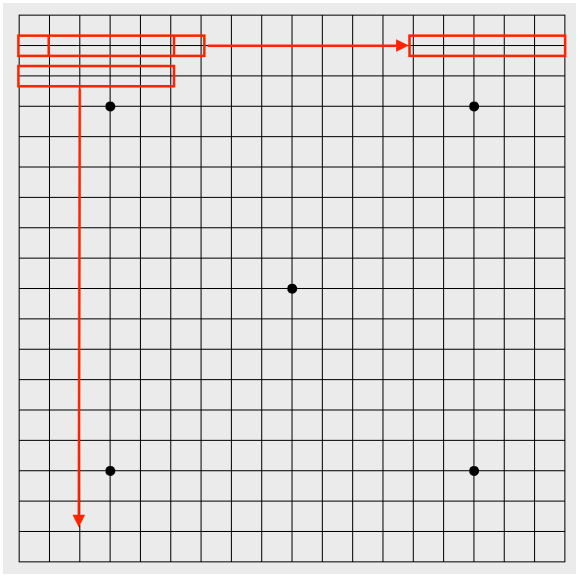


图 3-3 水平方向路的示意图

Figure 3-3 Horizon scanning way of road

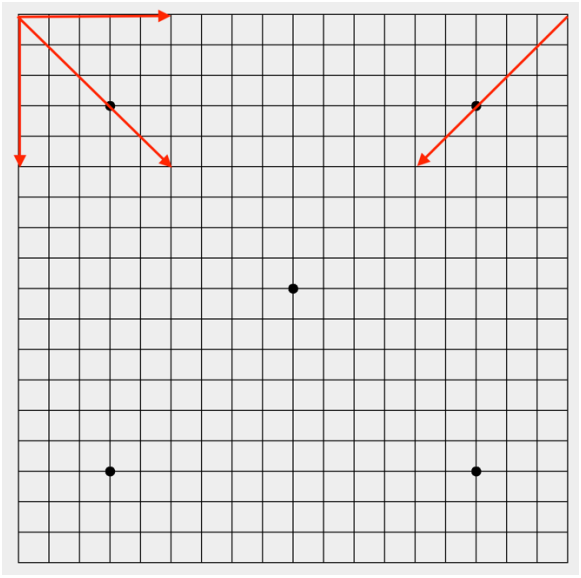


图 3-4 扫描的四个方向示意图

Figure 3-4 Four scanning direction

3.1.2 基于路的全局扫描评估函数

采用“路”思想以后，对棋局状态的评估就不再进行棋形的判定，只要对博弈双方“路”的情况进行统计和计算。在评估过程中只要将双方的路的状况记录下来再乘以相应的权重并求和就可以得到对棋盘的评估。本文采用 analysisHorizon, analysisVertical, analysisLeft, analysisRight 四个函数分别对棋盘的横向，纵向，左斜方向，右斜方向四个方向进行路的扫描。其中横向与纵向相类似，左斜与右斜相类似。以横向与右斜为例：横向扫描过程为遍历横向上各路的起点（总共有 19×（19 － 6 + 1)=266 个起点），以每个起点为开始向右扫描 6 个位置，即扫描这一路上 6 个位置，检查这 6 个位置上的状态为白子、黑子、还是无子。然后根据路的定义判断这条路的颜色。得到的结果分别用 numberOfMyRoad[1-6]与 numberOfEnemyRoad[1-6]两个数组进行存储，分别代表己方与对方每种路的个数。图 3-3 为横向扫描的示意图，analysisHorizon 主要逻辑代码如下所示。

```
private int analysisVertical(byte[][] position, int type) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS - 5; j++) {
            int number = position[i][j] + position[i][j+1] + position[i][j+2]
                    + position[i][j+3] + position[i][j+4] + position[i][j+5];
            if (number == 0 || (number > 6 && number % 7! = 0)) {
                continue;
            }
            if (number < 7) {
                if (type == BLACK) {
                    numberOfMyRoad[number]++;
                } else {
```

```
                        numberOfEnemyRoad[number]++;
                    }

                } else {
                    if (type == BLACK) {
                        numberOfEnemyRoad[number/7]++;
                    } else {
                        numberOfMyRoad[number/7]++;
                    }

                }
            }
        }
    }
```

上面代码中参数 position 为一个 2 维数组，用来记录当前棋盘的状态，数组值用 1 代表黑棋、7 代表白棋。Type 为当前执棋方的颜色分为 BLACK 和 White。COLS、ROWS 分别对应棋盘列和行的最大值 19，number 为这条路上所有值的加和。判断路的颜色的方式为：

- 路为无色 number = 0 或 (number > 6 且 number%7 ！= 0)
- 路为黑色 number <= 6
- 路为白色 number 为其他值

analysisVertical 与 analysisHorizon 类似，即把扫描横向改为扫描纵向。analysisLeft 与 analysisRight 相类似，二者扫描的路数为 14×（19 － 6＋1）=196 路，右斜扫描如图 3-5。

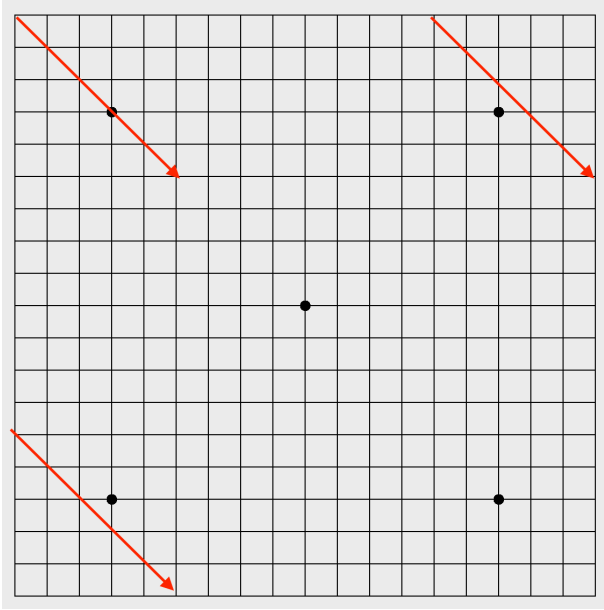


图 3-5 右斜扫描示意图

Figure 3-5Right diagonal scanning direction of road

除了拆解出敌我双方 1-6 路的数量之外，还要确定 1-6 路所对应的威胁值，这个威胁值表示不同路型所对应的价值。这个价值敌我双方应该是分别设置的。地方威胁值越高，我方威胁值越低，我方的行棋位置会更趋向于破坏敌方棋型，也就是会更趋向于防守，反之则会趋向与进攻。也就是说行棋进攻和防守的趋向性可以通过改变参数来调控。敌我双方路的威胁值分别用数组 scoreOfMyRoad[1-6]与 scoreOfEnemyRoad[1-6]来表示。假设“路“中存在六颗棋子，既存在某一方的 6 路，则棋局结束。所以 scoreOfMyRoad[6]与和 scoreOfEnemyRoad[6]应该赋予一个很大的数，以此代表 6 路的价值很大以及棋局的完结。同样的 scoreOfMyRoad[1-5]与 scoreOfEnemyRoad[1-5]也应该赋予与其价值相符合的数值。现阶段参数的设置是通过经验确定的，表 3-1 是一组参数示例。

表格 3-1 参数示例表

Table 3-1 An example of parameters

路数	scoreOfMyRoad	scoreOfEnemyRoad
1 路	1	1
2 路	5	10
3 路	10	15
4 路	25	35
5 路	25	25
6 路	10000	10000

根据上述思想，对具体棋局的估值 Score 可由式（3-1）计算获得：

$$\mathrm{Score} = \sum_{i=1}^{6}(\mathrm{NumberOfMyRoad[i]} * \mathrm{ScoreOfMyRoad[i]}) - \sum_{i=1}^{6}(\mathrm{NumberOfEnemyRoad[i]} * \mathrm{ScoreOfEnemyRoad[i]}) \quad (3\text{-}1)$$

计算 Score 的主要逻辑代码如下，下面代码主要为对四个方法的调用，以及 Score 的计算。

```
private int calScore(byte[][] position, int type) {
        int score = 0;
        int condition = 0;
        if (baseScore > ALERTSCORE) {
            condition = 1;
        }

        for(int i = 1; i < 7; i++) {
            numberOfEnemyRoad[i] = 0;
            numberOfMyRoad[i] = 0;
        }

        analysisHorizon(position, type, move);
        analysisVertical(position, type, move);
        analysisLeft(position, type, move);
```

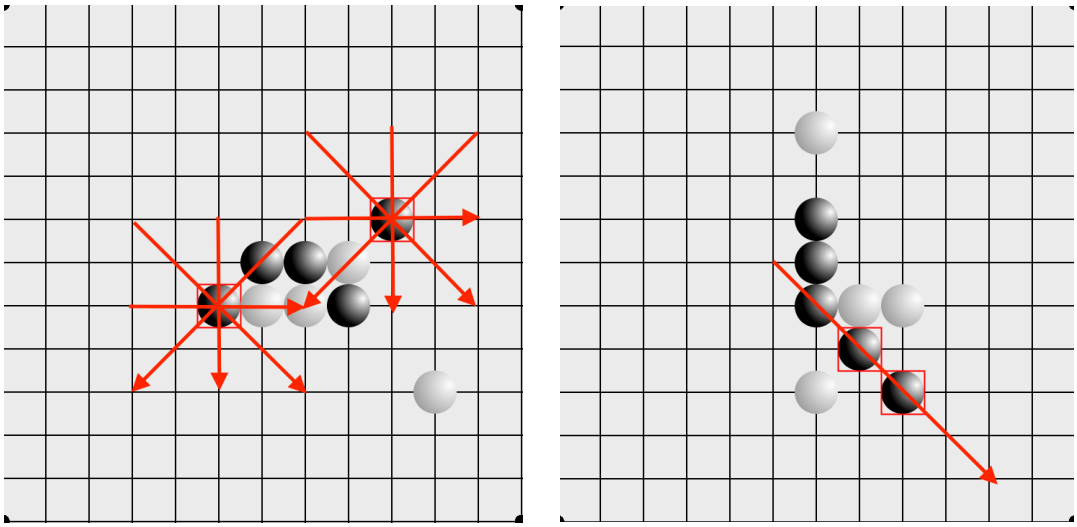
```
            analysisRight(position, type, move);

            for(int i = 1; i < 7; i++) {
                score += (numberOfMyRoad[i] * scoreOfMyRoad[condition][i] -
                        numberOfEnemyRoad[i] *
scoreOfEnemyRoad[condition][i]);
            }

            return score;
        }
```

3.1.3 基于路的局部扫描评估函数

基于“路”的全局扫描评估方式相较于基于棋型的评估函数虽然简化了局面拆解的复杂程度，由 3.1.2 可知每次扫描时需要扫描整个棋盘的四个方向上的所有路，总共 924 路，这也是不小的开销，并且大部分路并没有棋子，对局面的评估没有任何帮助。经分析得知棋局状态的变化均由新增棋子引起，而新增的棋子只影响以它为基点的水平方向，垂直方向，左斜方向，右斜方向四个方向上的棋型，对其他位置并无影响。对于路来说新增棋子也只影响了这四个方向上敌我双方“路”的数量。



(a) (b)

图 3-6 基于路的局部扫描方式示意图

Figure 3-6 An example of local scanning way of road

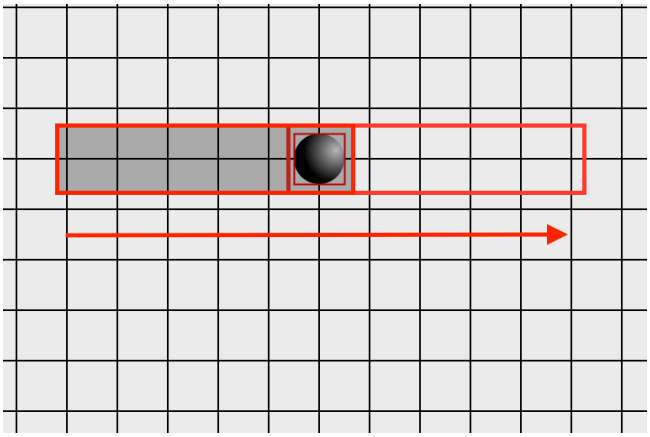


图 3-7 水平方向上基于路的局部扫描方式

Figure 3-7 Local scanning way of road on horizon direction

如图 3-6(a),用红方框选中的两颗棋子是最新的落子，这两颗落子只影响了与这两颗棋子有关的 4 个方向上的路。总的路数为：6 路/方向 * 4 方向/落子 * 2 落子 = 48 路。基于这种思想产生了基于“路”的剧部扫描方式，具体方法为统计落子前这 48 条路的情况，让后再统计落子后这 48 条路的情况，最后让之后的状况的估值减去之前状况的估值即为这两个子的估值。使局部扫描棋局估值为$V_{total}^{'}$，落子前估值为$V_{pre}^{'}$，落子后估值为$V_{after}^{'}$，计算$V_{total}^{'}$的公式为 3-2。图 3-6(b)所展示的两颗棋子在右斜方向上有重叠的部分，这个部分在对路进行统计时可能会重复统计，需要进行去重处理。

$$V_{total}^{'} = V_{after}^{'} - V_{pre}^{'} \tag{3-2}$$

局部路扫描也有 analysisHorizon, analysisVertical, analysisLeft, analysisRight 四个方法，与全局扫描的四个方法相比略微不同，其中仍以 analysisHorizon 举例，代码如下。

```
private int analysisHorizon(byte[][] position, int type, PointMove move) {
        if (move.getPoint1() != null) {
            int x = move.getPoint1().getX();
            int y = move.getPoint1().getY();
            for (int i = x-5>0?x-5:0; i <= x && i + 5 < 19; i++) {
                int number = position[i][y] + position[i+1][y] + position[i+2][y]
                        + position[i+3][y] + position[i+4][y] + position[i+5][y];
                if (number == 0 || (number > 6 && number % 7 != 0)) {
                    continue;
                }
                if (number < 7) {
                    if (type == BLACK) {
                        numberOfMyRoad[number]++;
                    } else {
                        numberOfEnemyRoad[number]++;
                    }

                } else {
```

```
                        if (type == BLACK) {
                            numberOfEnemyRoad[number/7]++;
                        } else {
                            numberOfMyRoad[number/7]++;
                        }

                    }
                }
            }
            if (move.getPoint2() != null) {
                int x = move.getPoint2().getX();
                int y = move.getPoint2().getY();
                for (int i = x-5>0?x-5:0; i <= x && i + 5 < 19; i++) {
                    int xx = move.getPoint1().getX();
                    if (i == xx || i+1 == xx || i+2 == xx || i+3 == xx || i+4 == xx || i+5
== xx) {//去重复
                        continue;
                    }
                    int number = position[i][y] + position[i+1][y] + position[i+2][y]
                            + position[i+3][y] + position[i+4][y] + position[i+5][y];
                    if (number == 0 || (number > 6 && number % 7 != 0)) {
                        continue;
                    }
                    if (number < 7) {
                        if (type == BLACK) {
                            numberOfMyRoad[number]++;
                        } else {
                            numberOfEnemyRoad[number]++;
                        }

                    } else {
                        if (type == BLACK) {
                            numberOfEnemyRoad[number/7]++;
                        } else {
                            numberOfMyRoad[number/7]++;
                        }

                    }
                }
            }

}
```

