学位类别 理 学 学 号 160701J102

ངབ་འབྱམས་པའི་བསྟབ་གནས་དཔྱད་རྩོམ།

硕士学位论文

MCTSགཞིར་བྱས་པའི་བོད་ཀྱི་བཙོར་འགྲེག་གི་རང་འགུལ་འགྲེག་ཨའིད་མ་ལག་ཞིབ་འཇུག་དང་མངོན་འགྱུར།

基于 MCTS 的藏式夹棋人机博弈系统的研究与实现

学位申请人姓名：尕藏扎西

导师姓名及职称：安见才让 教授

专 业 名 称：藏语信息处理工程

专 业 方 向：藏文信息处理及应用

2019 年 5 月 27 日

学位类别：理学

分 类 号：

学　　号：160701J102

密　　级：

མཚོ་སྨོན་མི་རིགས་སློབ་གྲྭ་ཆེན་མཐེ་ངན་འབྲམས་པའི་བསྟབ་གནས་དབྱད་རྩོམ།

青海民族大学硕士学位论文

MCTSགཞིར་བྱས་པའི་བོད་ཀྱི་བཙོར་འཕྲིག་གི་རང་འགུལ་འཕྲིག་འདེད་མ་འག་ཞིབ་འཇུག་དང་མངོན་འགྱུར།

基于 MCTS 的藏式夹棋人机博弈系统的研究与实现

学位申请人姓名：尕藏扎西

导师姓名及职称：安见才让 教授

专　业　名　称：藏语信息处理工程

研　究　方　向：藏文信息处理及应用

答辩委员会（签名）：

基于 MCTS 的藏式夹棋人机博弈系统的研究与实现

摘 要

随着计算机软硬件以及互联网飞速发展，相关学科也不断发展。其中人工智能是近期以来研究的热点，而计算机博弈是人工智能研究的重要载体，也是目前最具有挑战性的研究方向之一。人工智能技术的快速发展极大地推动了科技进步和社会发展。在国内外计算机围棋、国际象棋、亚马孙棋等研究已成熟。特别在 2016 年至 2017 年间 Google 旗下 DeepMind 团队开发的 AlphaGo 和 AlphaGo Zero，再次将全世界的目光聚集到计算机博弈上。但计算机藏棋领域的研究才刚刚起步，从而还面临着很大的挑战。虽然藏棋的起源有很多不同的观点，但在大量记载藏棋的文献和出土文物可以证明藏棋的历史悠久，并且藏棋具有自己独特的行棋表现形式和文化内涵。在这里讨论的藏式夹棋是藏族棋类文化中具有代表性的一项棋类，也是目前在藏区流传广泛、内容丰富多样的一项棋类。从计算机博弈角度看，藏式夹棋搜索空间大、下棋规则多、盘面变化快。因此研究计算机藏式夹棋博弈具有很大的技术性挑战和意义。

由蒙特卡洛方法和 UCB 策略结合的 MCTS 算法是目前高水平计算机棋类程序普遍采用的较好的算法，但此算法也有不足之处。本文研究了藏式夹棋独特的行棋规则、蒙特卡洛树搜索和 UCB 策略及其在藏式夹棋博弈中的应用，针对 UCT 算法存在的不足，提出了一种藏式夹棋静态评估和 UCT 算法结合的方法，并从实践上证明了算法优化的可行性和准确性。而后，将藏式夹棋人机博弈系统 TibetanGo 的总体框架进行了详细的分析，并系统分为博弈引擎模块和通信模块、博弈客户端等三大模块。

最后，测试了模块与模块之间的协调性。通过多次的对弈测试来证明藏式夹棋人机博弈系统 TibetanGo 的实用性。另外，应用原始 UCT 算法和优化后的 UCT 算法进行了对比测试，得出了优化后的藏式夹棋人机博弈系统 TibetanGo 的博弈胜率达到了 67.2%。

关键词： 藏式夹棋;蒙特卡洛树搜索;静态局面评估;UCT 算法

MCTSགཞིན་བྱས་པའི་བོད་ཀྱི་བཙོར་འཕྲིག་གི་རང་འགུལ་འཕྲིག་འདེད་མ་ཡིན་ཞིབ་འཇུག་དང་མངོན་འགྱུར།

ཁང་དོན་གནད་བསྡུས།

དེ་ཡང་རྩོལ་འགོར་གྱི་སྤ་མཉེན་ཚས་དང་སྦྱོལ་རེས་དུ་རུ་མཁོགས་སྡུར་དང་འཕེལ་རྒྱས་སོང་བ་བསྟན། འབྲེལ་ཡོད་ཀྱི་རིག་གཞུང་ཚན་ཞག་ཡང་མ་ཐུག་དུ་འཕེལ་རྒྱས་བྱུང་། དེའི་ཁྱོད་ནས་མིའི་བཟོས་རིག་རྣམ་ནི་ཉེ་ལམ་ཞིབ་འཇུག་གི་ཁ་ཐོག་གོ་ཅོ་ཞིག་ལ་འགྱུར་ཡོད་ངེ་བཞིན་རྩོལ་འགོར་གྱིས་འཕྲིག་འདེད་པ་ཡང་མིའི་བཟོས་རིག་རྣམ་ལ་ཞིབ་འཇུག་བྱེད་པའི་བྱེད་ཐབས་གཅོ་ཞིག་ཡིན་ལ། མིས་སྤང་ཁང་ནགའ་ཡོད་པའི་ཞིབ་འཇུག་གི་ཁ་ཐོག་ཞིག་ཀྱང་ཡིན། མིའི་བཟོས་རིག་རྣམ་ཀྱི་ལག་ལེན་ཁེ་ཕན་ཆེ་ཤོས་འཕྲིན་ལས་ཀྱི་ཞོ་ཏེ་ཞིག་ཡིན་པ་ལས་ཁ་ཅིའི་ཚན་རིག་ཀྱང་ཡིན་ཡང་། རྒྱལ་ཁབ་ཕྱི་ནང་ནས་རྩོལ་འགོར་ལ་ཞུགས་ཤུགས་ཆེ་ཤོས་ཀྱི་ཐོག་ནས་ཀྱང་ཁ་ཅིའི་ཚན་རིག་ཀྱང་ཡིན། རྒྱལ་ཁབ་ཕྱི་ནང་ནས་རྩོལ་འགོར་ལ་ཞུགས་ཤུགས་ཆེ་ཤོས་ཀྱི་ཐོག་ནས་ཀྱང་ཁ་ཅིའི་ཚན་རིག་ཀྱང་ཡིན། ཡ་མ་སོན་འཕྲིག་སོགས་འདེད་པའི་ལག་རྩལ་ལ་ཞིབ་འཇུག་གི་རྩེར་སོན་ཡོད། ལྷག་པར་དུ་ ༢༠༡༨ ལོ་ནས་ ༢༠༡༩ ལོའི་བར་དུ་ Google འགའི་ DeepMind རུ་ཚོགས་ཀྱིས་གསལ་སྟེལ་བྱས་བའི་ AlphaGo དང་ AlphaGo Zeroཡིས་སྣར་ཡང་འཇམ་གྱིང་གི་མིག་དབང་འཕྲོག་པ་དང་།རྩོས་འགོར་འཕྲིག་འདེད་ལག་རྩལ་ལ་དོན་སྣང་ཆེན་པོ་བྱས།འོན་ཀྱང། རྩོས་འགོར་གྱི་བོད་འཕྲིག་འདེད་པའི་ཞིབ་འཇུག་འགོ་བརྩམས་མ་ཐག་ཡིན་ལ། དུ་དུང་མིག་མདུན་དུ་ཐག་གཅོད་བྱ་དགོས་པའི་གནད་དོན་མང་པོ་སྣག་ཡོད། བོད་འཕྲིག་གི་འབྱུང་ཁོངས་ལ་ལྟ་ཚུལ་མི་འདྲ་མང་ཡོད་ཀྱང། མིག་སྤང་ཀྱེད་ཡོད་པའི་ཡིག་ཚང་དང་རང་ས་ནས་ཐོན་པའི་རིག་དོངས་མང་པོ་ལས་བོད་འཕྲིག་ལ་ཝོ་རྒྱས་ཡུན་རིང་སྤྱན་པ་ར་སྤྱོད་བྱ་ཐུབ། དེ་མིན་བོད་འཕྲིག་རང་ལ་ཐུན་མོང་མ་ཡིན་པའི་འདེད་ལུགས་དང་རིག་གནས་ཀྱི་སྤྱིད་བཅུད་ཤུན་སྐུམ་ཚོགས་པ་སྤྱན་ཡོད། རྩོམ་ཡིག་འདིར་བརྗོད་བཞིན་པའི་བོད་ཀྱི་བཙོར་འཕྲིག་ནི་བོད་འཕྲིག་རིག་གནས་ཁོད་མཚོན་བྱེད་རང་བཞིན་ཀྱི་འཕྲིག་རིགས་ཤེས་ཡིན་པ་དང་། མིག་སྤང་བོད་ལུལ་དུ་དར་ཁྲབ་ཆེ་བ་དང་ནང་དོན་ཤུན་སྐུམ་ཚོགས་པའི་འཕྲིག་རིགས་ཤེས་ཀྱང་ཡིན། རྩོས་འགོར་འཕྲིག་འདེད་ལག་རྩལ་ཐད་དོན་སྤྱིད་བཅུད་ཀྱི་བཙོར་འཕྲིག་ནི་འཚོལ་བཤེར་བར་སྤྲོད་ཆེ་བ་དང་། འཕྲིག་གི་སྤྱིག་ལམ་མང་བ་འཕྲིག་ངོས་འཕྲུང་སྤྲོ་ཆེ་བ་སོགས་ཀྱི་ཁྲད་ཚོས་ལྟན་པས་རྩོས་འགོར་ཀྱི་བོད་ཀྱི་བཙོར་འཕྲིག་འདེད་པར་ཞིབ་འཇུག་བྱས་ན་ལག་རྩལ་གྱི་དཀའ་གནད་དང་དོན་སྤྱིད་ཆེན་པོ་ལྟན་ཡོད་དོ།།

མིན་ཐུབ་ཀ་ཚོ་ཐབས་ཤེས་དང་UCBཐབས་ཇུས་ཟུང་འབྲེལ་བྱས་བའིMCTSའཆི་ཐབས་ནི་མིག་སྤར་རྩོས་འགོར་འཕྲིག་

འདེད་མ་ལག་ངོད་རྒྱུན་སྤྱོད་ཀྱི་བརྩི་ཐབས་ལེགས་གྲས་ཤེག་ཡིན། འོན་ཀྱང་། བརྩི་ཐབས་འདི་ལ་མི་འདེད་ས་མང་པོ་ཡོད། རྩོམ་ཡིག་འདི་ཙོད་ཀྱི་བཅོར་འགྱིག་གི་ཐུན་མོང་ཡིན་པའི་འགྱིག་གི་འདེད་ལུགས་དང་མིན་ཐུབ་ལུགས་ཁོ་སྤྱོད་འགྲེམ་རྩིབ་ཀྱི་བསམ་བློའི་སྤྱིད་ཤད་བསྡུགས་པ་དང་། UCTབརྩི་ཐབས་ཀྱི་མི་འདེད་སར་དམིགས་ནས་ཙོད་ཀྱི་བཅོར་འགྱིག་གི་འཛགས་རྩམ་དཔྱད་དཔོག་དང་UCTབརྩི་ཐབས་ཟུང་འབྲེལ་ཀྱི་ཐབས་ལམ་ཞིག་སྤྱོན་ཡོད། དེ་མིན་ལག་ལེན་སྤྱིད་ནས་བརྩི་ཐབས་ལེགས་བཅོས་བྱས་པའི་བྱིད་འཛུས་རང་བཞིན་དང་གནད་འགེལ་རང་བཞིན་ར་སྤྲོད་བྱས་ཡོད། གཞན་དུ་དུང་ཙོད་ཀྱི་བཅོར་འགྱིག་རང་འགྲུལ་འགྱིག་འདེད་མ་ལག་TibetanGoཡི་སྤྱིའི་སྒྲིག་གཞི་ལ་ཞིབ་ཕྲའི་དང་དབྱེ་ཞིབ་བྱས་ཡོད་པ་དང་། མ་ལག་སྤྱིའི་ཚ་ནས་འགྱིག་འདེད་དེས་དུས་དང་འཕེན་གཏོང་དེས་དུམ། སྤྱོད་མཁན་དེས་དུམ་བཅས་རིགས་གསུམ་དུ་དབྱེ་ཡོད་དོ།།

མཇུག་ཏུ་དེས་དུམ་དང་དེས་དུམ་བར་གྱི་མཐུན་སྤྱོར་རང་བཞིན་ལ་ཚོད་ལྟ་བྱས་པ་དང་། ཞིབས་མང་པོ་ལ་མི་རྩིས་བར་འགྱིག་དེད་ནས་ཙོད་ཀྱི་བཅོར་འགྱིག་རང་འགྲུལ་འགྱིག་འདེད་མ་ལག་TibetanGoལ་བཀོལ་སྤྱོད་རང་ལྟན་པ་ར་སྤྲོད་བྱས། གཞན་དུ་དུང་མ་གཞི་ཡིUCTབརྩི་ཐབས་དང་ལེགས་བཅོས་བྱས་པའིUCTབརྩི་ཐབས་གཉིས་གཤིབ་བསྟུན་གྱིས་ཚོད་ལྟ་བྱས་པ་དང་། ལེགས་བཅོས་བྱས་བའི་ཙོད་ཀྱི་བཅོར་འགྱིག་གི་རང་འགྲུལ་འགྱིག་འདེད་མ་ལག་TibetanGoཡི་ཐུབ་ཚད67.2%ལ་སྐྱེབས་ཡོད།

གཅོ་གནད་ཀྱི་ཚིག: ཙོད་ཀྱི་བཅོར་འགྱིག ཕྱིན་ཐུལ་ཁ་ལོ་སྤྲོད་འགེམ་འཚོལ་བཤེར། འཇགས་རྩམ་འགྱིག་ངོས་དཔྱད་དཔོག UCTབརྩི་ཐབས།

Research and Implementation of Tibetan-style Chess Man-machine Game System Based on MCTS

ABSTRACT

With the rapid development of computer hardware and software and the Internet, related disciplines have also developed. Among them, artificial intelligence is a research hotspot in recent years, and computer game is an important carrier of artificial intelligence research, and it is also one of the most challenging research directions. The rapid development of artificial intelligence has greatly promoted scientific and technological progress and social development. At home and abroad, research on computer Go, chess, and Amazon has matured. Especially between 2016 and 2017, AlphaGo and AlphaGo Zero, developed by Google's DeepMind team, once again brought the world's attention to the computer game. However, the research in the field of computer chess is just beginning, and it faces great challenges.Although there are many different viewpoints on the origin of Tibetan chess, a large number of documents and unearthed cultural relics can prove that Tibetan chess has a long history, and Tibetan chess has its own unique form of chess and cultural connotation. The Tibetan chess game discussed in this paper is a representative chess piece in the Tibetan chess culture. It is also a chess piece that is widely spread in the Tibetan area and has rich and diverse content. From the perspective of computer game, Tibetan chess has a large search space, many rules for playing chess, and a fast change in disk. Therefore, studying computerized Tibetan chess game has great technical challenges and significance.