与全局相比的不同点是需要多传入一个参数 move 代表这一次行棋，move 中包含两步 point1 和 point2 表示这一步的两个位置。有了 point1 和 point2 就可以确定 48 条路的位置。其中 if (i == xx || i+1 == xx || i+2 == xx || i+3 == xx || i+4 == xx || i+5 == xx)这句的作用为去重，既在计算 point2 的路时，如果这条路上包含点 point1，说明这条路在扫描 point1 是已经被包含进去，要跳过这条路。

下面的方法 evaluate 表示局部估值的计算过程，calScore 方法与上面所介绍的相同，fScore 为放置两子前的估值，bScore 为放置两子后的估值，最终估值为二者的差值，makeMove 方法与 unMakeMove 方法分别的作用为放置两子和撤销放置。

```
public int evalute(byte[][] position, int type, PointMove move) {
        int fScore = 0;
        int bScore = 0;

        fScore = calScore(position, type, move);
        makeMove(position, move, type);
        bScore = calScore(position, type, move);
        unMakeMove(position, move);

        return bScore - fScore;
    }
```

3.2 基于路的双评价参数估值方式

3.2.1 估值参数对棋局的影响

在基于“路”的扫描方式中，双方 1-6 路的估值选取是很重要的一环。多套不同参数的选择带来的局面效果是不同的，既不同 scoreOfMyRoad[1-6]和 scoreOfEnemyRoad[1-6]参数值的选取会使棋局选择的走势呈现不同的状态。例如，当 scoreOfMyRoad 各路的估值大于相对应 scoreOfEnemyRoad 路的估值时（ scoreOfMyRoad[1] > scoreOfEnemyRoad[1], scoreOfMyRoad[2] > scoreOfEnemyRoad[2],……），在敌我双方路的情况相同的条件下，估值函数会对我方局面给与更高的评分，呈现出的效果是计算机落子的选取会更具进攻性，即选子更倾向于组成自己的路。与其相对应的是当 scoreOfMyRoad 各路的估值小于相对应 scoreOfEnemyRoad 路的估值时，计算机落子的选取会更具防守性，即选子更倾向于破坏对方的路。在棋局中敌我双方的局面态势是时常发生变化的，仅适用固定的 scoreOfMyRoad 与 scoreOfEnemyRoad 估值无法灵活的应对不同的局面。但无论是基于“路”的全局扫描方式还是基于“路“的局部扫描方式使用的仅仅是固定的一套估值参数。因此本文在基于”路“的扫描方式基础上提出了双评价参数扫描方式，分别应对不同局势下，不同的走法选择。

3.2.2 基础局面估值

由 3.1.2 与 3.1.3 可以看出，全局扫描方式属于全量性质的。所谓全量性质就是全局扫描方式需要对整个棋盘进行扫描，得到的估值是对全局整体局面的一个评价。局部扫描方式属于增量性质的。所谓增量性质就是局部扫描方式只关心两

个落子周围棋局的形式，得到的估值体现的是两颗子的价值。增量性质的方法，只能反映出当前落子对棋局落子前后估值的影响，而不能反映出整盘棋局形势的状态，而全量性质的方法可以很好地反映出当前局面下敌我双方形势状态。

双评价参数评估函数的整体思路为，判断当前局面对己方的情势来选择哪个不同的应对方式。目前的应对方式为进攻和防守两种，当局面对己方有利时选择的走法更加偏进攻性，对己方不利时选择的走法更加片防守性。这就需要先确定己方的局面的利弊状况。采取的方式是计算当前整体局面的估值，用 baseScore 表示，另外定义一个警戒值 alertScore。当 baseScore > alertScore 时表示我方局面具有优势，可以选择进攻性较强的落子策略；当 baseScore < alertScore 时表示我方处于不利局面，应选择防守性较强的落子策略。AlertScore 没有一个比较好的确定方式，目前只根据经验确定。BaseScore 反应的是一个整体的局面评估，因此使用全局扫描的方式计算 baseScore 较为合理。

BaseScore 的计算规则与全局扫描规则相同，计算公式为：

$$\text{baseScore} = \sum_{i=0}^{6}(\text{NumberOfMyRoad[i]} * \text{ScoreOfMyRoad[i]}) - \sum_{i=0}^{6}(\text{NumberOfEnemyRoad[i]} * \text{ScoreOfEnemyRoad[i]}) \quad (3\text{-}3)$$

进攻与防守的倾向性则由选择不同的参数设置来实现。

3.2.3 基于路的全局扫描与局部扫描相结合

由 3.1.2 可知全局扫描方式虽然可以反映出棋局敌我双方的形势，但需要扫描整个棋盘的 924 路，扫描效率不高。由 3.1.3 可知局部扫描的方式具有很高的扫描效率。本文在计算 baseScore 时使用了全局的扫描方式，而在对每个生成的局面进行评估时使用更加高效的局部扫描方式。评估的过程为：1.首先通过全局扫描方式得到 baseScore 值，以此来得到棋局大致的形势。2.通过对局势的判断以此判断使用哪组估值参数。3.使用确定的估值参数，通过局部扫描的方法增量地更新 baseScore 值。4.重复 2、3 步直到搜索深度为 0。图 3-5 为整体计算的流程。计算的代码如下：

```
public PointMove searchAGoodMove(byte[][] position, int type) {
        TranspositionTable.calculateInitHashKey(position);
        int baseScore = evalution.evalute(position, type);
        this.position = position;
        alphaBeta(maxDepth, type, -200000, 200000);
        return bestMove;
}

public int evalute(byte[][] position, int type) {
        baseScore = Integer.MIN_VALUE;
        baseScore = calScore(position, type);
        return baseScore;
    }
```

```
private int calScore(byte[][] position, int type, PointMove move) {
        int score = 0;
        int condition = 0;
        /***********这一部分只在双评价参数评估函数中出现
        if (baseScore > ALERTSCORE) {
            condition = 1;
        }
        ***********/

        for(int i = 1; i < 7; i++) {
            numberOfEnemyRoad[i] = 0;
            numberOfMyRoad[i] = 0;
        }

        analysisHorizon(position, type, move);
        analysisVertical(position, type, move);
        analysisLeft(position, type, move);
        analysisRight(position, type, move);

        for(int i = 1; i < 7; i++) {
            score += (numberOfMyRoad[i] * scoreOfMyRoad[condition][i] -
                    numberOfEnemyRoad[i] *
scoreOfEnemyRoad[condition][i]);
        }

        return score;
    }
```

searchAGoodMove 方法是整体的流程，其中会调用评估函数的 evaluate 方法得到 baseScore，这里的评估函数是全局扫描评估函数。calScore 方法的作用为计算估值，计算的过程与 3.1.2 和 3.1.3 相同。

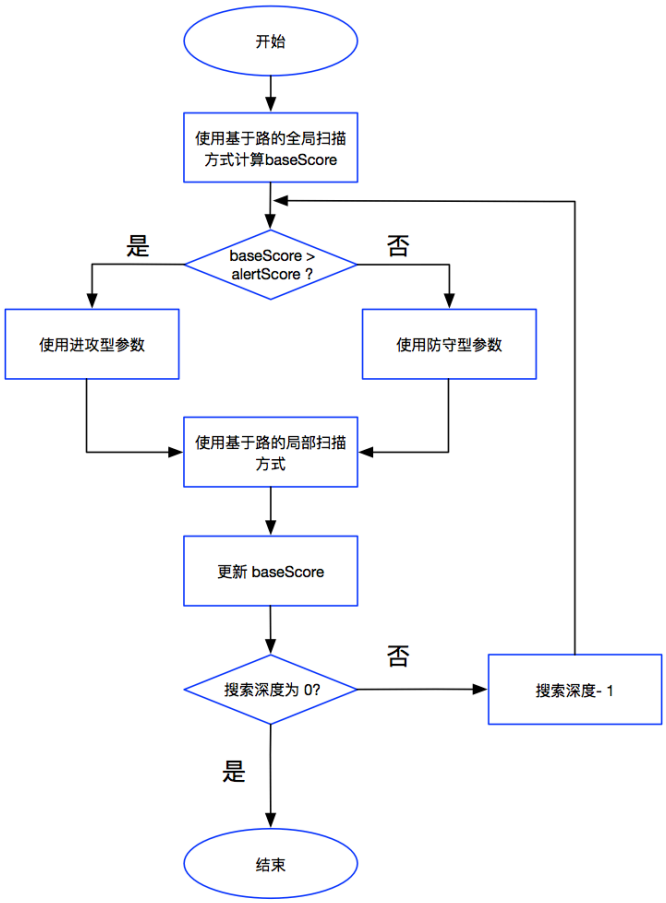


图 3-8 全局扫描与局部扫描结合流程图

Figure 3-8 The process of combination of complete scanning way and local scanning way

3.3 实验及结果

3.3.1 实验环境

实验的硬件环境为：

笔记本品牌：Apple MacBook Pro

操作系统：MacOS

CPU：2.2 GHz Intel Core i7

RAM：1600 MHz DDR3　大小 16G

3.3.2 实验构思

本章主要研究六子棋的评估函数，评估函数的估值是否准确，按照评估函数得出的估值是否可以引导棋局走向与我方有利的局面，是评价一个评估函数好坏的主要内容。对此方面本文设置了局面倾向性试验，估值准确性实验和棋力水平实验验证双参数评估函数的合理性。局面倾向性实验验证 3.2.1 种指明的攻击倾向性和防守倾向性问题，估值准确性实验验证在某些局面下双参数评估函数比传统基于路的评估函数评估局面更加准确，棋力水平实验验证使用双参数评估函数比传统评估函数在胜率上有所提高。此外由于对每一个生成出来的走法都要进行

评估，评估的节点数量很大，评估函数的效率也极大地影响了搜索的效率。本文设置了搜索效率实验，验证双参数评估函数的评估效率。最后设置了参数影响实验，试探性地观察参数改变对评估函数的影响。

3.3.3 棋局局面倾向实验

本小结实验主要验证参数对估值的影响。如 3.2.1 所阐述，当 scoreOfMyRoad 各路的估值大于相对应 scoreOfEnemyRoad 路的估值时，我方选取的落子位置更具进攻性，反之我方选取的落子位置更具防守性。

实验使用基于路的局部扫描方式作为评估方法，另设置了两组不同的参数。实验设置如表 3-2、表 3-3 和表 3-4。

表格 3-2 实验评估函数设置

Table 3-2 Experimental evaluation function setting

评估函数	设置
评估函数 1	基于路评估函数+局部扫描

表格 3-3 进攻倾向参数表

Table 3-3 The table of parameters of attacking bias

路数	己方路的价值	对方路的价值
1 路	1	1
2 路	10	10
3 路	15	15
4 路	350	35
5 路	250	25
6 路	10000	10000

表格 3-4 防守倾向参数表

Table 3-4 The table of parameters of defending bias

路数	己方路的价值	对方路的价值
1 路	1	1
2 路	10	10
3 路	15	15
4 路	35	350
5 路	25	250
6 路	10000	10000

表 3-2 表明试验所使用的评估函数为基于路的局部扫描评估函数。表 3-3 展示的是进攻倾向参数表设置，表 3-4 展示的是防守倾向参数表设置。对比表 3-3 和 3-4 可知，进攻倾向时设置己方 4 路、5 路的价值为对方的 10 倍，其他参数值相同；防守倾向时设置对方 4 路、5 路的价值为己方的 10 倍，其他参数值相同。这样设置参数是一种定性的设置，这样设置足以说明倾向性问题。用两组参数分

别对相同的一组局面进行评估，观察相同局面下不同参数设置选点的结果有何不同，结果如图 3-6。

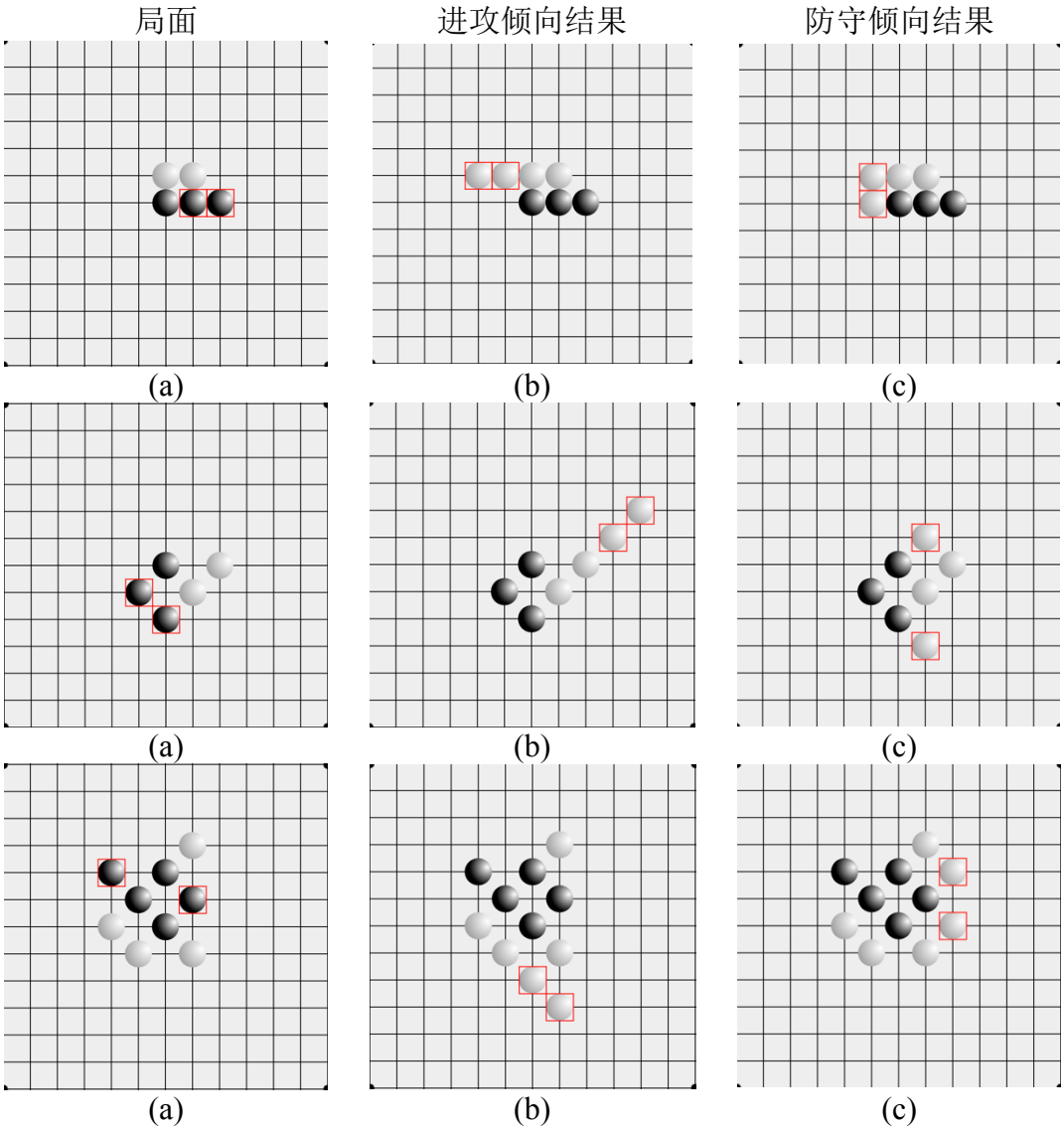
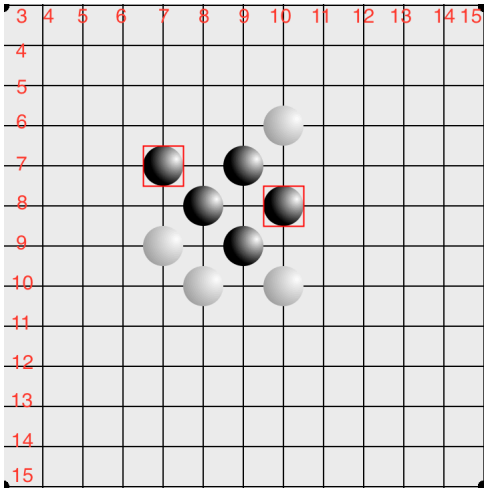


图 3-9　3 组不同局面倾向选点图

Figure 3-9 Three group of evaluation result pictures on different position

如图 3-9 为 3 组不同局面下，不同倾向的选点结果，由 3 组图可知进攻倾向的参数设定使选点更具进攻性，防守倾向的参数设定使选点更具防守性。用第 3 组局面为例，展示不同倾向参数设置下，评估值前 10 的走法。如图 3-10。



(a)

```
点1: x:10 y: 12 点2: x:9 y: 11 估值: 1139
点1: x:6 y: 8 点2: x:9 y: 11 估值: 1138
点1: x:9 y: 11 点2: x:6 y: 8 估值: 1138
点1: x:9 y: 11 点2: x:10 y: 12 估值: 1134
点1: x:9 y: 10 点2: x:7 y: 10 估值: 1115
点1: x:9 y: 10 点2: x:11 y: 10 估值: 1115
点1: x:5 y: 7 点2: x:6 y: 8 估值: 1103
点1: x:7 y: 10 点2: x:9 y: 10 估值: 1101
点1: x:6 y: 8 点2: x:5 y: 7 估值: 1094
点1: x:11 y: 10 点2: x:9 y: 10 估值: 1088
```

(b)

```
点1: x:11 y: 7 点2: x:11 y: 9 估值: 214
点1: x:10 y: 12 点2: x:9 y: 11 估值: 194
点1: x:11 y: 9 点2: x:11 y: 7 估值: 194
点1: x:6 y: 8 点2: x:9 y: 11 估值: 193
点1: x:9 y: 11 点2: x:6 y: 8 估值: 193
点1: x:9 y: 11 点2: x:10 y: 12 估值: 189
点1: x:12 y: 10 点2: x:12 y: 8 估值: 189
点1: x:9 y: 8 点2: x:11 y: 9 估值: 187
点1: x:12 y: 8 点2: x:11 y: 9 估值: 187
点1: x:11 y: 10 点2: x:11 y: 9 估值: 186
```

(c)

图 3-10　不同倾向下估值前 10 的走法

Figure 3-10 Top 10 of evaluation value of move for different position

图 3-10 (a)为局面图，(b)为进攻倾向下估值前 10 的点的坐标，(c) 为防御倾向下估值前 10 的点的坐标。其中 x 为横向坐标，y 为纵向坐标。把坐标对应到棋盘上可知进攻倾向下估值前 10 的点进攻性较强，防御倾向性下估值前 10 的点防御性较强。其原因为进攻倾向下，我方 4、5 路威胁值较高，形成 4、5 路时的估值较高；防御倾向下，对方 4、5 路威胁值较高，形成 4、5 路时的估值因为是负数的原因较低。

根据上述实验得出结论，参数设置的不同会导致倾向性的不同。

3.3.4　估值准确性实验

本小结主要验证估值的准确性。估值准确性可以笼统的概括为对我方有利的点应该给一个较高的估值，对我方越有利估值越高。同样也包括破坏对方有利的点。实验设计为使用传统基于路的局部扫描评估函数与基于路的双评价参数评估函数对同一局面进行评估，观察评估结果与选点的准确性。实验设置如表 3-5、3-6、3-7。

表格 3-5 实验评估函数设置

Table 3-5 Experimental evaluation function setting

评估函数	设置
评估函数 1	基于路评估函数+局部扫描

评估函数 2	基于路评估函数+双参数

表格 3-6 基于路局部扫描参数表

Table 3-6 The table of parameters of attacking bias

路数	己方路的价值	对方路的价值
1 路	10	1
2 路	100	10
3 路	150	15
4 路	350	35
5 路	250	25
6 路	10000	10000

表格 3-7 基于路的双参数评估函数进攻倾向参数表

Table 3-7 The table of parameters of attacking bias

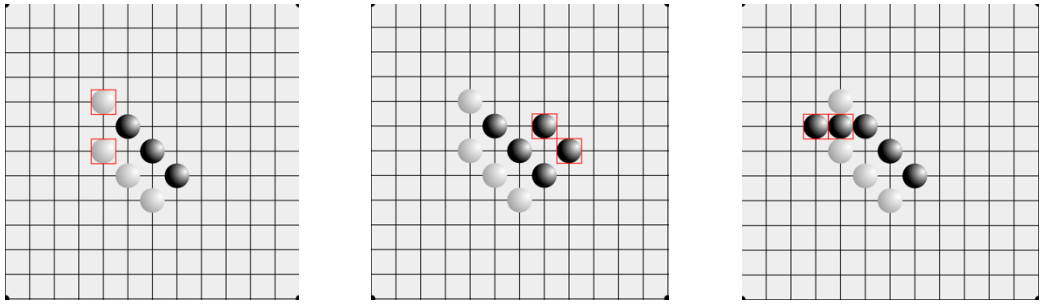
路数	己方路的价值	对方路的价值
1 路	10	1
2 路	100	10
3 路	150	15
4 路	350	35
5 路	250	25
6 路	10000	10000

表格 3-8 基于路的双参数评估函数防守倾向参数表

Table 3-8 The table of parameters of defending bias

路数	己方路的价值	对方路的价值
1 路	1	10
2 路	10	100
3 路	15	150
4 路	35	350
5 路	25	250
6 路	10000	10000

如上的参数设置，使传统基于路局部扫描与双评价参数进攻倾向参数相同。当双参数认定局面为进攻倾向时两种评估方法对局面的估值应相同，当双参数认定局面为防守倾向时，两种方法就会有不同的估值。实验使用的局面如图 3-11(a)。



(a)实验局面图　(b)评估函数 1 选点　(c) 评估函数 2 选点

图 3-11 两种评估函数选点对比

Figure 3-11 The comparison of choosing point of the two evaluation functions

在局面(a)处两种方法选点有了差异。图 3-11(b)是传统基于路局部扫描评估函数选点，图 3-11(c)是基于路双参数评估函数选点。此图中差异可以看出，(b)图只是摆出了自己的进攻棋型，并没有阻挡白棋形成活 4 棋型以及没有及时应对接下来的进攻。(c)图只是摆出了一个活 3 棋型，这两颗子与黑棋的其他子没有呼应关系，但黑棋破坏了白棋形成活 4 的机会。这种选点更偏于防守。

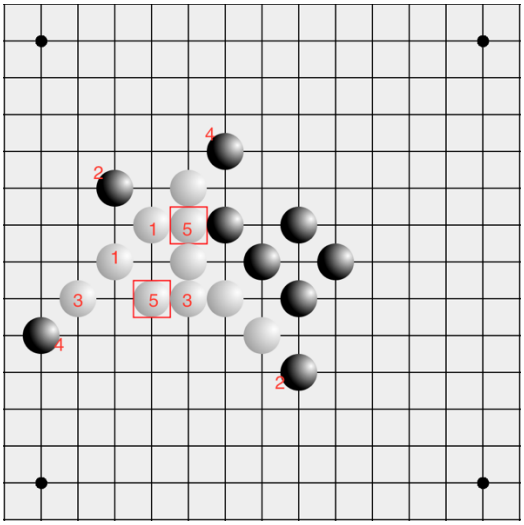


图 3-12 评估函数 1 选点演变

Figure 3-12 The evolution after choosing point by function1

图 3-12 展示了评估函数 1 选点后的几步演变，白棋可以通过迫着搜索获取胜利。这说明评估函数 1 没有及时发现黑棋的险境并加以应对。说明在一些情况下，评估函数 2 比评估函数 1 更能准确的对局面进行评估。

3.3.5 棋力水平实验

棋力水平实验主要验证双参数评估函数的实际应用的效果。实验主要过程是使两个使用不同评估函数的程序进行对弈，观察胜率，实验设置如下。

表格 3-9 实验设置

Table 3-9 Experimental setting

评估函数	设置
评估函数 1	基于路评估函数+局部扫描
评估函数 2	基于路评估函数+双评价参数

表格 3-10 基于路局部扫描参数表

Table 3-10 The table of parameters of local scanning way

路数	己方路的价值	对方路的价值
1 路	1	1
2 路	5	10
3 路	10	15
4 路	25	35
5 路	25	25
6 路	10000	10000

表格 3-11 基于路的双参数评估函数防守倾向参数表

Table 3-11 The table of parameters of defending bias

路数	己方路的价值	对方路的价值
1 路	1	1
2 路	5	10
3 路	10	15
4 路	25	35
5 路	25	25
6 路	10000	10000

表格 3-12 基于路的双参数评估函数进攻倾向参数表

Table 3-12 The table of parameters of attacking bias

路数	己方路的价值	对方路的价值
1 路	1	1
2 路	15	10
3 路	30	15
4 路	50	35
5 路	50	25
6 路	10000	10000

另外由 2.5 节所描述，不同的搜索宽度与深度对程序影响很大。实验要验证在不同的搜索条件下，两种使用不同评估函数的胜率。

设置宽度为 20，深度分别为 3、4，各个深度进行 100 场比赛，观察两种方法胜率。当深度为 2 时，搜索深度太浅导致双方都经常不能找到致胜走法，最终导致双方经常和棋，对研究没有太大意义，所以不与讨论。胜负关系由图 3-13 所示。