The MCTS algorithm combined with the Monte Carlo method and the UCB strategy is a better algorithm commonly used in high-level computer chess programs, but this algorithm also has shortcomings. Based on the unique rules of Tibetan chess and the idea of Monte Carlo tree search, this paper proposes a method of combining Tibetan static chess evaluation and UCT algorithm for the shortcomings of UCT algorithm, and proves theoretically. Then, this paper analyzes the overall framework of Tibetan chess and human game system TibetanGo in detail, and divides it into three modules: game engine module, communication module and game client.

Finally, the coordination between the module and the module was tested. Through many game tests, the practicality of the Tibetan chess and game system TibetanGo is proved. The original UCT algorithm and the optimized UCT are compared and tested. It is concluded that the optimized Tibetan chess player human game system TibetanGo has a game winning percentage of 67.2%.

Keywords: Tibetan chess; Monte Carlo tree search; static situation assessment; UCT algorithm

目 录

第1章 绪论

1.1 研究背景及意义

1.1.1 研究背景

藏式夹棋是藏族棋类文化中不可缺少一部分，也是我们中华民族的一份宝贵的文化遗产，在漫长的历史发展中奠定了自己独特的文化根基。它体现着藏族人民群众的生活哲学,也展示着他们对文化艺术的创造力。曾经是藏民族文化生活中不可或缺的重要内容之一。无论是在广阔的牧区草场还是农区田间村落，随处可见三五成群地围坐在一起下棋、探讨棋局的人们。但随着社会的发展和现代文明的冲击，除了比较偏远的地区外很难再看见人们下棋的情景，在许多藏区 30 岁以下会下藏棋的人甚少。因此，藏式夹棋的传承和发展面临严峻考验。

1.1.2 研究意义

夹棋是藏族民间棋类中有着系统性和全面性着法规则的一项棋类。作为民族传统文化的一部分，在 2006 年青海省将夹棋列为省级非物质文化遗产名录，在青海省夹棋得到了良好的保护和发展[1]。但随着现代社会文明加速发展的步伐，原有的发现、鉴定、登记等＂传统保护模式＂面临着严峻挑战。因而民族传统文化保护需要和信息技术的发展有效结合，将藏式夹棋转变为信息化产品。用＂数字化的保护模式＂为藏棋文化保护提供了一种新思路，促进藏棋文化的保护、传承和发展，对弘扬优秀的藏族民间文化具有十分重要的研究意义。

随着社会的快速发展和计算机的性能不断地提升，近几年来人工智能领域成为了研究热点。计算机博弈又是人工智能的重点研究领域之一，而藏式夹棋是藏族棋类文化中具有权威性和独特性的博弈项目。它与围棋博弈等较成熟的棋类相比，藏式夹棋有着它自己独特的着法规则。藏式夹棋的规则“易学难精”，下棋者应目光长远，不计较眼前的得失。因此它是计算机博弈研究领域中的一个崭新的研究方向之一，对计算机博弈的研究有着深远的意义和影响。

从理论意义方面考虑，计算机藏式夹棋博弈尤为特殊，搜索空间较庞大，盘面评估复杂，遇到了难以终结搜索的局面和难以评估选点的困难，因此常用的计算机博弈原理和方法在计算机藏式夹棋博弈的应用中有着一定的局限性。由此可见，藏式夹棋的智力要求和复杂程度不同于其他棋类，无法将其他棋类程序依葫芦画瓢地套用在藏式夹棋程序上，必须另作思考：将原有的搜索算法和评估方法结合藏式夹棋的着法规则作一定的改进，使计算机博弈程序的水平和效率得到提高。更重要的是藏式夹棋的博弈过程中人类大脑运行的机制与人类智能的本质有直接的联系，从而进一步对人工智能的研究提供了新的出发点，并加深了人类对人工智能乃至计算机科学的研究。

从应用价值方面考虑，开发一款藏式夹棋博弈软件，将计算机藏式夹棋软件产业化，可以取得很好的经济效益和社会效益。例如，计算机夹棋软件可用于藏式夹棋的教学和传承，还可以满足藏式夹棋爱好者对人机博弈的需求等。

1.2 国内外研究现状

1.2.1 国外研究现状

计算机博弈是人工智能研究的重要载体，也是目前最具有挑战性的研究方向之一[2]。20 世纪 50年代开始，冯诺依曼就提出了极大极小定理用于解决博弈问题。1950年，信息论的创始人香侬教授，又给出了极大极小算法，提出了为计算机象棋博弈编写程序的方案，成为了机器博弃的创始人[3]。1958 年 阿 伯 恩 斯 坦 （Alex Bernstein）等在 IBM704 机上开发了第一个达到成熟孩童博弈水平的国际象棋程序[4]。 1959 年，人工智能的创始人之一塞缪（A.L.Samuel）编了一个能够战胜设计者本人的西洋跳棋程序，1962年该程序击败了美国的一个州冠军[5]。1997年,美国IBM公司著名的国际象棋计算机“深蓝”(Deeper Biue）战胜了当时的国际象棋世界冠军卡斯帕罗夫（Gairy Kaspariv）[6]。2016年 1 月 Google旗下DeepMind开发的Alpha Go在没有任何让子的情况下以 5 ： 0 击败了欧洲围棋冠军。2016年 3 月Alpha Go再次 4 ： 1 击败韩国围棋高手李世石，成为围棋博弈发展的里程碑。2017年10月19日，DeepMind团队重磅发布AlphaGo Zero，再次震惊世人。相比上一代Alpha Go，该版本的Alpha Go实现了在AI发

展中非常有意义的一步——“无师自通”[7]。

蒙特卡洛树搜索是 UCT 算法结合蒙特卡罗评估就形成的一种算法。蒙特卡罗方法（Monte Carlo，简称 MC），或称随机模拟方法，它是以概率统计理论为基础的一种方法。在第二次世界大战期间，为了解决原子弹研制中裂变物质的中子随机扩散问题，美国数学家冯.诺伊曼和乌拉姆等提出了蒙特卡罗模拟方法。由于保密的原因，就用摩纳哥一个赌城的名字—蒙特卡罗命名[8]。在游戏中使用蒙特卡罗方法可以追溯到 1973 年，Widrow 等人将蒙特卡罗模拟应用到了扑克 21 点。2006 年，匈牙利的研究者 L.Kocsis 和 Czepesvari 将一种 UCT 算法结合蒙特卡罗评估应用于计算机围棋中，标志着进入现代计算机围棋博弈时代[9]。尽管上世纪九十年代初 Monte-Carlo 方法就已经被用来处理计算机围棋问题，但由于它不能在探索与利用之间很好的找到平衡点，因此在静态评估算法程序面前处于下风。2016 年 3 月，Alpha Go 运用了一种综合使用蒙特卡洛搜索树、决策网络和价值网络的算法以 99.8%的胜率大胜其它围棋程序。

1.2.2 国内研究现状

国内对于机器博弈的研究时间较晚，2003 年，台湾交通大学吴毅成教授发明了六子棋 （connect 6 ）[10]。之后，东北大学徐心和教授和他的团队研究开发了中国象棋软件“棋天大圣”，具有挑战国内中国象棋顶级高手的实力[11]；北邮刘知青带领学生开发的“本手（LINGO）”围棋程序，能够战胜高水平业余围棋选手[12]。2007年中国人工智能学会机器博弈专业委员会正式成[13]。机器博弈专业委员每年都会举办一次国内机器博弈的主要赛事＂全国大学生计算机博弈大赛＂，博弈大赛为全国的机器博弈研究人员提供一个交流、学习、展示自身成果的平台，极大的推动了机器博弈在国内的发展。2015年全国计算机博弈大赛经过 8 年的发展，首次纳入全国智力运动会，设置包括锦标赛和大学生博弈大赛共计17个项目，比赛队伍有230支之多，发展迅速成绩突出[14]。机器博弈专业委员会的成立标志着国内机器博弈开始迈入高速发展的时代。目前蒙特卡洛树搜索算法已经应用到围棋、亚马逊棋领域，但在藏棋领域仍是空白。

1.2.3 藏族夹棋人机博弈系统的研究现状

在藏民族悠久的文化历史长河中，其先民创造出了独特的藏棋文化。有关藏棋的起源在汉、藏史料中都没有详细准确的记载。在“《唐书》和《敦煌吐蕃历史文书》中记载有吐蕃大臣下棋的故事，说明在吐蕃时期，藏棋就以及在社会上流行了[15]。1996年西藏聂拉木县出土的227件石器中，有石棋盘和棋子[15]。1999年，在拉萨市墨竹工卡县（吐蕃松赞干布出生地）的“强巴敏久林宫殿”遗址，发现了石质的“密芒”棋盘[16]。2016年，在阿里地区札达县托林寺，发现了一块非常完整的古代密芒棋盘，据托林寺主介绍，此棋盘在该寺已有 1 千多年了[15]。以上证明西藏的弈棋活动历史悠久。19世纪中叶，藏族天文历算家丹巴加措撰写了《密芒吉单居》一书，翻译成汉语就是———“藏棋之理论”。据说现存于甘肃甘南夏河拉布楞寺[17]。2011年青海民族大学开发了全国第一款藏族棋类软件。2012年青海民族大学裴生雷设计开发王官双门棋（国王和大臣棋）博弈系统[18]。2014年青海民族大学仁增多杰设计开发了“搭链”棋[19]。2016年青海民族大学才让拉毛设计开发了＂鱼儿＂游戏软件[20]。2016年青海民族大学仁青东主设计开发了藏族王棋[21]。以上棋类只实现了单机游戏，而没有人机博弈系统。2017年5月中央民族大学邓颂庭设计开发了基于贝叶斯网络结构的久棋博弈系统[22]。2018年5月中央民族大学李霞丽设计开发了基于棋型的藏族“久”棋计算机博弈系统[23]。但在藏族夹棋博弈系统方面仍然处于空白阶段。

1.3 主要研究内容

本文中计算机夹棋博弈中的算法问题是主要的研究内容。在学习和研究现有理论和实践成果的基础上，对目前较为先进的蒙特卡洛方法和上限信心界应用树算法构成的蒙特卡洛树搜索应用到藏式夹棋中，并结合藏式夹棋的着法规则进行一定程度的改进，提高搜索和评估的性能和效率，最后进行了验证和分析。具体研究内容为：

(1) 借鉴围棋等其它棋类，研究藏式夹棋的棋盘和棋子的表示、棋局的表示和存储、棋规的实现等。

(2) 了解计算机博弈的发展与现状；从理论上学习和研究计算机博弈的理论

和方法，特别是借鉴计算机围棋采用的评估和搜索方法；从理论上研究计算机藏式夹棋博弈的主要特点与难点。

(3) 从理论上研究极大极小算法和 Alpha-Beta 剪枝算法，并深入探索蒙特卡洛方法，多臂匪徒模型和上限信心界应用树算法等诸模型算法在计算机藏式夹棋博弈中的应用。

(4) 结合藏式夹棋的着法规则，将静态评估和 UCT 结合的优化算法应用到藏式夹博弈中，提高搜索和评估的性能和效率。

(5) 实现了计算机藏式夹棋博弈系统 TibetanGo。基于 C#开发工具实现了计算机藏式夹棋人机博弈系统，并进行了数据测试，最后给出了实验结果。

1.4 论文组织结构

全文共分为六章，每章内容要点如下：

第1章 绪论。该章主要介绍研究背景和选题意义以及国内外对藏式夹棋人机博弈系统研究状况，明确了研究对象、目标和难点。并阐述了论文组织结构；

第 2 章 计算机藏式夹棋博弈的相关理论知识。该章节首先介绍了藏式夹棋的基本规则，然后介绍了计算机博弈的相关理论与方法，包括博弈树的概念和局面评估、博弈树常用的搜索算法等；

第3章 藏式夹棋博弈中MCTS的改进与应用。本章首先介绍了蒙特卡洛方法，然后介绍了蒙特卡洛方法应用在博弈树搜索中的蒙特卡洛树搜索。其次介绍了一种完备的蒙特卡洛树搜索算法UCT，最后将该算法应用在藏式夹中，并按藏式夹棋的特点提出了一种静态局面评估与UCT算法结合方的MCTS算法。

第4章 基于MCTS藏式夹棋博弈系统的设计与实现。 该章介绍了藏式夹棋博弈系统整体架构、数据结构和博弈流程、系统中各种功能模块的设计与实现。

第5章 实验与结果分析。本章对藏式夹人机博弈系统的棋力做了总体测试。主要包括应用原MCTS的博弈系统和改进的博弈系统进行对比测试，最终对

测试结果进行分析。

第 6 章　总结与展望。对整个研究工作进行了总结，并对今后需要做的研究工作进行了展望。

第2章 计算机藏式夹棋博弈的相关理论知识

随着计算机的出现，计算机博弈的研究也慢慢起步。在十几年研究过程中人们提出了很多的研究方法，并且也有满意研究成果。在多年的研究中计算机博弈要解决问题主要是搜索和评估。

2.1 藏式夹棋介绍

2.1.1 藏式夹棋基本规则

藏式夹棋起源于藏族漫长的历史长河中，是藏棋文化中的一个重要组成部分。夹棋，藏语音译为“孜久”。 其中“孜”指从左右两方相持，从两旁限制的意思，而“久”是对弈的意思。藏式夹棋主要流行于安多地区，特别在青海地区有很多人都会下此棋。藏式夹棋也类似于围棋等大多数棋类，属于双人零和有限确定完全信息棋类游戏的一种。双人，指的是下棋过程中有两个下棋人[9]；零和，是指参与下棋的两方，在激烈的对弈过程中，一方赢了另一方必定是输了，下棋两方的收获和损失总和一定为“零”；有限，是指对弈到了一定程度会结束；确定，是指当棋局结束时不存在不确定的因素；完全信息，是指在博弈过程中双方的信息彼此完全了解[9]。每一方，不仅清楚当前的棋盘局面和对方过去行棋的所有步骤，而且还能估计对方未来的可能会走棋的步骤。藏式夹棋的棋子只有两种，是黑棋和白棋。夹棋棋盘种类比较多，较常见的形态有 5×5，7×7，9×9 的棋盘。