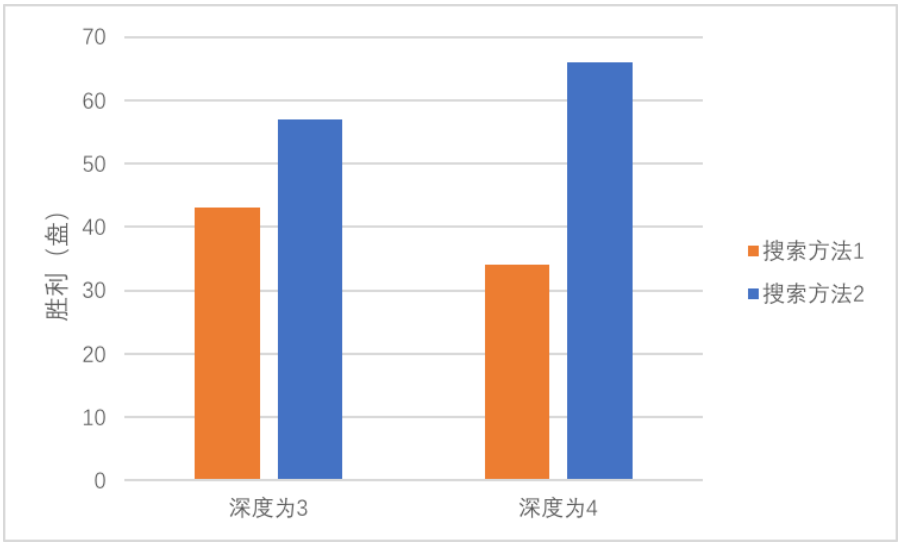


图 3-13 宽度为 20 两种方法胜率对比

Figure 3-13 The win rate of two method when the search width is 20

由图 3-13 可以看出，当宽度固定时，深度为 3 或 4 搜索方法 2 胜率更高，胜率在 60%上下浮动。

设置深度为 4，宽度分为设置为 10、20、30，每个宽度各进行 100 场比赛，观察两种方法胜率。胜负关系由图 3-14 所示。

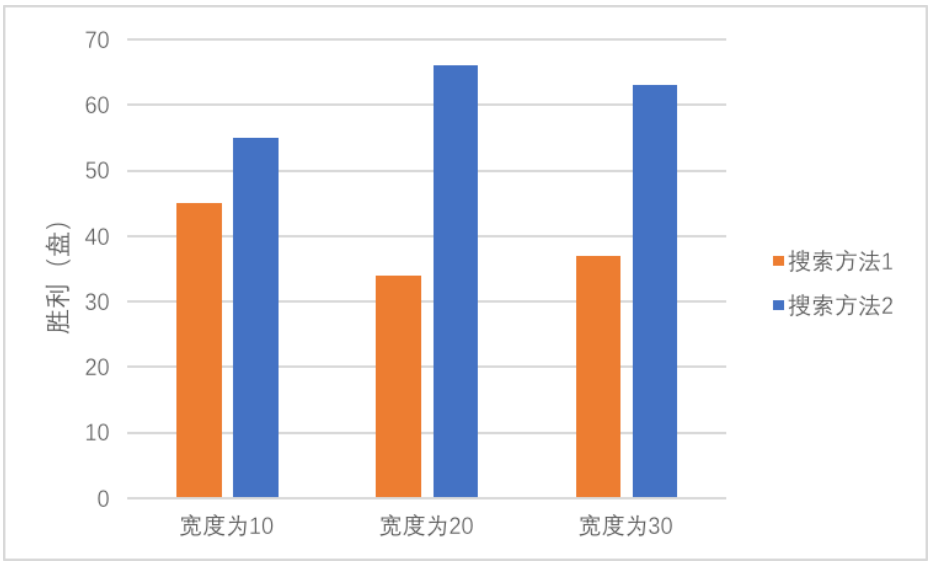


图 3-14 深度为 4 两种方法胜率对比

Figure 3-14 The win rate of two method when the search depth is 4

由图 3-14 可以看出，当深度为 4 时，宽度为 10、20、30 搜索方法 2 的胜率更高。

综上所述得出结论：

通过对相同深度不同宽度，与相同宽度不同深度对比，搜索方法 2 的胜率总要比搜索方法 1 的高，可以说双评价参数搜索方法比局部路的搜索方法棋力更高。

搜索深度对棋力的影响要大于宽度对棋力的影响。

总的来看当搜索深度或宽度较低时，双方更容易下出和棋。即双方都不能找到一个很好的走法。

3.3.6 搜索效率实验

由于要对生成的每一个节点都要进行评估，评估函数的效率对搜索效率有非常大的影响，需要用实验观察评估函数的搜索效率。搜索效率实验主要对比使用不同评估函数的搜索效率问题。实验的评估函数设置，以及参数设置同3.3.5。同样也要验证不同的搜索环境下的搜索效果。

搜设置宽度为20，深度分别为2、3、4，各对弈100局左右；比较相同宽度下每步平均落子时间。实验结果如图3-7所示。

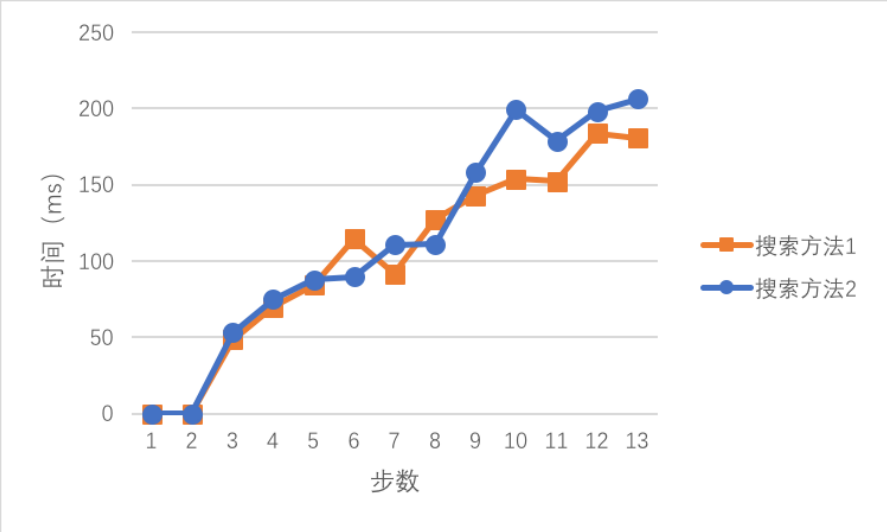


图 a 深度为 2

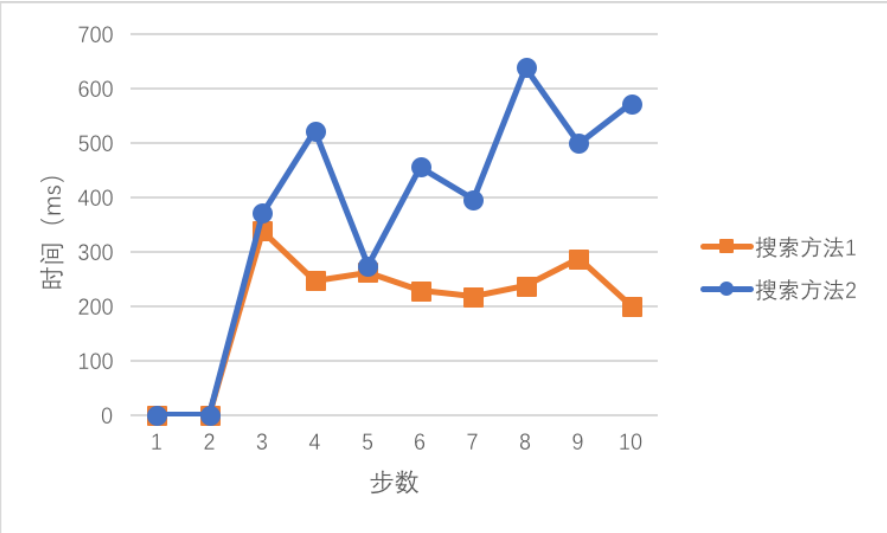


图 b 深度为 3

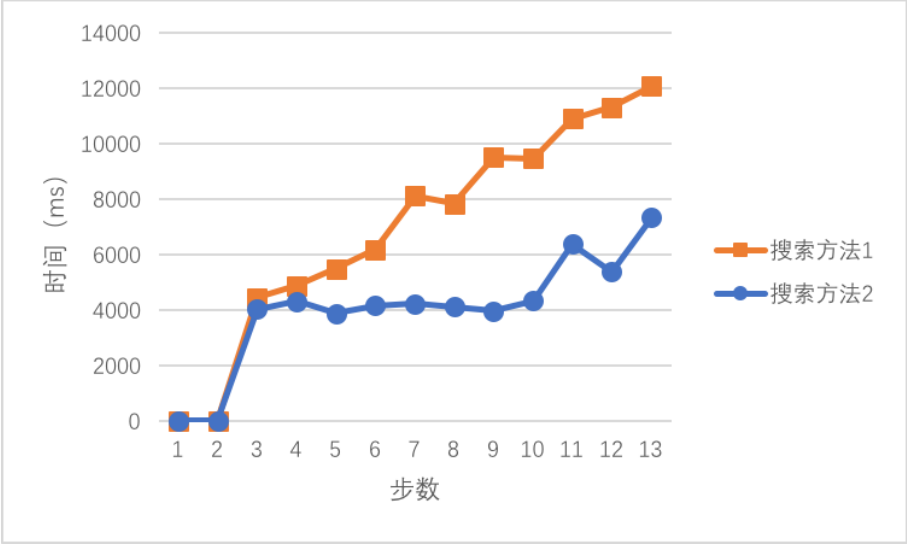


图 c 深度为 4

图 3-15 不同搜索深度效率对比

Figure 3-15 The search efficiency with different search depth

由图 3-15 可知，当搜索深度为 2、3 时，双评价参数搜索方法比局部路的搜索方法每步耗时更多；当深度为 4 时，双评价参数搜索方法比局部路的搜索方法每步耗时更少。分析原因为：1.当搜索深度较低时，每步耗时很少（实验中都小于 1s），误差较大。2.实验样本数量太少。每局棋局前两步耗时为 0 是由于使用了开局库，并没有进行搜索。就现有实验数据来看，双评价参数搜索方法在搜索深度较高时有更高的搜索效率。

设置深度为 4 时，宽度分别为 10、20、30，各对弈 100 局左右，比较相同深度下每步平均落子时间，实验结果如图 3-16 所示。

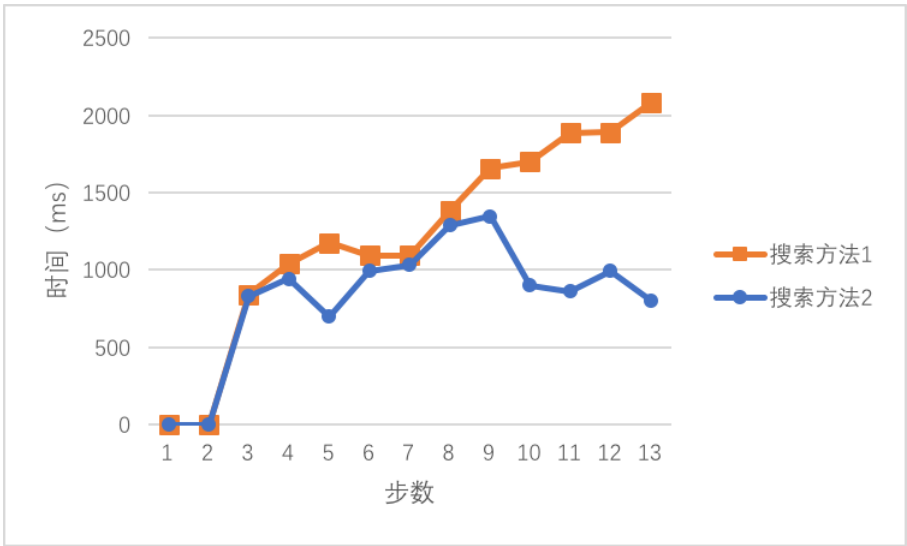


图 a 宽度为 10

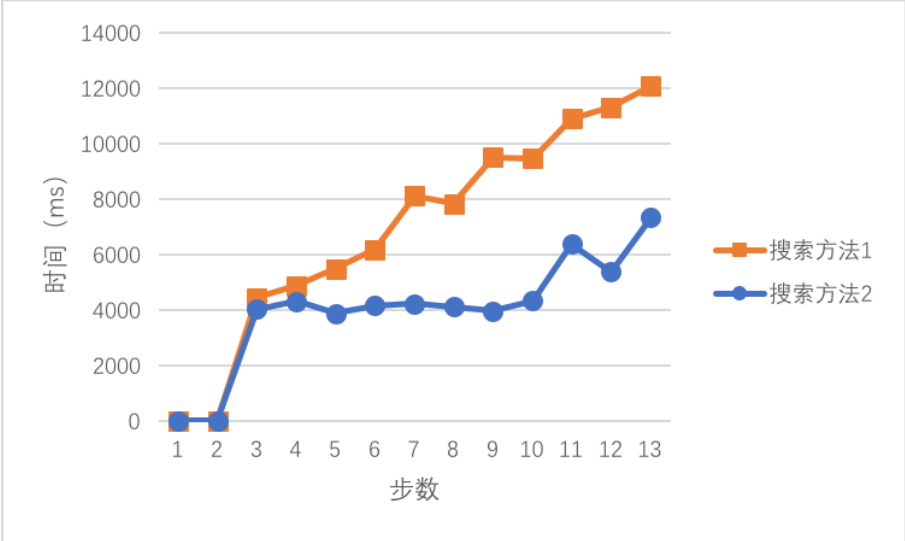


图 b 宽度为 20

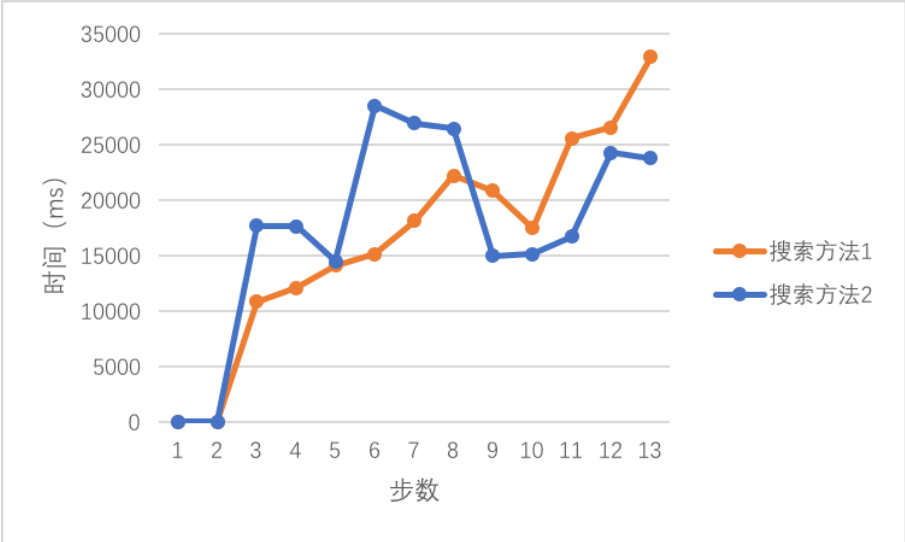


图 c 宽度为 30

图 3-16 不同搜索宽度效率对比

Figure 3-16 The search efficiency with different search depth

由图 3-16 可知，相同搜索宽度时，双评价参数搜索方法比局部路的搜索方法每步耗时增长更加缓慢。综合深度实验得出结论，当搜索深度为 4 时，无论搜索宽度为多少，双评价参数搜索方法比局部路的搜索方法的搜索效率更高。

3.3.7 参数影响实验

除了对比双评价参数搜索方法与局部路的搜索方法之外，我还想了解参数对双评价参数搜索方法的影响，为此设计了对比试验。是修改参数后与修改参数前双评价参数搜索方法对弈 80 局，观察胜负关系。实验设置如表 3-13，修改后参数如表 3-14。

表格 3-13 实验设置

Table 3-13 Experimental setting

搜索方法	设置
搜索方法 1	基于路评估函数+双评价参数（修改参数前）
搜索方法 2	基于路评估函数+双评价参数（修改参数后）

表格 3-14 己方优势参数表（修改后）

Table 3-14 The table of parameters when is advantage situation after modified

路数	己方路的价值	对方路的价值
1 路	1	1
2 路	30	10
3 路	40	15
4 路	50	35
5 路	60	25
6 路	10000	10000

由表 3-14 与表 3-12 对比可知，实验只修改己方优势时己方路的价值，调高了己方的参数。如此修改的影响是，己方在优势时进攻意识更强。实验结果如图 3-11。

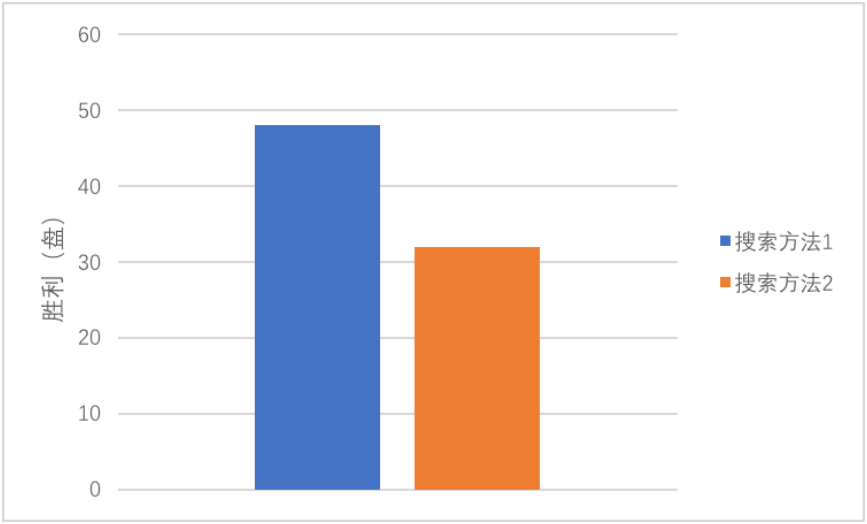


图 3-17 参数不同时实验结果

Figure 3-17 The win rate when the parameters are different

由图 3-17 可知修改参数后胜率下降了，参数对棋局影响比较复杂，现阶段不能得出合理结论。

3.4 本章小结

本章为本文研究的重点，即六子棋的评估函数，大致描述了六子棋评估函数的作用与现有的方法。六子棋目前评估函数大致分为两种，即基于棋型的评估函数和基于路的评估函数。重点描述了基于路的评估函数，展示出现有的方法与不

足。在此基础上提出了多参数对应多局面的思想，提出了双参数评估函数。为了验证双参数的评估函数的确定性，设计了棋局局面倾向实验、估值准确性实验和棋力水平实验。局面倾向性实验证明不同实验参数设置会导致不同倾向的说法。估值准确性实验证明在某些局面下，双评价参数评估函数对比传统基于路的评估函数能够更准确的判断当前局面，能更准确地引导棋局。棋力水平实验证明了使用双评价参数评估函数对比传统基于路的评估函数有着更高的胜率。但目前只是确定了优势和劣势两种局势，只有进攻倾向性和防守倾向性两组参数，可能在确定更多中局势，使用更多套参数情况下，得到更好的结果。此外由于评估函数对搜索效率有着很大的影响，设计了搜索效率实验对比两种评估函数的评估效率，实验结果表明双评价参数评估函数有着还不错的评估效率。最后设计了参数对比试验，参数对棋局影响比较复杂，其影响有待挖掘，现阶段没有很好的结论。

第4章　基于双参数评估函数六子棋程序的设计与实现

本章基于所提的双评价参数评估函数设计实现了六子棋程序 Executor，详细介绍了各个功能模块的功能与实现方法，进一步验证了本文方法的有效性。由于评估函数部分前文已有详细叙述，这里不再额外叙述。

4.1　六子棋体系结构

一个六子棋程序基本要包括交互界面、数据表示、走法生成、局面评估函数、走法选择和排序、搜索引擎、开局库。有些程序还带有一些特殊的功能，例如知识库、时间控制等。本章介绍的是我自己设计实现的六子棋程序 Executor。Executor 大致分为下述功能模块：人机交互界面模块、走法生成器、搜索引擎、评估函数模块、开局库模块。各模块之间调用关系如图 4-1 所示:

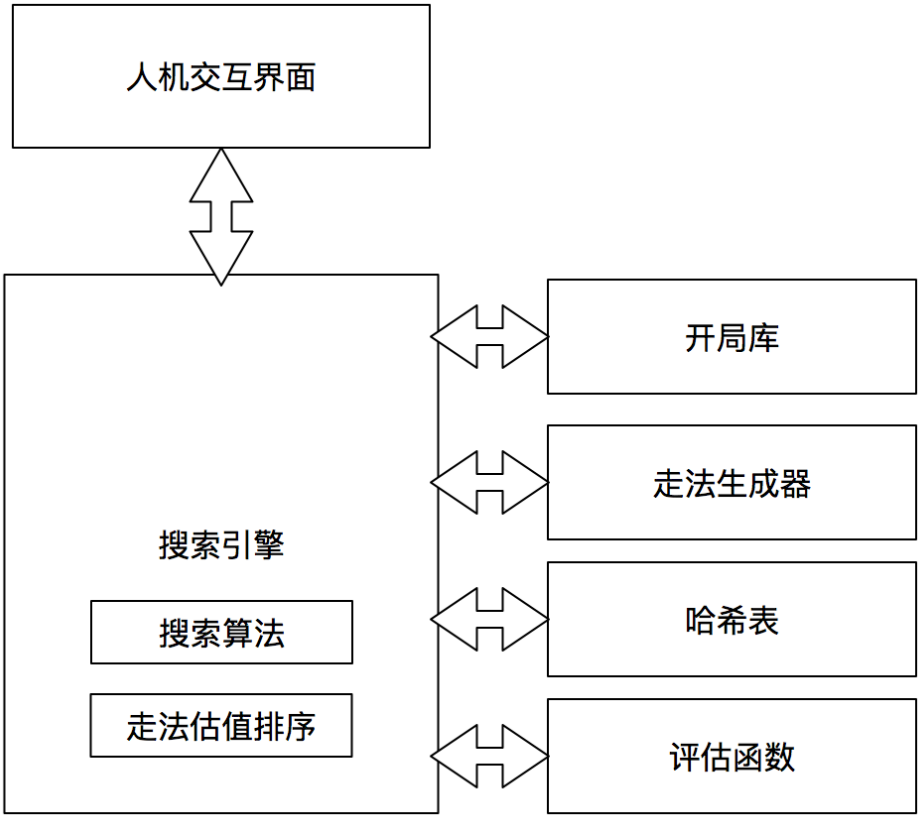


图 4-1 六子棋程序功能模块关系图
Figure 4-1 The relationship of functional model of connect6 process

4.2　人机交互界面

人机交互界面确定了人机交互的方式，六子棋人机交互界面通常有棋盘部分与其他功能部分。在棋盘部分中程序会接受鼠标的点击事件，发生点击事件位置

的横纵坐标（x, y）可以获取到，并以此坐标为圆心画圆（具体落子位置有一个映射关系），在棋盘相应位置就会形成落子。Executor 界面与落子如图 4-2 所示:

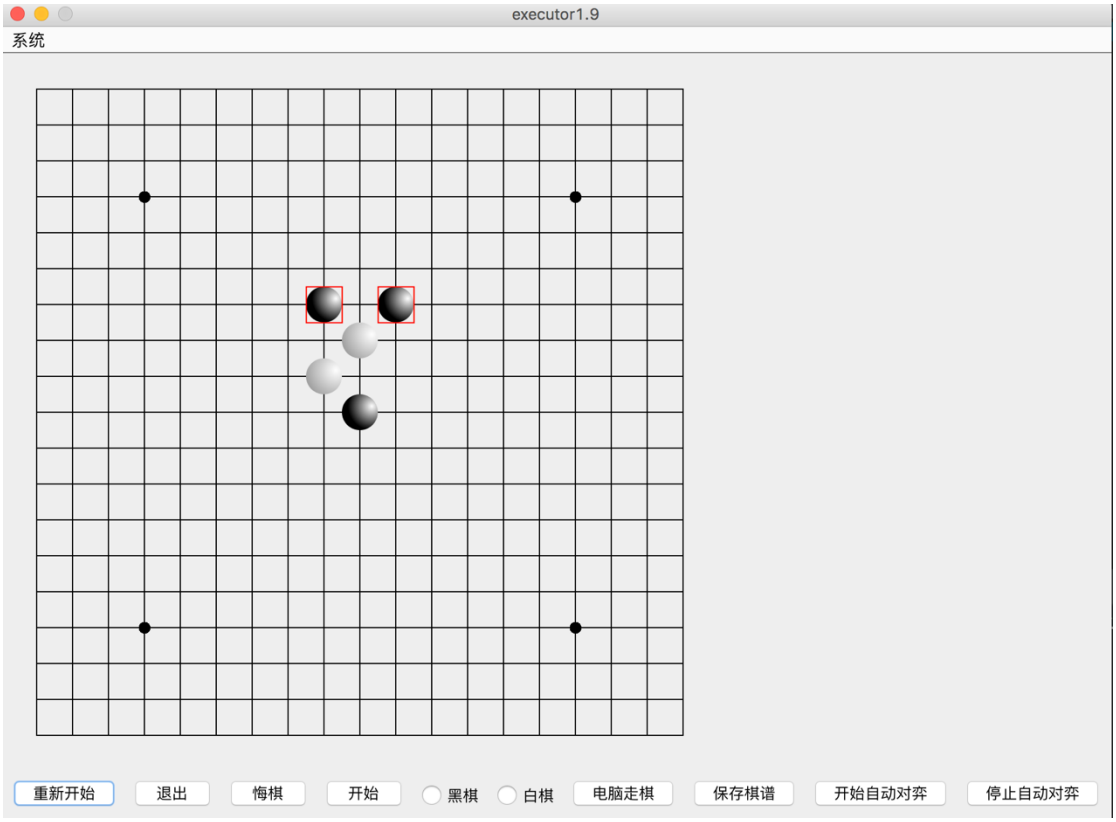


图 4-2 Executor 界面

Figure 4-2 The user interface of Executor

落子的颜色会根据走棋的数量做出相应的判断，比如第 1 步为黑棋，2、3 步为白棋，4、5 步为黑棋，以后每走两步交换落子权利。每次落子都会判断当前是否有 6 个相同颜色的棋连成一条直线，如果有会终止棋局，并宣布胜负关系，如图 4-3 所示。