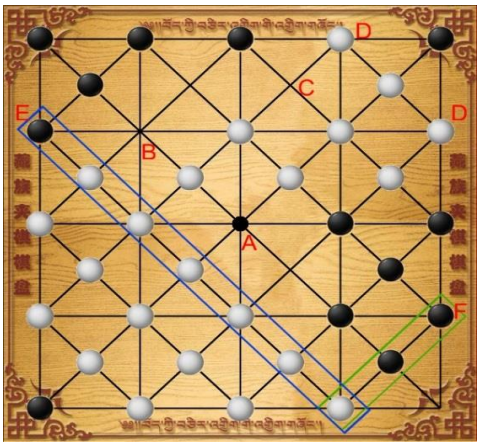


图 2-1 藏式夹棋 5×5 棋盘

本文研究的是五路（5×5）形态的藏式夹棋。棋盘是由五条横线和五条竖线交叉，另外交叉线形成的小方格的对角线，共组成四十一个棋点。棋盘中心点（在棋盘上的 A 点）称为首府或棋路，小方框中两条直线交叉的点为焦点（如棋盘上的 C 点）。大方框中的四条线交叉的点叫关节点（如棋盘上的 B 点）。

藏式夹棋的基本规则如下：

1) 落子

对局双方各执行一枚棋子。先落子一方后移的原则。首先正中间的点首府上不能落子，正中间的首府是所有棋子落完后移动的地方。其次寻找有没有可吃的棋子，并阻挡对方侵入自己的地域，然后对弈双方轮流地落在空棋点上。落子有插入法（己方的棋插入到对方的棋行中）、逼法（对方构成威胁的着发，大多数用于夺取对方周围的根据地）封法（封锁对方棋子的移动出路）等落子方式。

2) 移子

除了正中心的点，棋盘的其余点全部下满后，棋子开始移动，每步移动一格，上下左右都可以。此时，后落子的先移。 移动到合理的点上，能移动到正中心的点上，则可以移动到正中心的点上。

3) 吃子

藏式夹棋规则中落子或移子的同时可以吃掉对方一枚或多枚棋子，只要打抢法、夹发等满足条件的情况下一次可以吃对方多枚棋子，每次吃掉对方一枚棋子后，并用己方棋子来填补该棋位。如果新落的棋子又能吃子，则继续吃子，直到没有可以吃的棋子为止。

4) 打抢法

棋盘的横向、竖向、斜向的任意边上除端点有一枚敌棋外，在这条线上其余的所有点上都有自己的棋子，则可以吃掉对方端点的棋子。“打枪”分为“长枪”（如在棋盘上的 E 方框）和“短枪”（如在棋盘上的 F 方框）。

5) 夹法

在一条直或斜线上，对弈的一方用两枚棋子将另一方的棋子夹在中间行棋的方法（如在棋盘上的 D 点）。只要能夹住对方的棋子，可以吃对方多个棋子，每吃一枚对方的棋子，并在该棋位补放一枚己子。

6) 让路

在对弈的过程中任意一方没有移动的路线时，则重新移动或跳棋给对方让路，在让路是可进行夹棋吃子，但不许将对方的行棋路线故意堵住。

7) 判断胜负

为终局：一是对局一方认输。另一个是一方将对方的全部棋子夹完时，也就是一方的棋全部被吃光，无棋可走则终局。

2.1.2 计算机藏式夹棋博弈简介

计算机博弈是在人工智能领域具有挑战性的研究方向之一，也可以称为机器博弈。简单的讲，就是让计算机像人一样下棋并思考。而计算机藏式夹棋博弈是指计算机构造计算方法或程序，是计算机可以像人一样进行博弈。因为藏式夹棋具有它自己独特的博弈规则，与其他计算机博弈项目相比，计算机藏式夹棋存在以下三个特点：

1）搜索空间大

虽然藏式夹棋的棋规则简单易懂，但实际的对弈过程中相当复杂。藏式夹棋的规则“易学难精”，几乎每一次行棋都会有大量的落子提供让对方选择，所以每一次行棋的搜索空间很大。根据复杂度的概念得出，藏式夹棋的棋盘上大约有 1.22×10^{19} 个不同的盘面。

2）局面评估复杂

由于在藏式夹棋对弈过程中，棋盘上棋子的数量和局势判断的容易度之间没有太大的关系，对计算机来说都一样困难。主要是藏式夹棋的落子布局阶段是不能移动棋子，所以布局阶段棋子的位置是固定的，且每个棋子的功能理论上都是相同。但实际的功能和重要程度是棋子和棋子之间的关系和位置来决定。另外满盘后要开始移动。这时棋子位置改变会影响棋子与棋子之间的关系，因此每一枚棋子的实际功能也会随着变化。从而导致了藏式夹棋的局面评估很复杂。

3）局面评估和博弈树搜索关系密切

在计算机博弈中局面评估和博弈树搜索是要解决的两个核心问题。随着棋类的规则不同，两者之间的关系密切程度不同。而在藏式夹棋中，局面评估和博弈树搜索关系密切。因为棋子之间的关系不确定，想要知道一枚棋子与其他棋子之间准确的关系，就必须搜索未来可能的着子点。在这一点很显然地明白，藏式夹

棋的静态局面评估方法并不能计算准确估值，所以只能通过搜索来计算相对准确的局面估值。

2.2 博弈树

目前，在计算机博弈领域中属于双人零和博弈的棋类是主要的研究对象之一。双人博弈中，当前初始盘面的基础上，双方按照棋规轮流下子直到结束，以获取最大的收益并选择最佳的落子点是计算机博弈理论的目标。

计算机博弈树描述的是博弈双方从最初的状态出发，根据行棋的先后次序将博弈双方可以落子的点抽象地表示为一个有很多分支的树形结构。从逻辑上看博弈树是一棵“与或树”，对弈时所有的可以行棋着法都用树的分枝来表示，由行棋规则产生的不同棋局在树中都用不同的节点来表示，理论上用棋规则所行棋的步骤也称之为博弈树的展开。

不妨用井字棋来举例说明。假设图 2-2 中的“X”和“0”为博弈过程中的博弈双方，$S_{(0,0)}$为当前的初始状态，如果首先是“X”来落子。那么可以建立一个博弈树来描述博弈双方可能的落子状态。其中，$S_{(0,0)}$为博弈树的根节点，“X”在盘面为 $S_{(0,0)}$的前提下所有可能的落子点会生成一个新树枝 $S_{(1,0)}$……$S_{(1,n)}$，这些新的树枝都是 $S_{(0,0)}$的子节点。之后“0”会将 $S_{(0,0)}$的每一个子节点为基础，生成一个新的树枝 $S_{(1,0)}$……$S_{(1,n)}$。这些是 $S_{(0,0)}$的子节点的子节点，也就是 $S_{(0,0)}$的第二层子节点。整个博弈树会将这种形式构建起来，直到不能生成新的动作时，表示博弈树构造结束。这时每一个叶子节点都代表着博弈终结，并可以在叶子节点上评估最终的胜负。

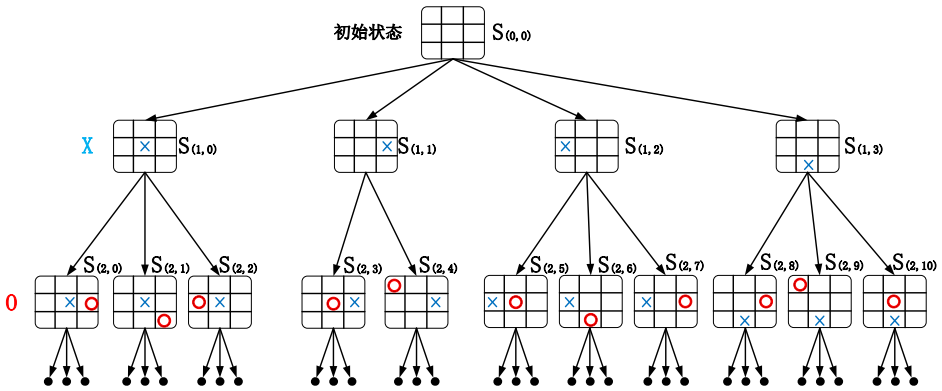


图 2-2 井字棋的部分博弈树

在实际应用中，像藏式夹棋这种行棋规则复杂、局面变化很快的博弈问题而言，其博弈树的节点数是非常多，无法生成一个完整的博弈树，因为计算机效率和对弈的时间是有限的。在有限的时间和部分的博弈子树中，想要选择一个最佳的落子点就与下面的博弈树局面评估和博弈树的搜索息息相关。

2.3 博弈的局面评估

局面评估是衡量当前局面对己方好坏的标准。保证博弈引擎较快地找到最佳着法点的关键是要有一个良好的局面评估函数。而局面评估函数对整个棋局的综合评估的好坏直接影响着计算机博弈系统棋力的强弱。计算机博弈的局面评估方法有静态局面评估和动态局面评估。下面将两种局面评估方法做了详细介绍。

2.3.1 静态局面评估

静态局面评估方法是传统计算机博弈中最常用的方法。静态局面评估也称为专家系统方法，采用人工智能等技术大量地搜集和整理某个棋类的专业知识、方法与经验，从而达到人类专家处理此棋类较为复杂的智能问题的水平。而在计算机博弈中静态局面评估方法主要是通过棋知识来总结出一定的规律，并对当前局面进行量化判断评估。在静态局面评估时，尽可能为了获得准确的评估结果，所需要考虑的问题很多。比如：棋盘的基本状况、棋盘中棋子的数量、棋子的排列形状、棋子的位置、棋子与棋子之间的关系等等。并需要对该棋的知识具有透彻的理解和精湛的棋艺。具体的评估方法还是取决于棋类本身的性质与特点。

静态局面评估或专家系统方法在一定范围内有可以得到较为有效的结果，但随着棋的知识和规则越复杂，静态局面评估方有着一定缺陷。比如像藏式夹棋的知识和规则很难归纳，归纳之后也有例外。并且人工构建藏式夹棋知识和规则的过程中，只能归纳一些局部和底层次的知识和规则，因此局面评估时有一定的局限性。局面评估的准确性将直接影响搜索的方向，同时局面评估的效率也对搜索性能有着直接的影响，从而静态局面评估方法应用在计算机博弈系统中有一定的局限性。

2.3.2 动态局面评估

动态局面评估是不同于静态评估。动态局面评估在某一个棋局状态在整个博弈树中所在的位置和对应的关系，以及与先辈棋局的变化而变动。在近几年的计算机博弈程序中蒙特卡洛评估方法是最为典型的动态局面评估方法。它不像静态局面评估方法那样棋子与棋子之间关系或棋盘的某个点与其它点之间的逻辑关系都是固定，在同样的对弈局面中，评估出来的结果相同。而蒙特卡洛评估方法随着棋盘上棋子与棋子之间位置变化，它评估的结果也跟着变动，因为它用大量的随机模拟，并统计每一次模拟的胜负结果来评估某一个局面。这也就在一定程度上避免了预定义知识的不确定而造成的评估结果的不准确[26]。

2.3.3 局面评估的准确性和性能

在计算机评估棋盘局面时，通常会遇到两个互相矛盾的因素要解决：准确性高低与性能好坏[9]。当计算机评估的越来越准确时，博弈水平也同时提高，但是博弈性能对博弈水平很受限制一个因素，计算机博弈最理想的状态是要肯定有一定准确率来评估，也要保证访问局面树尽量的多一点。在评估过程中考虑的越细致准确率越高，但是对计算时间的消耗很大，局面访问的数量也就不多了，从而影响搜索博弈树的深度与广度。因此在准确性和博弈性能之间找出一个平衡的点极为重要。

2.4 常用搜索算法

2.4.1 极大极小算法

极大极小算法英文缩写名称为 Minimax 算法，是把己方的利益最大化，把对方利益最小化的一种博弈树搜索算法。极大极小过程起源于双人零和博弈理论，即适合用于像下棋这样轮流对弈的情况。它的核心思想是在博弈过程中己方最小的损失中获取尽可能大的收益。极大极小算法在传统计算机博弈中得到了非常广泛应用，并在计算机博弈的发展史上有着重要的贡献。

极大极小定理是由著名的数学家计算机之父约翰•冯•诺依曼（John Von

Neumann）提出。极大极小定理为：对于某一个状态有限的双人零和博弈，可以取出一个评估值 V 与适用于对弈双方的一种策略，同时满足以下条件：（1）已知对弈方 B 的策略，对弈方 A 的策略，对弈方 A 的最好的收益是 V；（2）已知选手 A 的策略，选手 B 的最好回报是-V[24]。

在双人零和博弈中，由于博弈双方的利益矛盾，可以将努力争取获胜的一方称为 Max，尽量将 Max 方的利益做到最大化；对弈另一方则用来 Min 表示，在对弈过程中 Min 方的利益控制在最小的范围；两者是完全对立的。极大极小算法的基本思想如下：

（1） 当轮到 MAX 走棋时，MAX 方应该考虑，MAX 方所有合理的走法中最好的情况，即评估函数取极大值。

（2） 当轮到 MIN 方行棋时，MAX 方应该选择 MIN 方所有的走法中最坏的走法，也就是选择评估函数计算的估值为最小的那个走法。

（3） 按照双方对弈的原则，轮流使用（1）和（2）两种方法传送评估值。

如图 2-3 所示，假设某一个棋局的初始状态 $S_{(0,0)}$，MAX 方和 MIN 方服从轮流行棋的规则，MAX 方先行棋。在博弈树中 MAX 方用圆圈的节点来表示，MIN 方用方框的节点来表示。叶子节点上的数值为棋局在当前状态下评估函数计算的估值。