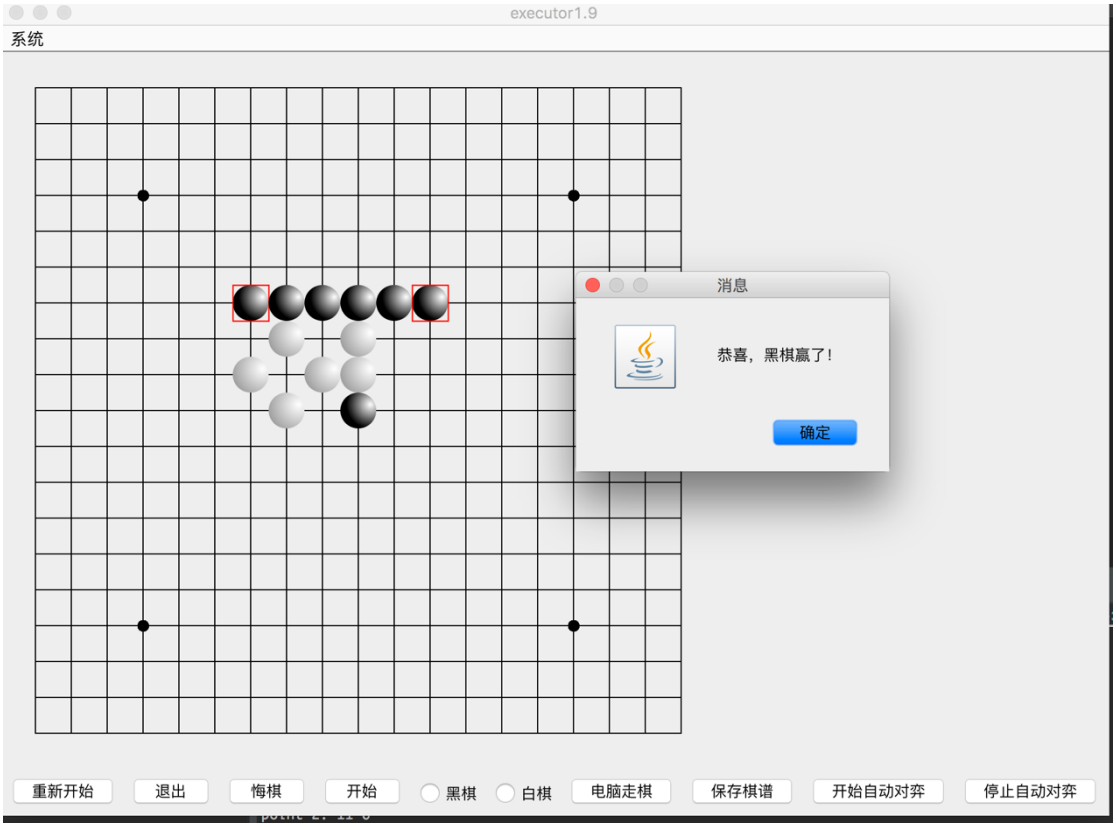


图 4-3 棋局结束

Figure 4-3 The end of a game

Executor 界面中包含了重新开始、退出、悔棋、保存棋谱与自动对弈等功能。每次落子时，落子的相关信息会保存到数组中，相关信息包括棋子位置、棋子颜色、棋子步数等。保存棋谱时，会把相应数据按自定义格式写入文件。悔棋时只需在数组尾部去掉落子信息。重新开始功能会把响应状态进行初始化。当开始自动对弈功能时，无需手动控制，黑白双方会自动对弈。自动对弈双方使用的搜索引擎与其他功能组件可以自由搭配。此功能可以方便的更换对弈双方的版本，达到对不同版本棋力做测试的效果，方便进行对比试验。上述为 Executor 实现的功能，其他六子棋程序一些会有棋谱展示、计时等功能。

4.3 走法生成器模块

走发生成器的主要作用是从当前局面中选择出几个候选走法。六子棋使用的棋盘为 19*19 的规格，总共 361 个交叉点。每步棋所有合理走法大约有 361*360/2 种，要把这么多种走法生成和列出是没有必要的。人类再走每一步棋时，会根据经验先挑出几个概率较高的候选走法，在通过几步推演后选出最佳的走法。走法生成器也有类似的过程。Executor 的走法生成器，候选走法生成的范围是每个棋子距离两个的范围。如图 4-4，所有黑点所表示的位置都是有效的候选位置。

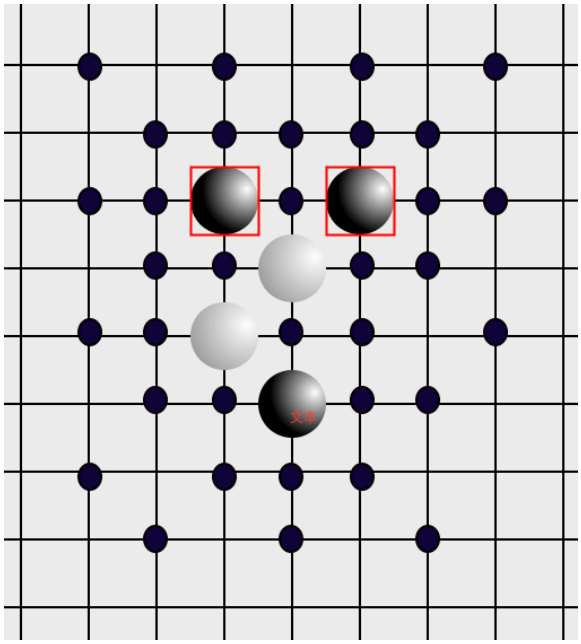


图 4-4 走法生成器有效位置范围

Figure 4-4 The valid position chooses by move generator

4.4 搜索引擎模块

搜索引擎也是六子棋程序的主要功能之一。经过走法生成器对相应走法进行展开，会形成一颗博弈树（如图 4-5），博弈树的每一个节点都对应棋局中相应的局面。对每一个节点使用评估函数，会得到每一个节点的估值。图是一颗估值之后的博弈树。搜索引擎的作用就是通过搜索找到最优的那一个节点。

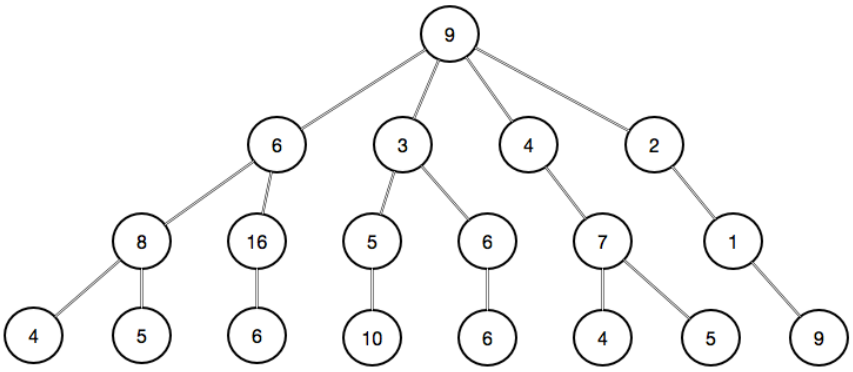


图 4-5 估值后的博弈树

Figure 4-5 A game tree after evaluated

Executor 采用的搜索方法是 TSS 迫着搜索+ α - β 剪枝搜索。α - β 剪枝前面已叙述过不再赘述，下面详细描述下 TSS 迫着搜索。

TSS（Threat Space Search）又叫威胁空间搜索，主要思路为不断的进攻迫使对方进行防守，在对方守不住的情况下获取胜利。TSS 在五子棋中被广泛应用，如在五子棋中不断冲四，最终获取胜利。

六子棋的规则为先连六子方获胜，当一方连成 4 子或 5 子时，对方由于要阻止其获胜，必须使用一子或两子进行防守，这时候就称对对方形成了迫着。对方需要用一子进行防守时称为单迫着，对方需要用两子进行防守称为双迫着。图 4-6 为一种单迫着的局面，此局面下黑方需要选择 1 位置或 2 位置进行防守；图 4-7 为一种双迫着的局面，此局面下黑方需要选择 1 和 3 或 2 和 4 或 2 和 3 位置进行防守。

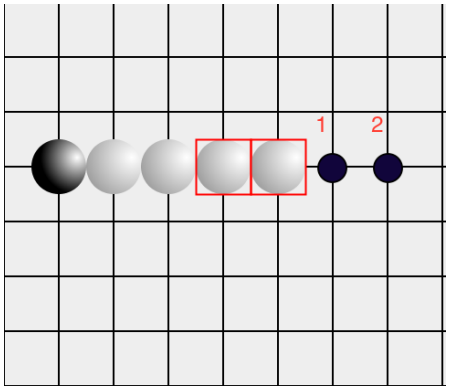


图 4-6 单迫着

Figure 4-6 Single threat

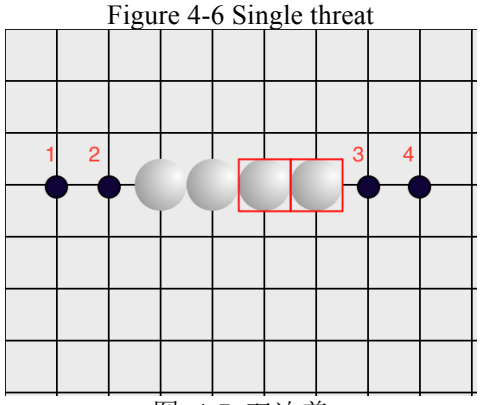


图 4-7 双迫着

Figure 4-7 Double threat

在使用 TSS 进行搜索时只对可以产生迫着的走法进行搜索，每层搜索的节点数量大大减少，搜索深度加深，可以有效滴提早搜索到胜利走法。

4.5 hash 函数模块

在搜索过程中，有可能在不同的节点搜索到相同的局面。如图 4-8 与图 4-9，两张图中棋子的落子顺序不同，在形成博弈树中是两个节点，但生成的局面是一样的。如果对两个局面都进行估值，或造成搜索时间上的浪费。解决的方法是当对一个节点进行过估值之后，将节点数据存储到 hash 表中，当搜索到其他局面

时先去 hash 表中进行比对。如果 hash 表中以存在当前局面则跳过对其的估值与搜索，否则在进行估值。

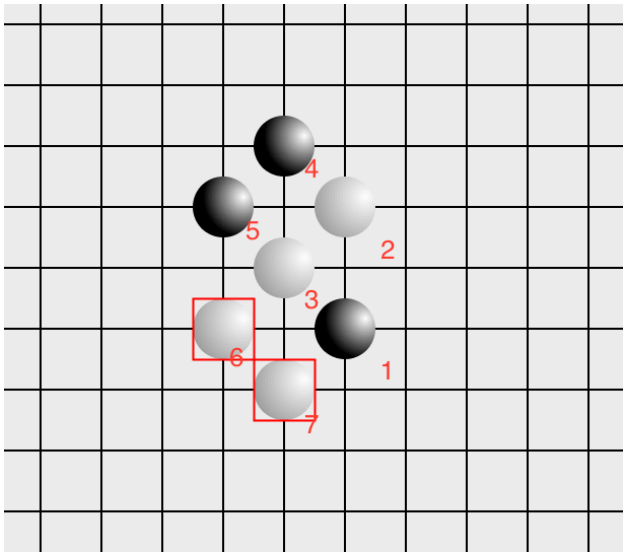


图 a

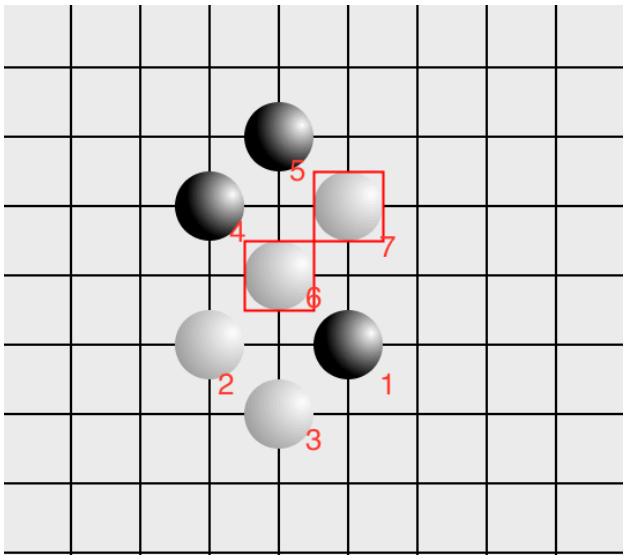


图 b

图 4-8 两种相同的局面

Figure 4-8 Two same situation

Executor 使用容量为 220 的数组作为 hash 表，表中的每一项为称作 HashItem，具体表示方式为 HashItem[1024*1024]。Executor 在 hash 表设计中，对棋盘 361 个位置用一对数值表示，这一对数值为 HashKey32 和 HashKey64。HashKey32 代表一个 32 位数值，HashKey64 代表一个 64 位数值。黑棋、白棋用不同在同一点用不同的数值表示，具体表示为 HashKey32[2][19][19]、HashKey64[2][19][19]。数组的第一维区分黑棋与白棋，后两维区分 19*19 个落子位置。Hash 表初始化时会分别对 HashKey32 与 HashKey64 初始化，具体代码为如下：

```
public static void initializeHashKey() {
    for (int k = 0; k < 2; k++) {
```

```
        for (int i = 0; i < ROWS; i++) {
            for (int j = 0; j < COLS; j++) {
                m_nHashKey32[k][i][j] = (int)(Math.random() * Integer.MAX_VALUE);
                m_ulHashKey64[k][i][j] = (long)(Math.random() * Long.MAX_VALUE);
            }
        }
    }
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 1024 * 1024; j++) {
            m_pTT[i][j] = new HashItem();
        }
    }
}
```

当前局面如果不存在 hash 表中，会插入到 hash 中。插入的步骤为：1.根据当前局面黑白子的分布情况，算出当前局面下的 HashKey32 与 HashKey64，计算伪代码为：

foreach：

m_HashKey32 = m_HashKey32 ^ m_nHashKey32[type][i][j]

m_HashKey64 = m_HashKey64 ^ m_ulHashKey64[type][i][j]