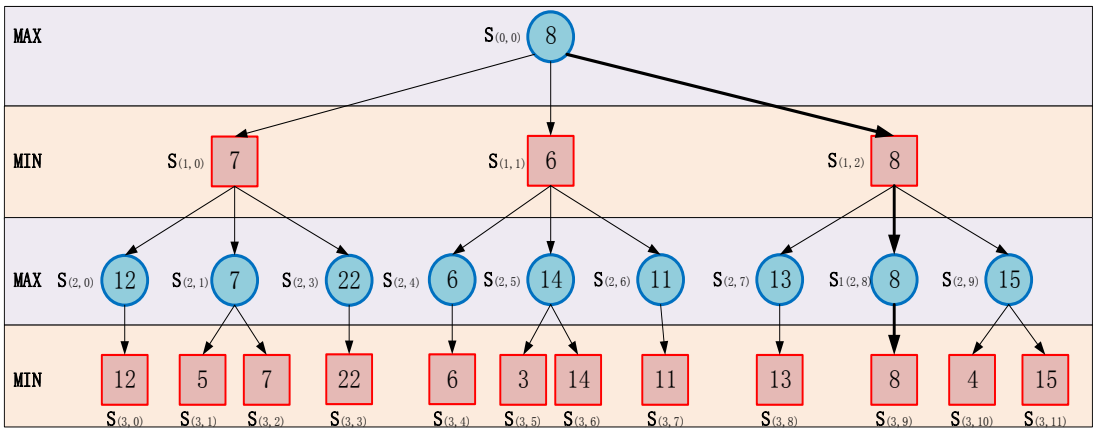


图 2-3　极大极小搜索推导示意图

采用极大极小搜索算法进行对博弈树搜索。假如评估函数 f 对第四层的所有节点计算的估值为：

$f(S_{(3,0)})=12$；　$f(S_{(3,1)})=5$；　$f(S_{(3,2)})=7$；$f(S_{(3,3)})=22$；$f(S_{(3,4)})=6$；$f(S_{(3,5)})=3$；$f(S_{(3,6)})=14$；$f(S_{(3,7)})=11$；$f(S_{(3,8)})=13$；$f(S_{(3,9)})=8$；$f(S_{(3,10)})=4$；$f(S_{(3,11)})=15$；

在此基础上由自低向上推到第三层为 MAX 方，则计算最大值为：

$f(S_{(2,0)})=MAX\{f(S_{(3,0)})=12\}=12;$

$f(S_{(2,1)})=MAX\{f(S_{(3,1)}),f(S_{(3,2)})\}=7;$

$f(S_{(2,3)})=MAX\{f(S_{(3,3)})\}=22;$

$f(S_{(2,4)})=MAX\{\ f(S_{(3,4)})\}=6;$

$f(S_{(2,5)})=MAX\{\ f(S_{(3,5)}),f(S_{(3,6)})\}=14;$

$f(S_{(2,6)})=MAX\{f(S_{(3,7)})\}=11;$

$f(S_{(2,7)})=MAX\{f(S_{(3,8)})\}=13;$

$f(S_{(2,8)})=MAX\{f(S_{(3,9)})\}=8;$

$f(S_{(2,9)})=MAX\{f(S_{(3,10)}),f(S_{(3,11)})\}=15;$

第二层为 MIN 方，则计算最小值为：

$f(S_{(1,0)})=MIN\{f(S_{(2,0)}),f(S_{(2,1)}),f(S_{(2,3)})\}=7;$

$f(S_{(1,1)})=MIN\{f(S_{(2,4)}),f(S_{(2,5)}),f(S_{(2,6)})\}=6;$

$f(S_{(1,2)})=MIN\{f(S_{(2,7)}),f(S_{(2,8)}),f(S_{(2,9)})\}=8;$

在到根节点为 MAX 方，则计算最大值为：

$f(S_{(0,0)})=MAX\{f(S_{(1,0)}),f(S_{(1,1)}),f(S_{(1,2)})\}=8;$

因此此棋局状态为 $S_{(0,0)}$ 时最佳的行动路径为 $S_{(1,2)}$， $S_{(2,8)}$， $S_{(3,9)}$。

2.4.2Alpha-Beta 剪枝算法

alpha-beta 剪枝算法是于 1958 年由美国卡耐基•梅隆大学的三位学者最先提出的。alpha-beta 剪枝算法是基于极大极小搜索的一种算法。极大极小算通过搜索有一定深度的整棵博弈树，然后挑选对己方最有利的节点序列。这种搜索方式的效率非常低。如果博弈树非常庞大，在有限的时间和空间内无法将博弈树完全展开。因此首先访问博弈树的前几个节点，然后用叶子节点上的评估值作为启发信息，那样尽量可以搜索到最深的一层。但极大极小算法不可能搜索到很深的节点，因为在博弈树上有很多的子树。

极大极小算法的短板被 alpha-beta 剪枝算法有效的改进了。alpha-beta 剪枝算法会启发式的搜索，确定对己方以后的局势不好的着法会从根节点的分枝开始剪掉，从而在有限的时间内能搜索到很深的节点。这样不仅提高了搜索效率，还提高博弈水平。

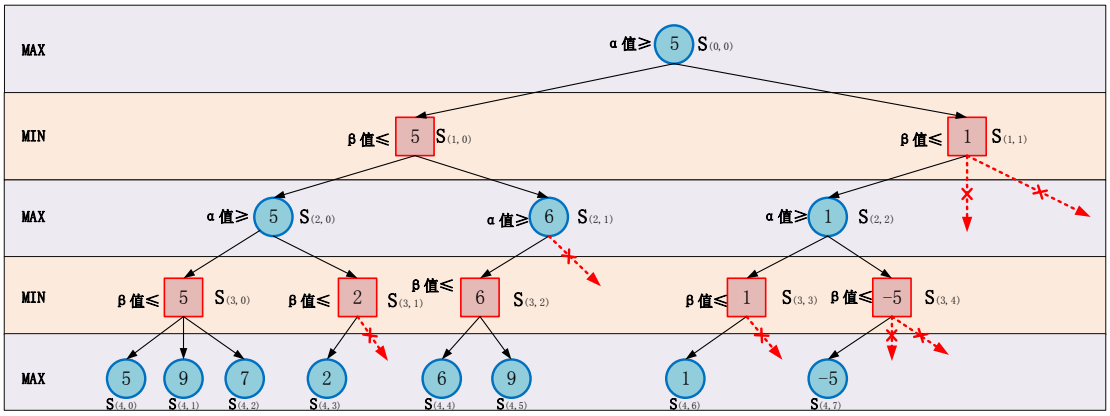


图 2-4　alpha-beta 剪枝示意图

图 2-4 为 alpha-beta 剪枝示意图，其中最下面一层的叶子节点里边的数字是假设的评估值。在该图中，$S_{(4,0)}$、$S_{(4,1)}$、$S_{(4,2)}$的估值推出节点 $S_{(3,0)}$的到推值为 5，即 $S_{(3,0)}$的 β 值为 5，由此可推出节点 $S_{(2,0)}$的到推值≥5。记 $S_{(2,0)}$的到推值的下界为 5，不可能再比 5 小，故 $S_{(2,0)}$的 α 值为 5。由节点 $S_{(4,0)}$的估值推知节点 $S_{(3,1)}$的倒推值小于≤2, 无论 $S_{(3,1)}$的其它子节点的估值是多少，$S_{(3,1)}$的倒推值都一定不能比 2 大。因此，2 是 $S_{(3,1)}$的倒推值的上界，所以 $S_{(3,1)}$的值≦2 。另已知 $S_{(2,0)}$的倒推值≥5，$S_{(3,1)}$的其它子节点又不可能使 $S_{(2,0)}$的倒推值增大。因此对 $S_{(3,1)}$的其它树枝不必再搜索下去，同等于把这些树枝已剪去。由 $S_{(3,0)}$、$S_{(3,1)}$的倒推值可推出节点 $S_{(2,0)}$的倒推值≥5 ，再由 $S_{(2,0)}$可推出节点 $S_{(1,0)}$的倒推值≤5，即 $S_{(1,0)}$的 β 值为 5。另外，由节点 $S_{(4,4)}$、$S_{(4,5)}$推出的节点 $S_{(3,2)}$的倒推值为 6，因此 $S_{(2,1)}$的倒推值≥6，即 $S_{(2,1)}$的 α 值为 6。此时，$S_{(2,1)}$的倒推值无论是多少它的子节点的都不能将 $S_{(2,1)}$及 $S_{(1,0)}$的倒推值减少或增大，所以 $S_{(2,1)}$的其他分枝被减去，并可确定 $S_{(1,0)}$的倒推值为 5 。以此类推，最终推出 $S_{(0,0)}$的倒推值为 5。

2.5 本章小结

本章主要介绍了计算机博弈技术的博弈树和局面评估、常用的搜索算法等几个理论基础。同时也介绍了藏夹棋的棋盘规划和行棋规则。博弈树和局面评估是计算机博弈要解决的两大问题也最主要的研究对象。因此，进行详细介绍和分析了彼此之间的关系。搜索算法是计算机博弈实现的关键技术，在本章介绍一个经典的搜索算法，并分析它们的优缺点。

第3章 藏式夹棋博弈中 MCTS 的改进与应用

藏式夹棋的棋局可以分为开局阶段的移局阶段，在开局阶段应用高效的搜索算法和评估方法来布好局，是在移局阶段获胜的关键。因此博弈树生成和搜索时要分别考虑开局阶段的情况和移局阶段的情况。这两个阶段的博弈树生成和搜索的方法要有一定的区别。

3.1 蒙特卡洛方法

蒙特卡落(MC)是一种基于概率统计和大数定理的数值计算方法。科学家很久以前就发现并采用了蒙特卡洛方法，但该方法在第一次世界大战才提出来。在美国研制原子弹时著名的数学家 John Von Neumann 和 Stan Ulam 首次提出。蒙特卡洛方法是一种依赖重复随机模拟来获得最终结果的一种概率算法，该方法通过多次的模拟运行来计算那些建立于同一个概率分布上的事件，最终统计出一个近似又有效的结果[25]。通常蒙特卡洛方法用于求解优化问题、数值计算问题以及从一个概率分布之中产生抽样的问题[25]。使用蒙特卡洛方法，始终可以获得问题的解决方案，尽管此解决方案不一定是该问题精确的解决方案，蒙特卡洛方法求得精确解决方案的概率会随着计算时间的增加和模拟的次数而增大[25]。

蒙特卡洛方法通过大量的模拟抽样得到结果，方法本身不属于任何领域知识，是一个纯数学模型，因而应用领域也不属限制。可以说明蒙特卡洛方法基本思想的最典型的实例是计算不规则图形面积的问题。

如图 3-1 所示，因为计算不规则图形的面积没有一个准确的公式，因此用规则图形正方形来围住不规则图形，然后可以用蒙特卡洛方法来求解这个不规则图形的面积。假设随机向这个正方形内投点，投到一定数量后分别统计正方形内的点个数和不规则图形内的点个数。假设正方形中有 1000 个点，不规则图形中有 850 个点。如果投一个点并投入图形内部作为事件，数学上不规则图形的面积等于这个事件发生的概率。

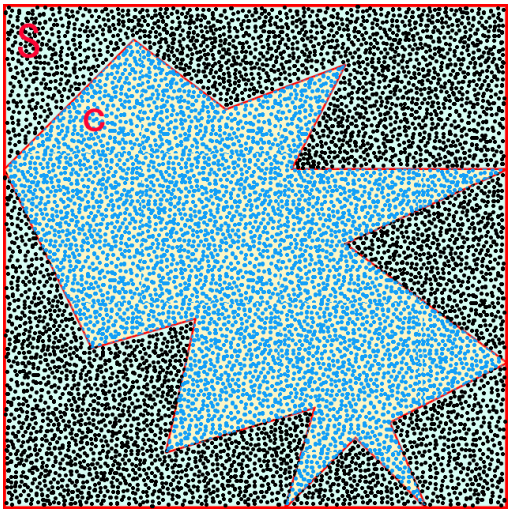


图 3-1 不规则图形面积求解示意图

正方形面积 S，不规则图形面积 c，事件：当随机投一个点并在落在不规则图形中，投点服从二项分布，所以 P(A)=c/S；k 次投点中 m 次在不规则图形中，根据大数据定理：

$$\lim_{k\to\infty} P\left\{\left|\frac{m}{k}-P(A)\right|<\varepsilon\right\}=\lim_{k\to\infty} P\left\{\left|\frac{m}{k}-\frac{c}{S}\right|<\varepsilon\right\}=1 \qquad (3-1)$$

在公式(3-1)当 k 趋于无穷大时，频率 m/k 根据概率敛到 c/S。此时可以估计不规则图形的面积 c 的大小为：

$$c=1\times\frac{850}{1000}=0.85$$

虽然蒙特卡罗方法的想法早已被提出，但由于其对大量模拟的强烈依赖性，因此未得到有效应用。计算机的出现，使这种依赖快速、大量模拟的方法成为了可能，并应用领域也越来越广泛。

3.2 蒙特卡洛树搜索及其在藏式夹棋博弈中的应用研究

MCTS 是一种基于蒙特卡罗方法思想的规划方法。跟传统的博弈树的生成，蒙特卡落树搜索算法根据最初的棋局形势开始反复地给出随机模拟事件，并渐渐扩散博弈树的每个分支，每个模拟事件都是“当前形势-动作-收益”的三维序列。因此，蒙特卡罗树搜索过程是相对于在未评估之前扩展博弈树的静态评估方法而言是动态的搜索方法。

蒙特卡洛树搜索算法可以分为四个基本步骤，分别为选择步骤(selection)、扩展步骤(Expansion)、模拟步骤(Simulation)和回溯步骤(Back propagation)，如图 3-5 所示。对弈过程中某一个局面的状态是用图中各个节点表示，节点与节点之间每条边表示在父节点上采取的动作并获得与子节点相对的状态。要完成一次搜索就要依次执行完这四个基本步骤。

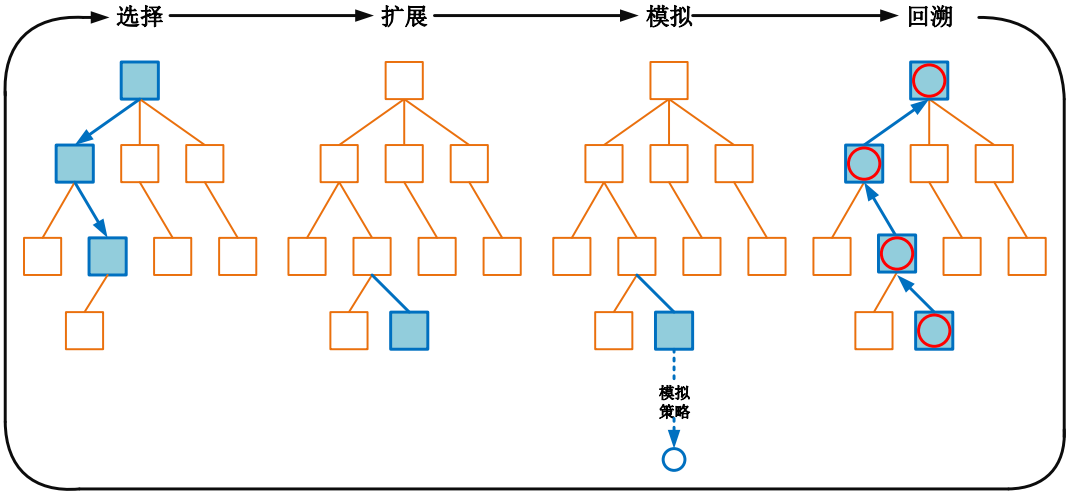


图 3-5 蒙特卡洛树搜索算法的流程

1. 选择：从根节点出发，在搜索树中从上到下在所有合法的节点中随机选择一个叶子节点；
2. 扩展：将一个或多个合法的节点添加到所选择节点下以基于当前执行动作展开博弈树；
3. 模拟：根据扩展出来的一个或多个节点上，按棋规则计算机随机模拟，直到整个棋局结束并确定当前节点的评估值；
4. 回溯：根据模拟结果从原路返回并更新路过的所有节点的评估值。

```
算法：蒙特卡洛搜索(MCTS)
    function MCTS
      while 尚未达到循环次数或预设时间 do:
            {
            Node StochasticNode = Tree_Select(当前根节点)  随机选择一个节点；
            Node NewNode=Extend(StochasticNode)  给选择按棋规节点添加
            double Reward=Simulate_policy(NewNode)  模拟走棋并计算收益值；
            Backup(StochasticNode, Reward) 回溯到根节点并一路更新收益值；
            }
       Node Best_Node(当前根节点) 获取胜率最大的节点
```

通过以上方法，最终可以给计算机藏式夹棋博弈系统提供一个落子点，但提供的落子点不是最佳点。当然，这种解决方法并不直接用于计算机藏式夹棋博弈中，因为蒙特卡洛树搜索对模拟对弈次数有很高的要求，而对弈次数多了收敛效率低。因此在下面介绍以此算法为基础的效率更高的算法。

3.3 UCB 策略

UCB(Upper Confidence Bound，简称为 UCB)策略是奥地利格拉科技大学的 AUER 于 2002 年提出。该策略是为了解决多臂老虎机(如图 3-1)一类问题而产生的，多臂老虎机问题(multi-arm bandit)是指有多个拉杆的老虎机，每个拉杆中具有不同的收益，拉动拉杆获得收益，要想获得最大的收益就必须要拉动收益最大的拉杆。在没有任何先验知识的情况下，判断拉动那个拉杆收益最高是问题的关键。UCB 策略采用了离线机器学习策略，根据拉杆访问次数等信息，通过学习解决下次拉动那个拉杆的问题。

图 3-2 单臂老虎机

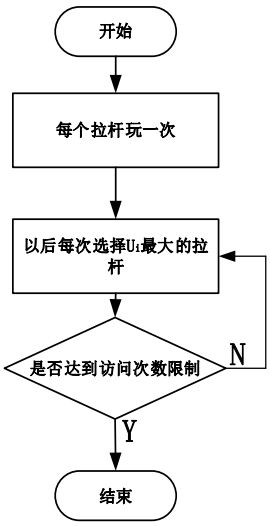


图 3-3 UCB 算法流程图

UCB 策略首先将拉动每个拉杆的收益归一到[0,1]的区间内，而对于决策的计算值包含了两个部分，第一是已知收益的均值，用于表示已经获取到的经验；第二部分是与平均收益在单侧置信区间的大小有关，用于保证所期待的收益能以极大的可能落在平均收益范围内[26]。UCB 算法是 UCB 策略中最基本的算法，其 UCB 算法的基本过程为：

```
算法：信心上限算法（UCB）
    function UCB
        for each 拉杆 i:
            访问该拉杆并记录收益
        end for
        while 尚未达到访问次数限制 do:
            计算每个拉杆的 UCB1 信心上界 Ui（如下所述）
            访问信心上界最大的拉杆
        end while
```

对于每一个拉杆的 U_i 值，其计算公式为：

$$U_i = \overline{X_i} + C\sqrt{\frac{2\ln N_p}{N_i}} \qquad (3-2)$$

在公式(3-2)中，$\overline{X_i}$是第 i 拉杆截至目前的平均收益值，表示已知收益的均值；N_i为第 i 拉杆被玩过的次数，N_p为所有的拉杆被玩过的总次数，C 为可以调整的加权系数，一般为$\sqrt{2}$，$\sqrt{\frac{2\ln N_p}{N_i}}$表示要探索的未知收益。

对于拉杆 i 来说，当它被玩的次数越少，$\overline{X_i}$的值就会越小，其他拉杆被玩的总数N_p自然就大，公式的$\sqrt{\frac{2\ln N_p}{N_i}}$值就会很大，在$\overline{X_i}$不变的情况下，此拉杆 i 被玩的可能性就会高；当拉杆 i 被玩的次数越多，假设它一直有很好的收益，$\overline{X_i}$在稳定增长，那么它可能一直被玩家玩下去；假设它的收益不理想，在N_i和N_p同步增长的情况下，对于函数的增长速率会明显低于常数，公式的$\sqrt{\frac{2\ln N_p}{N_i}}$值就会很小，当$\overline{X_i}$的值不变情况下，此拉杆就不会被玩家玩下去。

3.4 UCT 算法及其在藏式夹棋博弈中的应用研究

UCT(Upper Confidence Bound Apply to Tree，简称 UCT)算法或信心上界树算法是由匈牙利的列文特·科奇什(Levente Kocsis)与加拿大乔鲍·塞派什瓦里(Csaba Szepervári)在 2006 年合作提出的。该算法将蒙特卡洛树搜索与 UCB 策略结合，在庞大的博弈树搜索中具有于传统搜索算法的时间和空间优势[5]。利

用 MCTS 算法来解决藏式夹棋博弈问题，结果不太理想。由于 MCTS 算法在没有知识引导时博弈树搜索的深度很浅。因此将 UCB 策略添加到 MCTS 构造过程中形成了置信上限树算法(UCT)。

UCT 算法是一种完整的 MCTS 算法，它能从所以的落子点中选择最好的着法点，因为它将博弈树搜索过程中应用了 UCB 策略，而 UCB 策略有效地平衡探索和收益。在 2016 年和 2017 年由谷歌旗下 DeepMind 团队开发推出的 AlphoGo 和 AlphoGo Zero，将这种蒙特卡洛树搜索和深度学习结合应用到了围棋程序中，第一个击败人类职业围棋选手。这是应用此蒙特卡洛树搜索的一个非常成功的案例，也是计算机博弈发展史上有着里程碑的意义。

UCT 算法基本步骤如下：

```
算法：UCT 算法
    function UCT
1)    Node Present=Duild(当前根节点) 当前根节点建立第一层；
2)    Node MaxNode=MaxUCB(Present) 计算 UCB 值并选择 UCB 值最大的节点；
      If MaxNode 不是叶子节点
3)    返回 2）步；
      Else
          If MaxNode 的访问次数到达了预设；
4)            Node NewNode=Extend(MaxNode) 扩展节点；
              Int Visit_time= Visit_time+1 访问次数加 1；
          Else
5)            double Reward=Simulate_policy(MaxNode)  模拟走棋并计算收益值；
6)            Backup(Present, Reward) 回溯到根节点并一路更新收益值；
              If 尚未达到循环次数或预设时间
7)                由 1）开始重复执行；
              Else
8)                Node Best_Node(当前根节点) 获取 UCB 值最大的节点并结束算法；
```

下面将 UCT 算法将应用到藏式夹棋中的工作原理描述。 在对弈程序中棋局的初始状态为 S(0,0)，并且轮到计算机落子。在每一个节点中存储了两个属性：评估累计值 W 和在该节点或其下运行的访问次数 N。最初计算机可以从 S(0,1)、S(0,2)、S(0,3)、S(0,4)、S(0,5)节点中进行选择。 蒙特卡罗树搜索是由 S(0,0)的单个节点生成的树开始。 通过尝试每个动作并为每个动作构造相应的子节点来扩展该节点。 下面展示藏式夹棋的部分博弈树扩展：

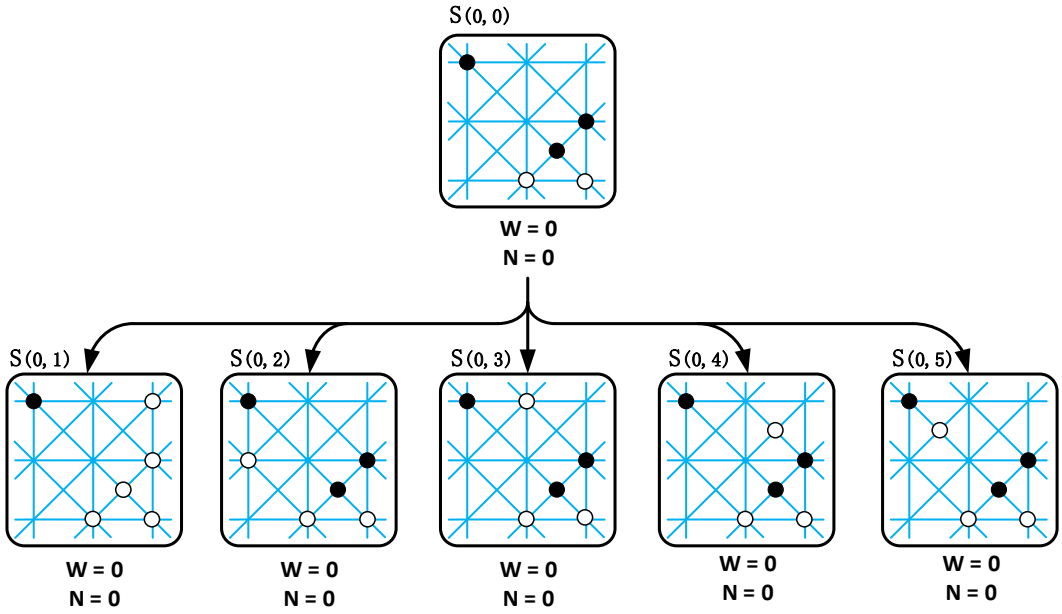


图 3-4 部分藏式夹棋博弈树

在选择过程选择在具有高利润率 、具有高估计值、和相对未开发、具有低访问次数之间取得平衡的节点。通过从根节点向下遍历树来选择叶节点，始终选择具有最高上限信心界（UCB）得节点 Si：

$$U_i = \frac{W_i}{N_i} + C\sqrt{\frac{2lnN_p}{N_i}} \qquad (3-3)$$

其中 W_i 是第 i 孩子的评估累积值，N_i 是第 i 孩子的访问次数，N_p 是父节点的访问次数。 参数 C≥0 控制选择收益丰厚的节点（低 C）和探索具有访问次数（高 C）低的节点之间的权衡。上述图 3-4 得知：

$$U_{S(0,1)} = U_{S(0,2)} = U_{S(0,3)} = U_{S(0,4)} = U_{S(0,5)} = \infty$$

在这种情况下，可以按顺序选择第一个节点 S(0, 1)。按蒙特卡洛树搜索的原理，选择完节点后需要判断当前的节点 S(0, 1)是不是叶子节点，在这里叶子节点是指节点是否被扩展过。很显然 S(0, 1)节点未被展开过，所以是叶子节点。接下来按照算法的步骤，需要判断节点 S(0, 1)被访问次数是否为 0。上述情况下，节点 S(0, 1)的访问次数为 0, 所以要进行下一步模拟走棋。

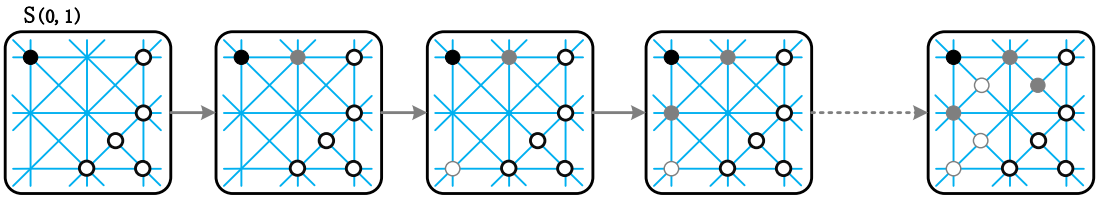


图 3-5 模拟走棋示意图

模拟走棋是指计算机按棋规则轮流随机下子，直到棋局胜利、失败或平局为止。模拟总会产生一个评估值，对一般的计算机博弈来说，胜利时评估值为 1，失败是评估值为-1，平局时评估值为 0。而藏式夹棋中开局阶段和移局阶段的棋规有所不同，因此在模拟时只模拟开局阶段。模拟完开局阶段后对局面进行一定的评估。开局阶段结束后棋盘中的己方棋子个数对随后的移局阶段的影响很大，所以用公式对模拟结果评估或给与奖励：

$$W_i = \frac{K}{N^2 + (N-1)^2 - 1} \qquad (3-4)$$

在公式(3-4)中 N 为棋盘的大小，K 为己方棋子的个数，W_i 为第 i 节点的模拟评估值。在5×5藏式夹棋的棋盘中有 41 个点，其中 40 个点可以落子。开局阶段棋盘的中心点或首府上不能落子，首府是移局阶段首次移动棋子的点，因此 N 一般为 5。在图 3-5 中第 S(0,1)节点的模拟评估值为$W_{S0,1} = \frac{8}{3^2+(3-1)^2-1} \approx 0.67$。

模拟评估完成后进入下一步回溯，根据模拟结果从原路返回并更新路过的所有节点的评估值。如图 3-6 所示，S(0,1)节点模拟评估完成后回溯的结果。这样就执行了一次 UCT 算法的四个步骤，完成了一次循环。但 UCT 算法的思想是就是从 S(0,0)开始不断的进行迭代，不断更新节点值，直到达到了一定的迭代次数或时间。