2.分别算出 HashKey32 与 HashKey64 后 HashKey32 代表当前局面插入 hash 表的位置，由于 hash 表的大小只有 220，所以 HashKey32 只取前 20 位作为 hash 表的索引，计算公式为 index = HashKey32 & 0xFFFFF。3.由于可能会存在 index 冲突的情况，需要用 HashKey64 做校验。HashKey64 会被当做校验数存入 HashItem 中，除 HashKey64 外，HashItem 中保存的信息应该还包括当前局面通过评估函数算出的估值 value 与节点在博弈树搜索中的深度 depth。

Hash 表的查询步骤为：1.与插入过程相同，算出当前局面下的 HashKey32 与 HashKey64。2.与插入过程相同，通过 HashKey32 算出 index。3.判断数组中 index 位置是否存在 Item，如果不存在则 hash 表中不存在当前局面信息。4.如果存在 Item 需要比对表中 HashKey64、depth 与当前局面 HashKey64、depth 是否一致，如果一致则查询存在，不一致则查询不存在。

为验证 hash 函数效果，本文设计了一组搜索效率的对比试验。实验配置如表 4-1。

表格 4-1 实验设置

Table 4-1 Experimental setting

搜索方法	设置
搜索方法 1	基于路评估函数+双评价参数（带有 hash 函数）
搜索方法 2	基于路评估函数+双评价参数（不带有 hash 函数）

实验搜索深度为 4，搜索宽度分别为 10、20、30，双方分别对弈 20 盘。实验结果如图 4-11。由于深度为 4 为较常用的搜索深度，且实验能够说明 hash 函数对效率的影响，所以不在设置不同搜索深度的对比试验。

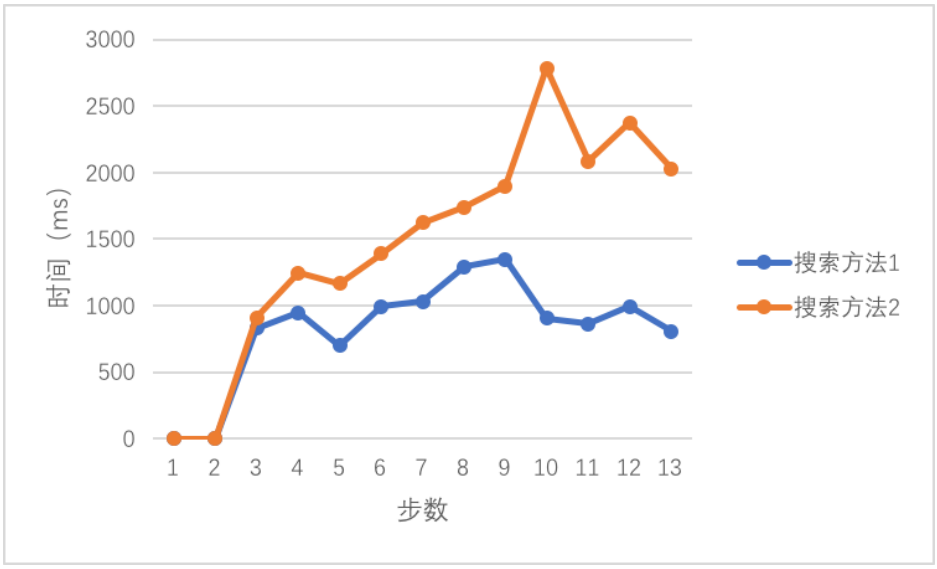


图 a 宽度为 10

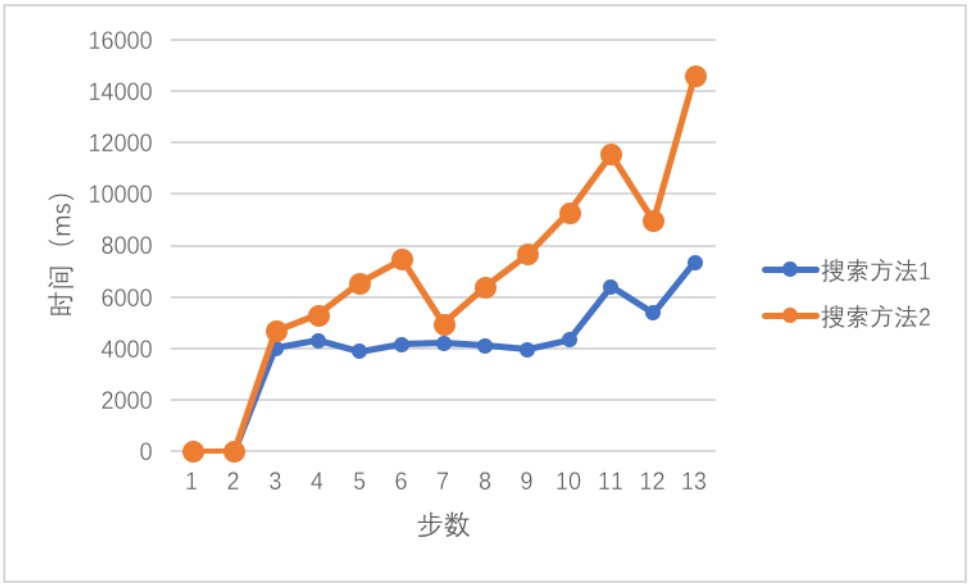
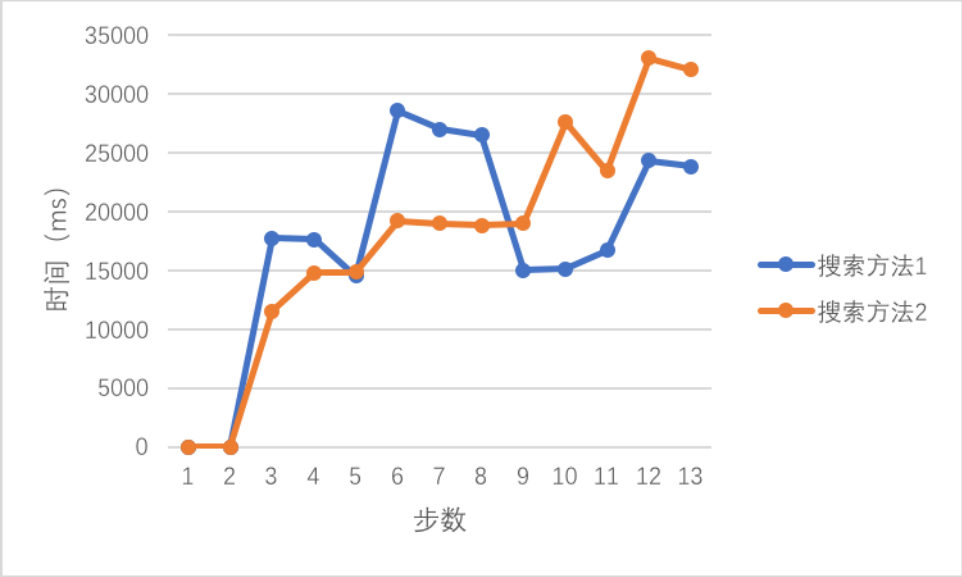


图 b 宽度为 20



(c) 宽度为 30

图 4-9 hash 函数搜索效率实验结果

Figure 4-9 The experimental result of using hash function

由实验结果可知，未使用 hash 函数时，每一步时间消耗趋势增加幅度更大，使用 hash 函数搜索效率明显有提高。

4.6 开局库模块

开局库以文本的形式存储在磁盘中，在程序启动时被加载到程序中。由于六子棋棋谱没有统一的格式，这里使用自定义的棋谱格式。程序启动时，OpeningLoader 会到指定目录下解析棋谱文件，棋谱解析后会保存到 OpeningTable 中，OpeningTable 的设计与 hash 函数设计相同。当行棋时，程序会先判断局面是否脱谱。如果已脱谱，程序会调用搜索引擎相应方法进行搜索；如果未脱谱，程序会到 OpeningTable 中查找是否存在当前局面下的下一步走法，如果存在则取出相对应走法并返回，如果未存在则把程序标志为以脱谱，并调用搜索引擎进行搜索。

目前 executor 使用的开局库为 3 子开局库，既黑方走一子，白方走两子的开局。六子棋常见 3 子开局有 20 种，图 4-12 列出了其中四种。

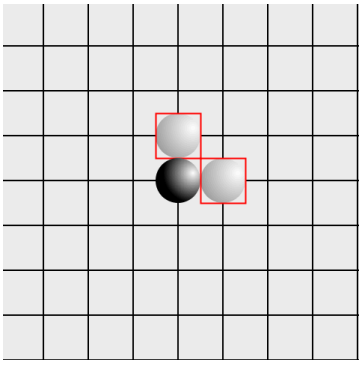


图 a

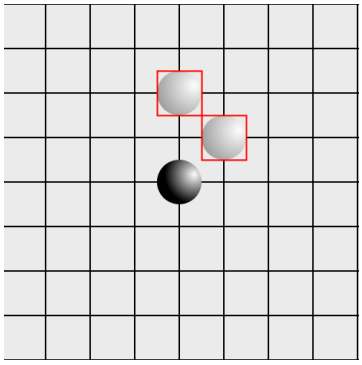


图 b

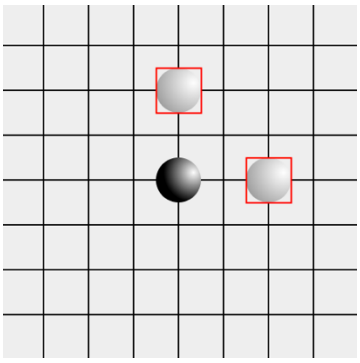


图 c

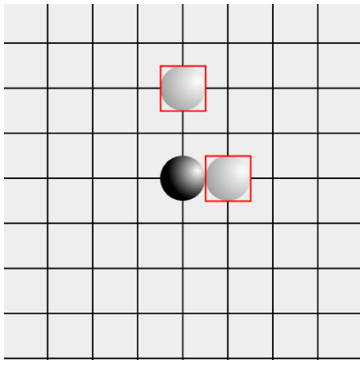


图 d

图 4-8 六子棋 4 种 3 子开局

Figure 4-10 Four kind of 3 stone opening of connect6

4.7 程序整体流程

最后描述 Executor 程序的整体流程。程序启动时程序会加载棋谱到开局库中，机器要落子时首先回到开局库中查询是否存在当前局面下的下一步走法，存在则返回此走法，不存在则调用搜索引擎进行搜索。搜索时，搜索引擎会先使用 TSS 迫着搜索方法进行搜索，迫着搜索会直接判断当前局面是否有迫着赢棋下法，如果存在返回该下法，如果不存在则进行 α - β 剪枝搜索。剪枝搜索开始时会先调用走法生成器，生成当前局面下的候选走法，再对所有候选走法是用评估函数进行评估，得到每种走法后局面的估值。根据估值对走法进行排序，选出估值最高的几种走法进行下一层的搜索，选择走法的数量与搜索的层数可以自行设定。完成搜索，找到估值最高的走法并返回。流程图如图 4-13：

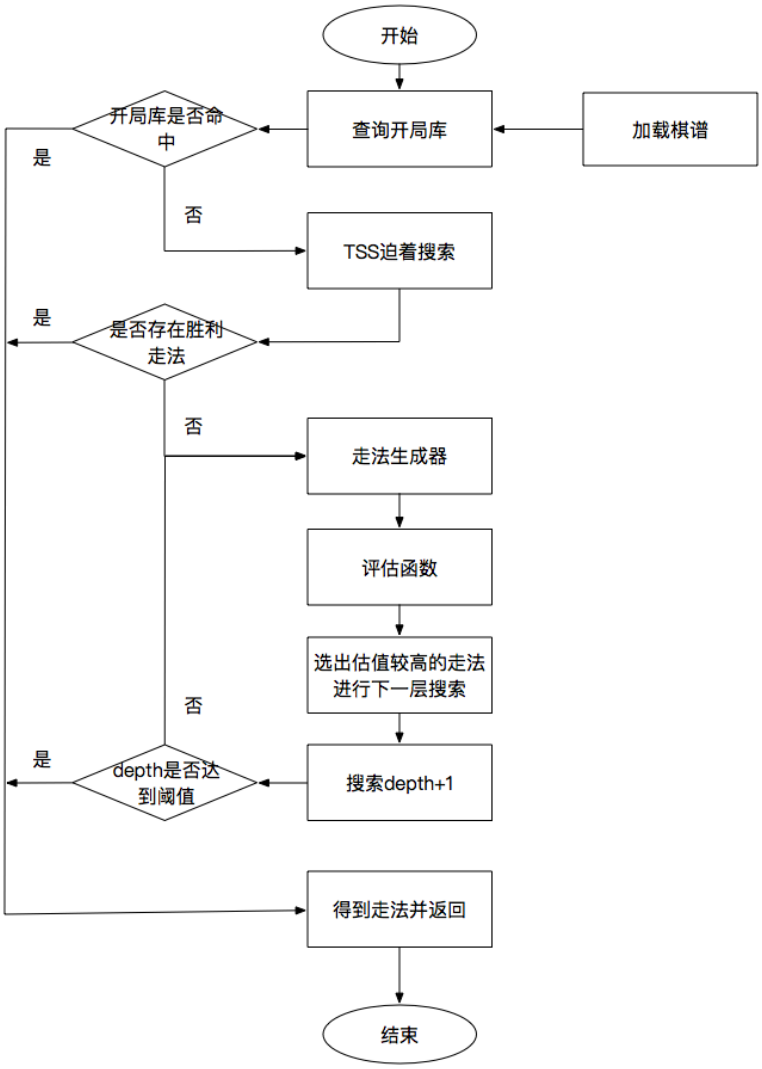


图 4-9 六子棋整体流程图

Figure 4-11 The process of connect6 system

4.8 本章小结

本章主要介绍了六子棋程序 Executor 的整体架构，展示了 Executor 程序各功能模块之间的关系，并分别详细叙述了各个功能模块的作用与实现方法。程序的评估函数模块使用了双评价参数评估函数。程序整体运行效果良好。