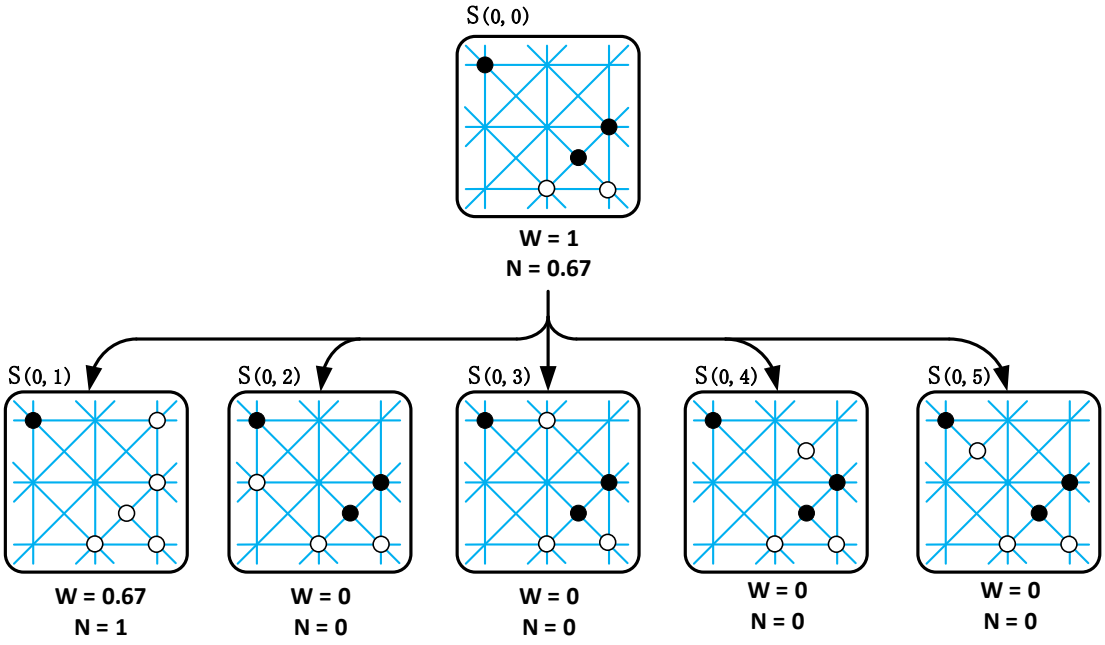


图 3-6 UCT 算法一次循环示意图

在第二次迭代时，在次用上限信心界（UCB）算法来选择节点。如图 3-6 得

知：

$$U_{S(0,1)} = 0.67 \qquad U_{S(0,2)} = U_{S(0,3)} = U_{S(0,4)} = U_{S(0,5)} = \infty$$

所以选择 S(0, 2)节点，同时判断当前的节点 S(0, 2)是不是叶子节点，在判断节点 S(0, 2)被访问次数是否为 0。跟第一次循环一样按顺序执行模拟步骤，在执行回溯步骤。以此类推用完循环次数或时间。

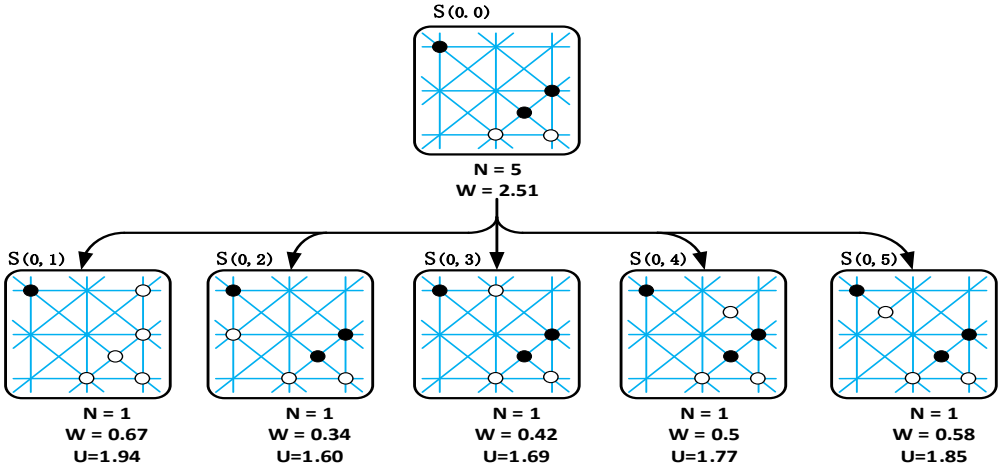


图 3-7 UCT 计算后的第一层示意图

在图 3-7 这种情况下，选择第一个节点 S(0, 1)。因为通过 UCB 公式计算得出：

$$U_{S(0,1)} = 1.94 > U_{S(0,5)} = 1.85 > U_{S(0,4)} = 1.77 > U_{S(0,3)} = 1.69 > U_{S(0,2)} = 1.60$$

在所有节点中 UBC 值最大的是 S0, 1 节点，因此选择该节点。之后按 UCT 算法的步骤进行计算，计算后将所有值会向上传播到父级节点(如图 3-8)。

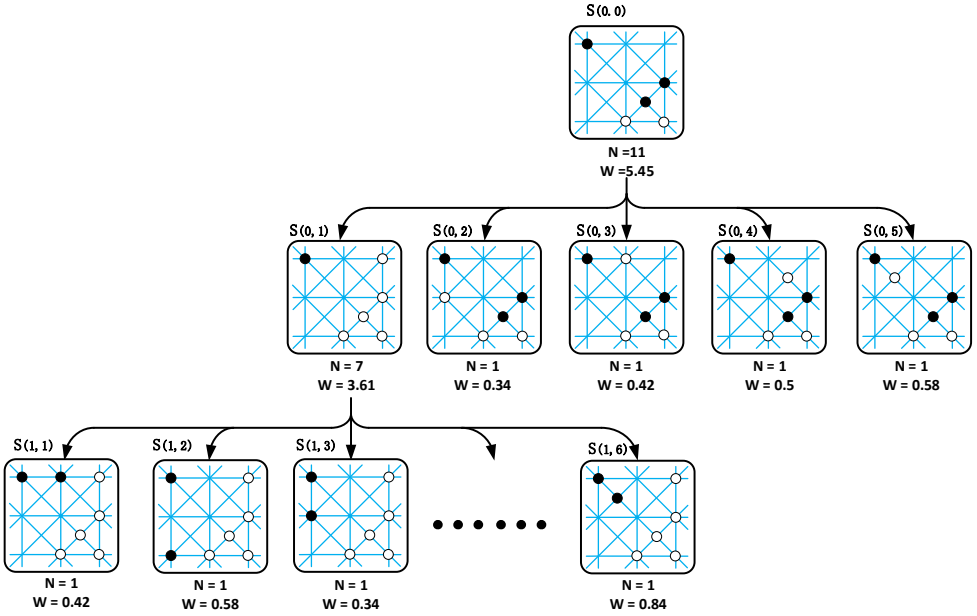


图 3-8 UCT 算法扩展示意图

继续运行蒙特卡罗树搜索的迭代，直到用完时间或者迭代次数达到了预设的次数。 博弈树逐渐扩大，（希望）探索可能的节点，确定最佳节点。 然后计算机通过选择访问次数最多的第一个孩子实际上在原始的真实游戏中落子。例如，假设最终博弈树的一层为图 3-9：

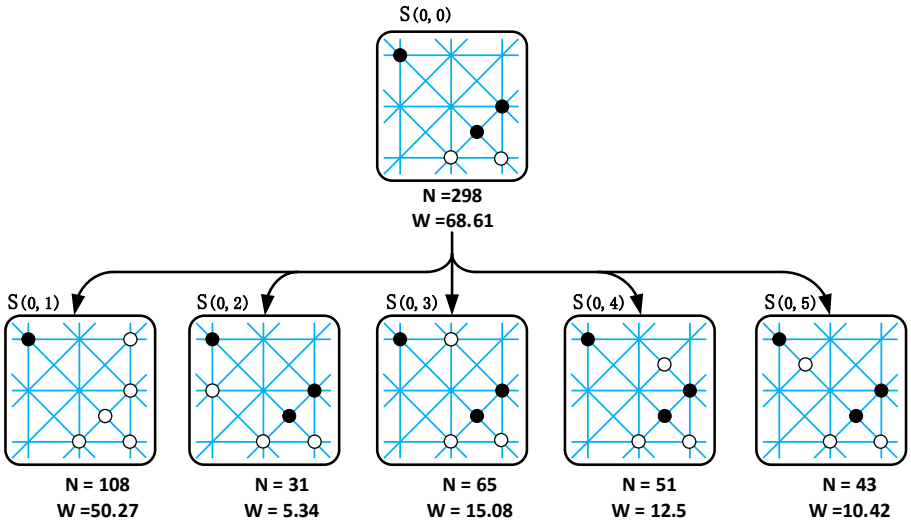


图 3-9 UCT 算法生成的博弈树第一层示意图

最后通过 UCB 公式计算每个节点的估值，选择评估值最高的点做真实游戏中落子的位置。如图 3-9 所示，选择 S(0,1)做最终的落子点。UCT 算法对 S(0,1)节点的估值如下：

$$U_{S(0,1)} = \frac{50.27}{108} + \sqrt{2}\sqrt{\frac{2\ln 298}{108}} = 0.70$$

以上是将蒙特卡洛树搜索结合 UCB 策略的 UCT 算法应用在藏式夹棋上的具体思想。算法中搜索的循环次数是固定有限的，因此，对**叶子节点的访问次数非常有限**，在同一层中不同的节点访问次数和节点的顺序是密切相关的。其次，UCT 算法是**脱离藏式夹棋知识**，只需结合 UCB 策略反复进行模拟评估就能得最终的落子点。从而**最终的落子点不理想**，因为它脱离了藏式夹棋的离线知识。

3.5 结合静态评估的 UCT 优化

在藏式夹棋计算机博弈中搜索是获胜的关键因素之一，因为搜索的过程和结果影响着最终的落子点，从而决定着藏式夹棋计算机引擎的输赢。但在 UCT 算法对藏式夹棋博弈树搜索的过程中，主要是用 UCB 公式来计算每个点的评估值，并

以此来决定搜索的方向，也就是搜索算法在所有的可选落子点中随机选择一个点计算 UCB 值，这样计算完所有可选点之后才做出比较选择最大的点。而没有一个有效的选择策略，从而博弈树庞大的藏式夹棋中该算法缺乏搜索效率和已选点准确性不高。由于这样的原因将藏式夹棋的一些离线知识加入 UCT 算法中，将 UCT 算法做一些优化，进一步提高搜索效率和已选点的准确性。

3.5.1 计算机藏式夹棋的静态评估方法

藏式夹棋是一种较复杂的棋类，直接从整体上进行判断或评估棋盘局面，其计算复杂性高而无法准确的判断。因此，分块评估在组合是非常重要的一步。在这里中按棋规来各个规则分块评估在组合；在进行评估之前，对棋规则做一个分块是十分重要的，要得出整个盘面的评估值，必须将各个模块被充分评估后，各个模块的评估结果综合起来。在藏式夹棋的局面评估中计算某一个点对当前局面起到进攻影响和威胁影响极为重要。因此计算藏式夹棋的局面评估值时，首先，计算某个点对当前局面的进攻值，在计算对当前局面的威胁值。最后，将两种值进行合并得出了藏式夹棋的静态局面评估值。

(1)进攻值

在藏式夹棋中进攻方法主要是夹法和打枪法两种。首先计算夹法的估值，在计算打枪法的估值。最后把两个估值综合起来，确定为最终的进攻值。夹法规则的估值计算公式如下：

$$exp_J = \sum_{i=0}^{n} P_i \qquad (3-5)$$

exp_J为夹法规则的进攻值，i 为被夹的棋子，n 为被夹的总数。P_i为棋子 i 的权重。在藏式夹棋中每个棋子的作用都差不多，因此在这里每个棋子的权重定义为 1。

打枪规则的估值计算公式如下：

$$exp_Q = \sum_{j=0}^{m} P_j \qquad (3-6)$$

exp_Q为打枪规则的进攻值，j 为已打枪的棋子，m 为被打枪的总数。P_j为棋

子 j 的权重（棋子的权重定义为 1）。

按以上的夹法规则的进攻值和打枪规则的进攻值综合为藏式夹棋的进攻值。藏式夹棋的进攻值计算公式为如下：

$$\exp_{JQ} = \sum_{i=0}^{n} P_i + \sum_{j=0}^{m} P_j \qquad P_i = 1, P_j = 1 \qquad (3-7)$$

(2)威胁值

所谓的威胁值是在下一步轮到对方落子时，对方的棋子把己方棋子吃掉的概率。也就是己方棋子受到对方棋子威胁的估值，也称对方对己方的威胁程度。在藏式夹棋中对方对己方的威胁方式也是夹法和打枪法两种。对方用夹法来威胁的估值计算公式如下：

$$axp_J = \sum_{i=0}^{n} P_i \qquad (3-8)$$

axp_J为对方用夹法规则来威胁的估值，i 为对方夹己方的棋子，n 为被夹的总数。P_i为棋子 i 的权重。评估函数是用来计算当前局面对己方有利的估值，因此，计算威胁值时对方的每个棋子的权重设为-1。

对方用打枪规则威胁的估值计算公式如下：

$$axp_Q = \sum_{j=0}^{m} P_j \qquad (3-9)$$

axp_Q为对方用打枪规则威胁的估值，j 为对方已打枪法威胁的己方棋子，m 为被打枪的总数。P_j为棋子 j 的权重（棋子的权重定义为-1）。

按以上对方用夹法规则的威胁值和打枪规则的威胁值综合为藏式夹棋的威胁值。藏式夹棋的威胁值计算公式为如下：

$$\mathrm{axp}_{JQ} = \sum_{i=0}^{n} P_i + \sum_{j=0}^{m} P_j \qquad P_i = -1, P_j = -1 \qquad (3-10)$$

按以上谈论，对藏式夹棋的静态局面估值方法用了分块组合的方法。首先计算了当前局面的进攻值，在计算当前局面对己方的威胁值。最后将进攻值和威胁值的综合后确定为当前局面的评估值。藏式夹棋的静态局面评估函数为如下：

$$\mathrm{Eval(board)} = exp_{JQ} + axp_{JQ} \qquad (3-11)$$

exp_{JQ}为当前局面的进攻值，axp_{JQ}为当前局面的威胁值。

3.5.2 藏式夹棋博弈中对 UCT 进行优化

虽然利用 UCT 算法可以解决藏式夹棋博弈问题，但是由于 UTC 算法没有藏式夹棋离线知识的指导，搜索时博弈树的扩展层数较少。因此，想到将藏式夹棋的静态评估方法加入到 UCT 算法当中。虽然静态评估方法对藏式夹棋的知识概况的不全面，但加入到 UTC 算法可以提高算法的搜索效率，还能起到剪枝的效果。

应用 UCB 策略和蒙特卡洛树搜索的 UCT 算法包括四个步骤：选择节点、扩展节点、棋局模拟、回溯更新。在主要优化的步骤是扩展节点方面，在博弈树中加入离线的知识控制节点的扩展。当选择的节点到扩展的时候，最理想的结果是花最短的时间寻找到最好的节点。因为在节点展开的过程中，时间是非常有限的资源，以相等的权重一次性扩展节点的所有子节点是非常浪费时间和空间资源，而且这种浪费往往以指数的形式增长。并且在有限的时间内采用 UCT 算法搜索时，叶子节点的访问次数也非常有限，从而导致了在节点的排序顺序严重地影响着同一层中不同节点的访问次数不一样。同一层的节点中排在后面的节点比前面的节点访问机会少，甚至访问不到。如果有个相对好的节点排在后面有可能被访问不到，而时间浪费在无用的节点上，从而导致了计算机藏式夹棋的棋力下降。因此，在节点扩展子之前，如果能有效利用棋谱知识对所有的节点进行排序，即把胜率大的节点排在前面，这样就会使得计算机藏式夹棋引擎能够充分发挥计算效率，最终能搜索到好的节点，达到事半功倍的效果。

静态评估的优点是能够帮助计算机博弈引擎判断当前局面的效率，但是由于藏式夹棋中棋局的形势特别复杂，针对每个形势的方法也不一样，有些局面形势连职业棋手也很难对其进行准确的总结，因此不可能采用一套知识来概括所有的局势，最重要的是当知识概括的过于复杂时，计算机处理的时间又会更长，从而对开发人员的藏式夹棋知识要求很高。基于以上原因，静态局面评估也有一定的局限性。考虑以上问题，最终将静态局面评估和 UCT 算法结合的方法来判断棋局形势和扩展胜率大的节点。

针对以上问题，根据藏式夹棋的下棋特点，提出了引入经验知识对 UCT 算法

进行优化。

（1）在藏式夹棋计算机博弈中，棋盘上棋子的个数对棋力的影响非常大，因此在所有可选的落子点中，能够夹击或长枪吃掉对方棋子数量多的落子点进行加分处理。如图 3-10 所示，白方进行模拟扩展时，可选的点很多。假设白方选择落在 A 点，能够吃掉标记为 3 号和 9 号、1 号等棋子；如果选择落在 B 点，能够吃掉标记为 5 号和 7 号的棋子；经过比较之后，很明显 A 点能吃掉对方三枚棋子，B 点能吃掉对方两枚棋子。所以所有可选点中最优的选择是 A 点，应对 A 需要优先考虑，A 点权重最高，其次是 B 点。

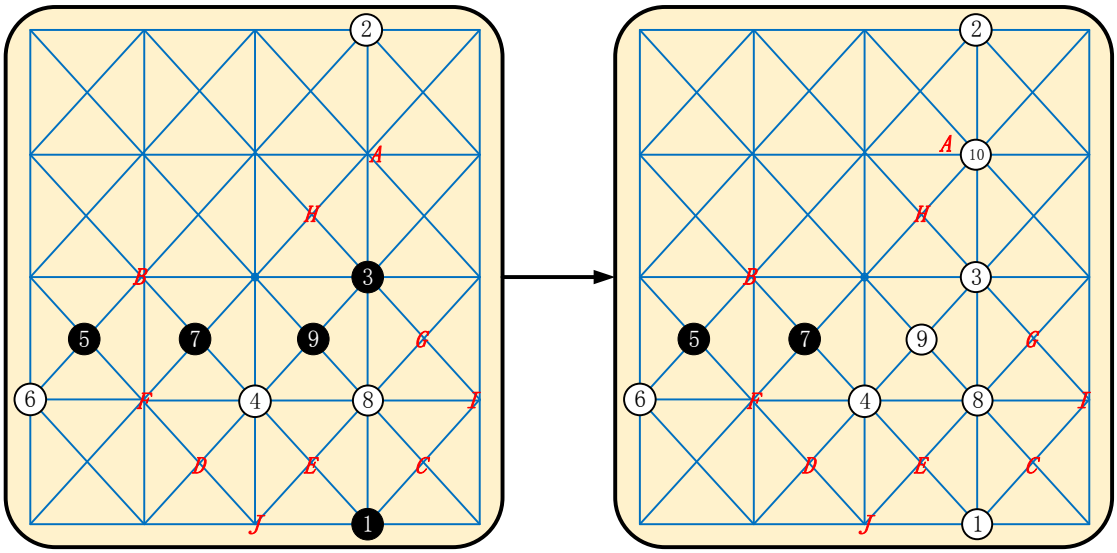


图 3-10 藏式夹棋棋盘局面

（2）在藏式夹棋中，有时己方棋子受到对方棋子的限制，在下一步对方行棋时面临着被吃掉的危险，这样的棋子可以通过防御的形式将对方的路堵住，或者将对方有威胁的棋子吃掉。如果能吃掉对方有威胁的棋子，这种办法即有进攻又有防御的效果，但一般情况中这种形势较少。如图 3-10 所示，当轮到对方落子的时候，标记为 4 号和 8 号的棋子面临着被吃掉的危险。所以己方要保护这两枚棋子。在这里最好的保护方法是己方在 A 点落子吃掉对方有威胁的棋子 9 号和 3 号，这样既保护了 4 号和 8 号棋子而且进攻了对方的 3 号和 9 号棋子。另外还可以在 D 点或 E 点、C 点落子保护其中一个棋子，但这样效果不是很好。经过分析得知，A 点有进攻的效果也有防御的效果，D、E、C 点只能起到防御的效果。因此 A 点的权重又比 D、E、C 点高。

（3）在藏式夹棋中一枚棋子与它周围棋子之间相互的影响力非常大，如果

落子点的周围对方棋子多，那么对该棋子的威胁较大的。因此落子时如果不能吃掉对方，一般不能把己方棋子落在对方棋子周围。如果己方棋子落在对方棋子周围而没能吃掉对方棋子，那么下一步轮到对方行棋时该棋子有被吃掉的危险。如图 3-10 所示，不能在 C、F、G、J、I、H 点落子，这些点有被对方棋子吃掉的危险。因此对 C、F、G、J、I、H 等点的权重设小一点。

（4）将所有的棋子按权重的大小进行快速排序，权重高的节点自然排在前面，更容易被搜索到，提高了 UCT 算法的搜索效率。

优化 UCT 算法基本步骤：

```
算法：优化 UCT 算法
    function UCT-Majorization
1)    Node Present=Duild(当前根节点) 当前根节点建立第一层；
2)    Node StaticNode=StaticEvaluate(Present) 用静态评估方法来评估
3)    StaticNode=RankNode(StaticNode) 按评估结果对节点排序
4)    Node MaxNode=Max- StaticEvaluate (StaticNode) 选择静态评估值最大的节点；
      If MaxNode 不是叶子节点
5)      返回 2）步；
      Else
        If MaxNode 的访问次数到达了预设；
6)        Node NewNode=Extend(MaxNode) 扩展节点；
          Int Visit_time= Visit_time+1 访问次数加 1；
        Else
7)        double Reward=Simulate_policy(MaxNode)  模拟走棋并计算收益值；
8)        Backup(Present, Reward) 回溯到根节点并一路更新收益值；
          If 尚未达到循环次数或预设时间
9)          由 1）开始重复执行；
          Else
10)           Node Best_Node(当前根节点) 获取 UCB 值最大的节点并结束算法；
```

用以上思想将 UCT 算法优化后，不像以前那样将所有的可选节点直接展开后进行计算 UCB 值并选择节点，而是先根据藏式夹棋的离线知识对所有的可选节点进行静态评估，然后按照静态评估的数值进行排序，再依据排序的结果对节点进行展开，从而提高有胜率大的节点被提前访问到的可能性。这样在有限的时间内尽可能访问到所有好的节点，间接性地提高计算机搜索博弈树的效率，比未做优化的算法更有价值。比如，用静态评估方法对所有可选节点进行排序后，没有进行绝对的剪枝策略，如果有足够的时间时，能够进行较多的搜索，此时可能有机会访问那些排在后面的不够优秀的节点。但用静态局面评估方法将所有可选点进

行快速排序，不像以往那样所有的可选节点公平对待，而是先对好的节点给予较多的机会，在对那些较差的节点给适当的机会。从而实现更好地在探索和利用之间找到平衡点来维持效率和效果的和谐。

3.6 本章小结

本章首先研究了传统的蒙特卡洛方法、蒙特卡洛评估、蒙特卡洛树搜索算法、UCB 策略；在研究了 UCB 策略和蒙特卡洛树搜索结合的 UCT 算法的原理和思想，并结合藏式夹棋分析了各自的优缺点；最后针对藏式夹棋的特点将 UCT 算法的选择策略和扩展策略进行了优化，提出了静态评估与 UCT 算法结合的优化算法；总体上该算法的搜索效率等在第五章进行实验证明。

第4章 基于 MCTS 藏式夹棋博弈系统的设计与实现

藏式夹棋下棋简单、方便，但是棋局复杂多变，有着极其复杂的理论分析和数学计算思维。在藏式夹棋人机博弈系统的研究中，除了基本的计算机博弈理论知识的完善还有程序设计的挑战，为了达到程序的方便和灵活可操作的目标，就必须要全方位的考虑以保证软件的可维护和升级。

4.1藏式夹棋博弈系统总体设计

“Tibetan GO”藏式夹棋人机博弈系统分为客户端和通信模块、博弈引擎三部分，客户端主要负责与用户交互操作，实现对博弈系统的控制。通信模块主要负责客户端与博弈引擎之间的通信功能。博弈引擎模块是藏式夹棋人机博弈系统的核心模块，主要负责博弈树的搜索以及选择最佳落子点。系统总体结构设计图如图 4-1 所示。

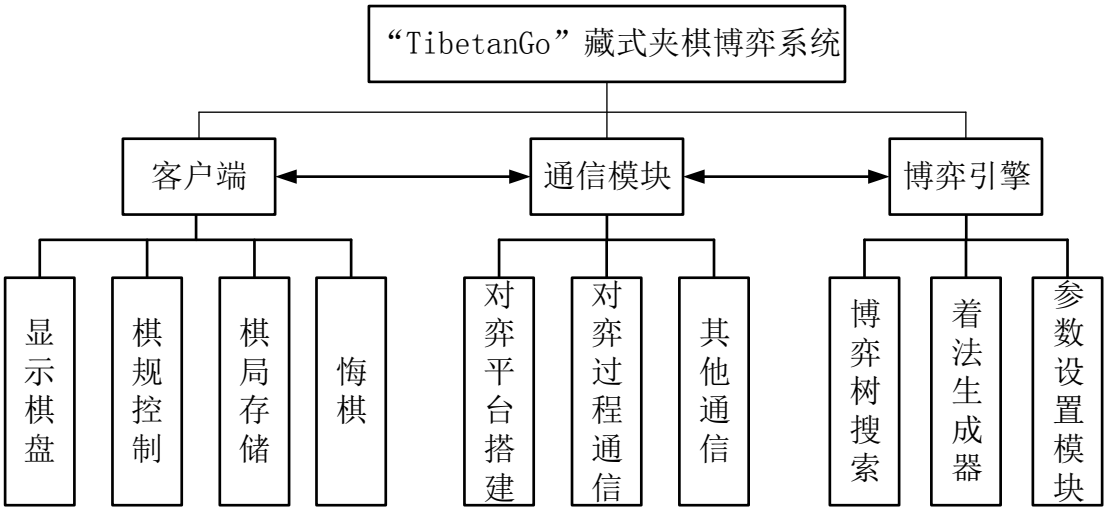


图 4-1“Tibetan GO”藏式夹棋人机博弈系统总体结构图

计算机藏式夹棋人机博弈系统是指计算机能够像人一样下藏式夹，进行思维、选择并做出决策的人工智能系统。其中，客户端主要包括显示棋盘、棋规控制、棋局保存、悔棋等；通信模块主要包括对弈平台搭建、对弈过程通信、其他通信等工作；博弈引擎模块主要包括博弈树搜索、着法生成器、参数设置模块等。整个人机博弈系统而言，博弈引擎模块是整个系统的核心模块，它是决定博弈胜负

的关键部分。

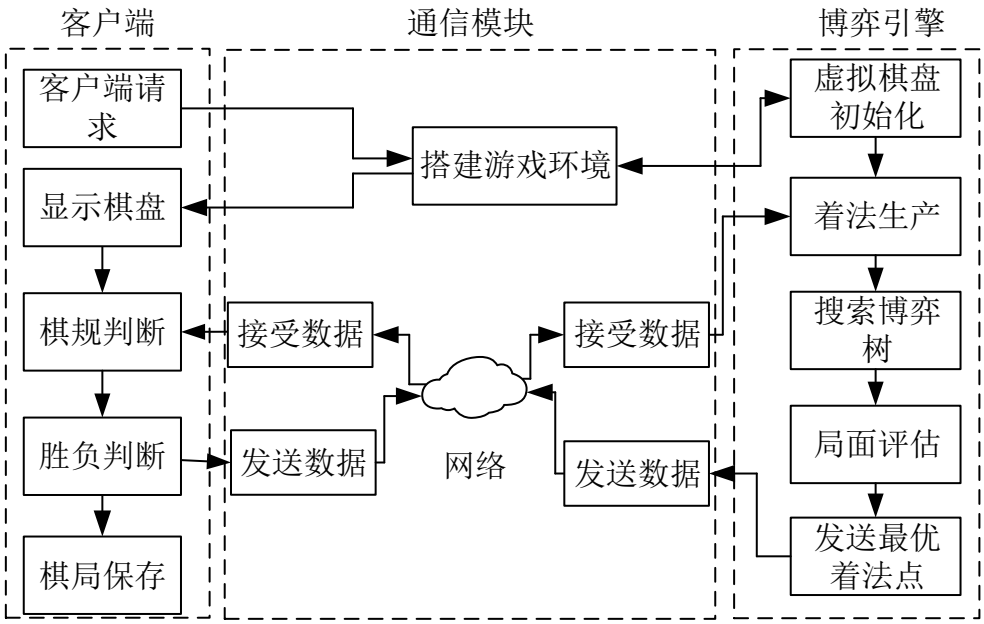


图 4-2 藏式夹棋人机博弈系统流程图

藏式夹棋人机博弈系统的流程如图 4-2 所示。系统首先根据用户的请求搭建对弈平台并虚拟棋盘进行初始化，然后对用户显示棋盘准备对弈。用户选好棋子颜色后按棋规在棋盘中落子，不管得到了用户的棋步之后或博弈引擎产生棋步之前，都必须要经过棋规判断模块进行检查合法性。落子跟棋规不合法则进行相应的报错处理，如果合法就通过通信模块发送到博弈引擎并更新模拟棋盘数据。然后通过着法生成器来生成所有的可落子点，之后用蒙特卡洛树搜索算法来搜索和选择最佳的落子点并通过通信模块发送到客户端。在客户端接受到计算机落子位置后通过棋规判断器进行检查，之后进行判断胜负。如果胜负结果明确了就存储所有的棋局并结果对弈。