结 论

人工智能是计算机科学的一个分支，它试图探求智能的本质，并产生出一类新的能和人类智能类似的方式做出反应的智能机器。随着计算机技术的发展，人工智能慢慢成为一个瞩目的研究热点，其研究成果在经济政治决策、机器人、仿真系统和控制系统中得到了广泛的应用。机器博弈常被称为人工智能的“果蝇”，它被认为是人工智能领域最具挑战性的研究方向之一。六子棋是机器博弈的其中一个项目，它是一个新兴棋种，近几年有着迅速的发展并得到了广泛的关注。

六子棋的评估函数大致分为两类，基于棋型的评估函数与基于路的评估函数，各自有其优缺点。基于棋型的评估函数规定了六子棋中常见的几种棋型，六子棋的每种局面都是六子棋棋型的组合。这样通过把局面分割成棋型就可以评估出当前局面的态势。基于棋型的评估函数可以较为准确的评估局面，但棋型分隔起来比较复杂且耗时，没有一种很好的解决方案。基于路的评估函数把局面拆解成不同的路，拆解起来较为方便，但由于一组路只用一组参数经行评估，不利于应付不同的局面。

因此本文以六子棋为研究对象，研究的重点是六子棋的评估函数，研究的主要内容有：

1) 本文在基于路的基础上提出了一种新的双参数评估函数，提出了不同局面使用不同组参数的新想法，初步解决了应付不同局面的问题。同时把基于全局路的评估函数与基于局部路的评估函数相结合，保证了评估的效率。

2) 设计实现了 Executor 六子棋程序，程序包括开局库、搜索引擎、走法生成器、评估函数、hash 表等功能模块。详细阐述了模块的构架关系，以及阐述了模块各自的功能的实现细节。

为了验证双参数的评估函数的确定性，设计了棋局局面倾向实验、估值准确性实验和棋力水平实验。局面倾向性实验证明不同实验参数设置会导致不同倾向的说法。估值准确性实验证明在某些局面下，双评价参数评估函数对比传统基于路的评估函数能够更准确的判断当前局面，能更准确地引导棋局。棋力水平实验证明了使用双评价参数评估函数对比传统基于路的评估函数有着更高的胜率。但目前只是确定了优势和劣势两种局势，只有进攻倾向性和防守倾向性两组参数，可能在确定更多中局势，使用更多套参数情况下，得到更好的结果。此外由于评估函数对搜索效率有着很大的影响，设计了搜索效率实验对比两种评估函数的评估效率，实验结果表明双评价参数评估函数有着还不错的评估效率。最后设计了

参数对比试验，参数对棋局影响比较复杂，其影响有待挖掘，现阶段没有很好的结论。

介绍了六子棋程序 Executor 的整体架构，展示了 Executor 程序各功能模块之间的关系，并分别详细叙述了各个功能模块的作用与实现方法。程序整体运行效果良好。

参 考 文 献

[1] I-C. Wu, D.-Y. Huang, and H.-C. Chang, “Connect6,” ICGA Journal, Vol. 28, No. 4, 2005, pp. 234-241.

[2] Herik V D, Jaap H, Uiterwijk, et al. Games solved: now and in the future[J]. Artificial Intelligence, 2002, 134(1):277-311.

[3] Uiterwijk J W H M, Van d H H J. The advantage of the initiative[J]. Information Sciences, 2000, 122(1):43-58.

[4] Lin P H, Wu I C. NCTU6 wins the man-machine connect6 championship 2009[J]. 2009.

[5] Csmiraz L. On a combinatorial game with an application to Go-moku[J]. Discrete Mathematics, 1980, 29(1):19-23.

[6] Wu I C, Huang D Y. A New Family of k -in-a-Row Games[M]// Advances in Computer Games. 2005:180-194.

[7] Pluhar A. The accelerated k-in-a-row game. [J]. Theoretical Computer Science, 2002, 270(1):865-875.

[8] Pluhar A. Generalizations of the game k-in-a-row[J]. Chicago Sun-Times, 2010.

[9] Alus L V, Herik H J V D, Huntjens M P H. GO- MOKU SOLVED BY NEW SEARCH TECHNIQUES[J]. Computational Intelligence, 1996, 12(1):7-23.

[10] Ming Y H, Tsai S C. On the fairness and complexity of generalized k -in-a-row games[J]. Theoretical Computer Science, 2007, 385(1):88-100.

[11] Alus L V, Herik H J V D, Huntjens M P H. GO- MOKU SOLVED BY NEW SEARCH TECHNIQUES[J]. Computational Intelligence, 1996, 12(1):7-23.

[12] Wágner J, Virág I. Solving Renju[J]. Icga Journal, 2001, 24(1):30--34.

[13] 李果. 六子棋计算机博弈及其系统的研究与实现[D]. 重庆大学, 2007.

[14] 徐心和, 王骄. 中国象棋计算机博弈关键技术分析[J]. 小型微型计算机系统, 2006, 27(6):961-969.

[15] Yen S J, Yang J K. Two-Stage Monte Carlo Tree Search for Connect6[J]. IEEE Transactions on Computational Intelligence & Ai in Games, 2011, 3(2):100-118.

[16] Tao J. Application and design of computer games system in connect6[C]// Control and Decision Conference. IEEE, 2014:3367-3370.

[17] LI Xuejun, WANG Xiaolong, WU Lei, et al. Game tree generation algorithm based on local-road scanning method for connect6[J]. CAAI Transaction on Intelligent Systems, 2015(2):267-272.

[18] Xue Y, Li H, Jiang T. Alpha-Beta-TSS in Connect6[C]// Control and Decision Conference. IEEE, 2015:3737-3742.

[19] Winands M H M, Uiterwijk J W H M, Jaap V D H H. An effective two-level proof-number search algorithm[J]. Theoretical Computer Science, 2004, 313(3):511-525.

[20] Yang J K, Tseng P J. Building Connect6 opening by using the Monte Carlo tree search[C]//

Eighth International Conference on Advanced Computational Intelligence. IEEE, 2016:331-336.

[21] Wu I C, Kang H H, Lin H H, Dependency-Based Search for Connect6[J]. Lecture Notes in Computer Science, 2014, 8427:1-13.

[22] Allis L V, Herik H J, Huntjens M P H. Go-Moku and Threat-Space Search[J]. Interview Questions, 1993.

[23] Alus L V, Herik H J V D, Huntjens M P H. GO- MOKU SOLVED BY NEW SEARCH TECHNIQUES[J]. Computational Intelligence, 1996, 12(1):7-23.

[24] Wu I, Lin H H, Lin P H, et al. Job-Level Proof-Number Search for Connect6[J]. Lecture Notes in Computer Science, 2010:11-22.

[25] Wu G, Tao J. Design and application of machine learning algorithm computer in connect6 of computer games system[C]// Control and Decision Conference. IEEE, 2016:4279-4282.

[26] Liu C, Wu B, Wu S. The design and optimization of Connect6 computer game system[C]// Control and Decision Conference. IEEE, 2014:3936-3940.

[27] 王骄, 王涛, 罗艳红,等. 中国象棋计算机博弈系统评估函数的自适应遗传算法实现[J]. 东北大学学报(自然科学版), 2005, 26(10):949-952.

[28] Takeuchi S, Kaneko T, Yamaguchi K. Evaluation of Game Tree Search Methods by Game Records[J]. IEEE Transactions on Computational Intelligence & Ai in Games, 2011, 2(4):288-302.

[29] 何华灿. 人工智能基础理论研究的重大进展——评钟义信的专著《高等人工智能原理》[J]. 智能系统学报, 2015(1):163-166.

[30] Hashimoto J, Kishimoto A, Yoshizoe K, et al. Advances in Computer Games[J]. Ifip — the International Federation for Information Processing, 2010, 4250(2):xiv,383.

[31] Research and Implementation of Intelligent Gobang-playing[C]// International Conference on Intelligent Information Technology Application. 2010.

[32] Ke F. An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing[J]. Computers & Education, 2014, 73(1):26-39.

[33] 徐心和, 邓志立, 王骄,等. 机器博弈研究面临的各种挑战[J]. 智能系统学报, 2008, 3(4):288-293.

[34] 张明亮, 吴俊, 李凡长. 五子棋机器博弈系统评估函数的设计[J]. 计算机应用, 2012, 32(7):1969-1972.

[35] 许慧颖. 基于决策优化策略的认知引擎关键技术的研究[D]. 北京邮电大学, 2015.

[36] Qiao Z H, Yang M, Wang Z J. Technologies Analysis of Connect6 Computer Game[J]. Advanced Materials Research, 2011, 171-172:679-682.

[37] 张加佳. 非完备信息机器博弈中风险及对手模型的研究[D]. 哈尔滨工业大学, 2014.

[38] Qiao Z H, Yang M, Wang Z J. Technologies Analysis of Connect6 Computer Game[J]. Advanced Materials Research, 2011, 171-172:679-682.

[39] 徐长明. 基于连珠模式的六子棋机器博弈关键技术研究[D]. 东北大学, 2010.

[40] 徐长明, 马宗民, 徐心和. 一种新的连珠棋局面表示法及其在六子棋中的应用[J]. 东北大学学报(自然科学版), 2009, 30(4):514-517.

[41] Zhang R, Liu C, Wang C. Research on connect 6 programming based on MTD(F) and Deeper-Always Transposition Table[C]// IEEE, International Conference on Cloud Computing and Intelligent Systems. IEEE, 2013:206-208.

[42] Wu R, Zhou K. Optimization of the Connect6 Evaluation Function Based on Threat Theory and Game Strategy[C]// Second Wri Global Congress on Intelligent Systems. IEEE Computer Society, 2010:94-97.

[43] 李果. 基于遗传算法的六子棋博弈评估函数参数优化[J]. 西南大学学报:自然科学版, 2007, 29(11):138-142.

[44] Fossel J D. Monte-Carlo Tree Search Applied to the Game of Havannah[J]. 2010.

[45] Yang J K. Bitboard Connection Code Design for Connect6[C]// Conference on Technologies and Applications of Artificial Intelligence. IEEE Computer Society, 2013:386-391.

[46] Wu I C, Lin P H. Relevance-Zone-Oriented Proof Search for Connect6[J]. IEEE Transactions on Computational Intelligence & Ai in Games, 2010, 2(3):191-207.

[47] Silver D. Reinforcement learning and simulation-based search in computer go[M]. University of Alberta, 2009.

[48] Wu I C, Yen S J, Wu C, et al. NCTU6 wins Connect6 tournament[J]. Icga Journal, 2006, 29(3):157-158.

[49] Wu I C, Lin Y S, Tsai H T, et al. The Man-Machine Connect6 Championship 2011[J]. Icga Journal, 2011, 34(2):103-105.

[50] Xu C M, Ma Z M, Tao J J, et al. Enhancements of Proof Number Search in Connect6[C]// 中国控制与决策会议. 2009:4561-4565.

[51] Qiao Z H, Yang M, Wang Z J. Technologies Analysis of Connect6 Computer Game[J]. Advanced Materials Research, 2011, 171-172:679-682.

[52] 陈光年. 基于智能算法的六子棋博弈行为选择的应用研究[D]. 重庆理工大学, 2010.

攻读硕士学位期间取得的研究成果

论文：

[1] Yifei Qi, Jin Wang, Qing Zhu, Wei Xu, Xi Wu. Double Parameter Evaluation Function Based on Path(Chinese Control and Decision Conference CCDC 2018)（论文第三章）

项目：

[1] 2016 年 3 月：web 项目--大医精诚

[2] 2016 年 9 月：web 项目--智慧农业二期

奖励：

[1] 2016 年 8 月　荣获“全国大学生机器博弈比赛二等奖”

[2] 2016 年 10 月　荣获“学习优秀一等奖”

[3] 2016 年 11 月　荣获“国家奖学金”

[4] 2017 年 10 月　荣获“优秀研究生”

致谢

三年转眼之间就要过去了，紧张而忙碌的学生时代即将结束。三年中，我收获了许多，感慨良深。

首先要感谢我的导师朱青教授，在日常学习工作中，朱老师一直以高标准严格要求我，专业支持和耐心指导给予我很大的帮助，让我丰富了自己的专业知识，提升了自己的动手实践能力。在毕业论文和学术研究过程中，每当我遇到问题时老师总能为我指明研究思路，使我顺利地完成课题研究。朱老师精益求精的研究精神和一丝不苟的工作态度为我们树立了良好的榜样，是我今后学习工作的宝贵财富。

感谢王瑾老师在科研工作和论文撰写中给予我的帮助，每当我遇到困难，都会耐心解答，与我共度难关。在项目的开发过程中，王老师的辛勤付出帮助我提升了自己的专业水平，为求职打下了良好的基础。在平时生活中，王老师平易近人，不仅是我的学习导师，更是我生活中的好朋友。在此，向我敬爱的老师们致敬。

感谢徐洪涛、刘轩、姚范三位前辈在日常工作和生活中对我的关心和帮助，在他们的帮助下，顺利地完成了项目的开发工作，提升了自己的动手能力。

感谢实验室中一起学习、科研的同学们。三年中我们有欢笑有苦恼，大家一起奋斗相互鼓励，终于迎来了即将毕业的日子。

最后感谢我的家人，你们始终默默支持着我，关心着我，是我开展科研工作强大的动力，是我永远的支柱。