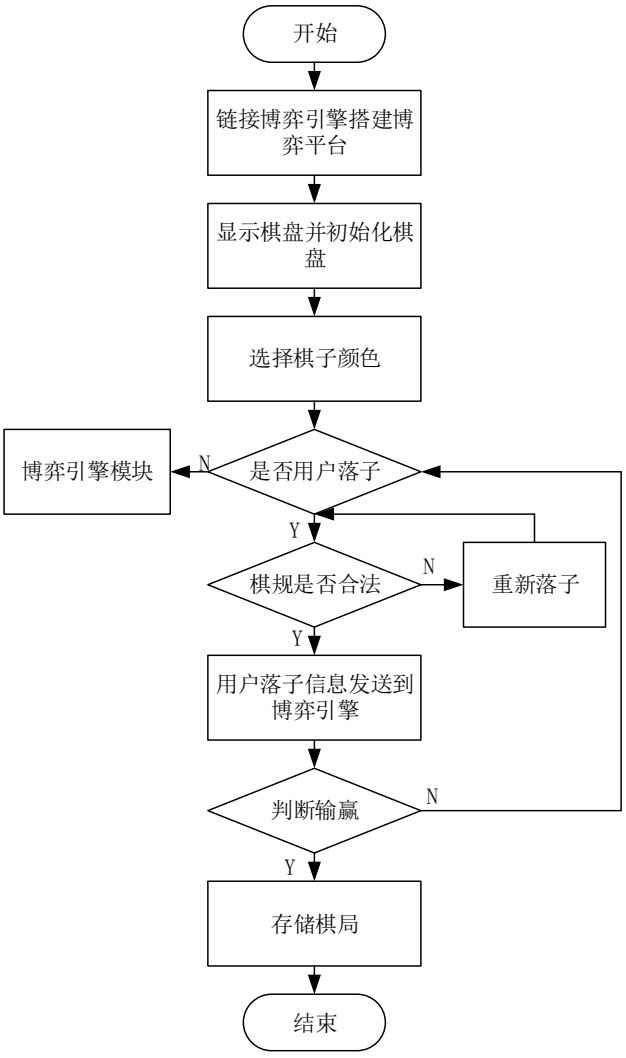
4.2 藏式夹棋博弈客户端设计

4.2.1 博弈客户端总体设计

博弈客户端也就是系统用户界面，主要是显示棋盘和棋子，控制用户下棋规则是否合法。棋规控制中主要有控制落子、控制移动、控制让路、判断胜负等功能。在博弈客户端模块中的显示棋盘和棋子的任务是与机器和用户交互，控制棋规模块的任务是控制用户按实际藏式夹棋的规则下棋。具体的实现在下面几节进

行详细介绍。

藏式夹棋人机博弈系统的客户端模块的总体流程如图 4-3 所示，首先搭建好博弈平台后系统会向用户显示棋盘并虚拟棋盘初始化，在客户端界面中用户选择好棋子颜色后进入对弈状态中。进入对弈状态后系统开始检测鼠标点击事件，如果轮到了用户落子，检测用户在棋盘中点击位置来计算用户在棋盘中落子的位置。得到了用户落子的位置后系统检查用户落子的是否合法，倘若不合法将重新落子，若合法在相应的位置显示棋子并落子位置将发送到博弈引擎模块。最后判断是否赢棋，若赢棋将整个对弈过程保存并对弈结束。



4-3 客户端总体流程图

4.2.2 落子模块设计与实现

藏式夹棋的布局阶段主要的行棋规则是落子，除了首府或棋路以外所有的点落子完成后说明布局阶段结束。落子控制模块的流程如图 4-4 所示。首先判断是否轮到用户落子，如果轮到用户落子则在棋盘中想落子的位置进行点击落子。点击后将获取点击位置并判断该位置是否在棋盘中，然后判断是否满盘。若还没满盘则继续判断轮到用户落子后是否第一次落子。倘若是第一次落子则判断是否落在首府或棋路上，因为首府或棋路上不能落子。若该点不在首府点上才进行真正的落子。完成了上面所有的判断则在虚拟棋盘中落子，然后在用户界面的棋盘中显示棋子并计算该棋子能吃的所有棋子保存在吃子集合中。最后将该落子点的位置发送在博弈引擎模块。

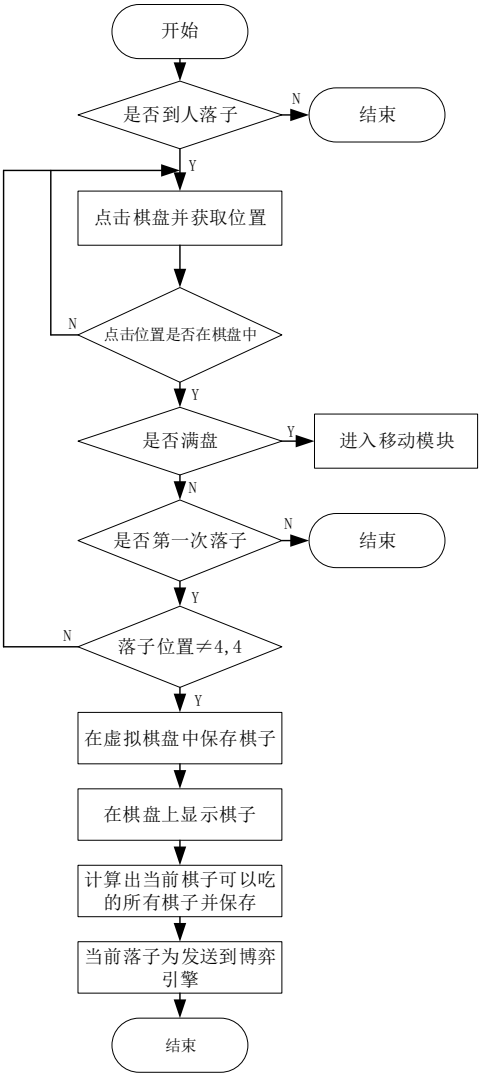


图 4-4 落子控制模块流程图

4.2.3 吃子模块设计与实现

在藏式夹棋中吃子是唯一的进攻方式，所谓的吃子是把对方的棋子颜色改成己方的棋子颜色，它贯穿在藏式夹棋的每个阶段。当一方吃掉对方所有的棋子后表示赢棋。当一方落子或移子后可以通过藏式夹棋的规则进行吃子，吃子控制模块的流程如图 4-5 所示。首先用户在能吃的棋子上右键点击，然后判断当前点击的棋子颜色和下棋方的颜色是否一致。若一致则判断吃子是否合法。吃子是否合法的主要依据是在落子模块或移动模块、让路模块，吃子模块中计算好能吃的棋子并保存好的吃子集合。以上判断正确后在将虚拟棋盘和用户界面棋盘中的棋子颜色更改，然后将已更改的颜色的棋子位置和颜色发送到博弈引擎模块。最后将计算已更改颜色的棋子能吃的所有棋子保存在吃子集合中。

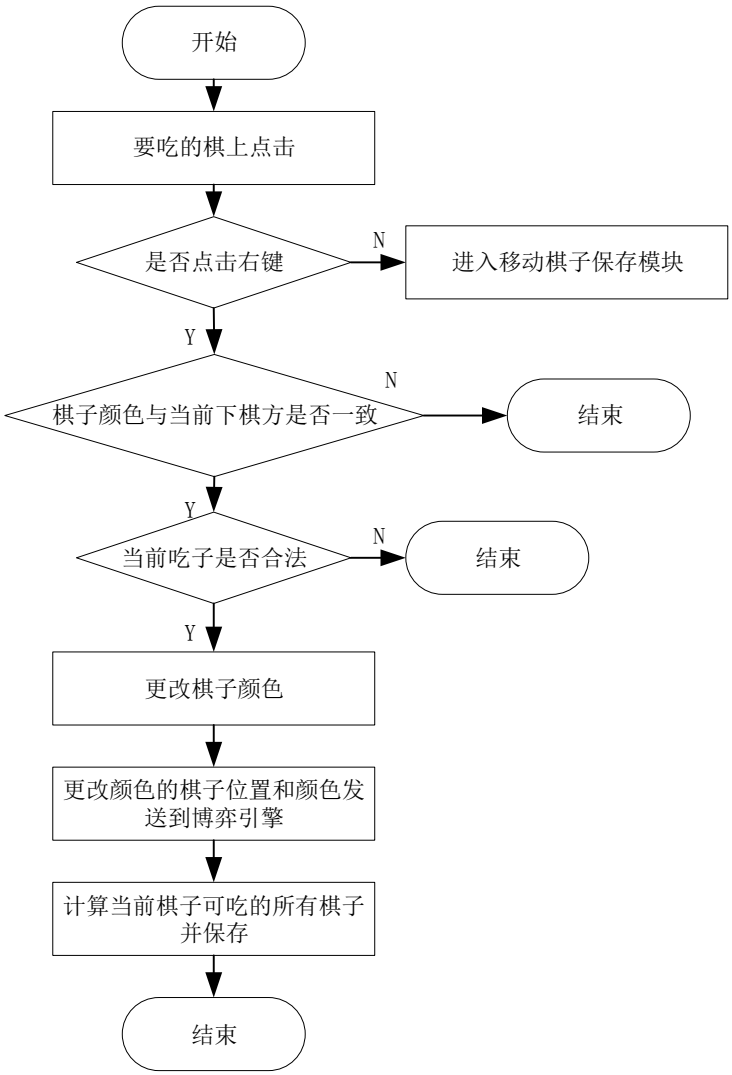


图 4-5 吃子控制模块流程图

4.2.4 移动模块设计与实现

在藏式夹棋中移局阶段的主要行棋规则是移动棋子，当落子满盘后进入了移局阶段。移动模块又分两个小模块，分别是移动棋子保存模块和移动棋子模块。移动棋子保存模块是对移动棋子准备。移动棋子保存模块流程如图 4-6 所示。首先要移动棋子上进行左键点击并获取棋子位置信息，然后判断该棋子是白棋还是黑棋，如果是白棋则判断当前是否轮到黑方移动，若是黑棋则判断当前是否轮到黑方移动。最后将当前棋子的位置和颜色保存准备移动。

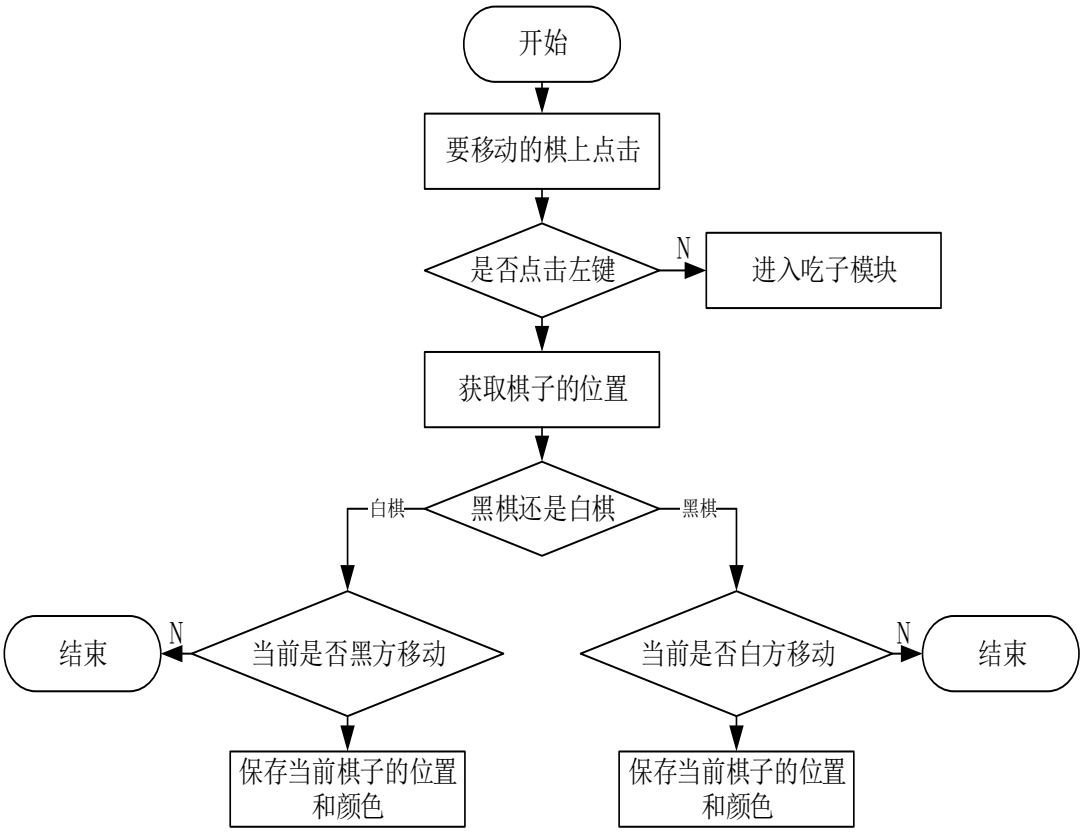


图 4-6 移动棋子保存流程图

在移动棋子保存模块将要移动的棋子保存完成后才能进行移动棋子。移动棋子保存模块负责选择要移动的棋子，而移动棋子模块主要确定棋子移动的目标位置。移动棋子控制模块的流程如图 4-7 所示。首先跟落子模块一样判断是否轮到用户行棋，倘若轮到了用户行棋则在棋盘中将要移动的位置点击并获取位置信息。然后判断当前位置是否为空，若空则继续判断是否有将要准备移动的棋子，在判断当前移动是否普通移动。如果是普通移动则判断当前移动是否合法，若合法才能在虚拟棋盘中进行移动棋子，然后在用户界面棋盘中移动棋子。移动完成后计

算当前移动的棋子能吃的所有棋子并保存在吃子集合，最后将移动的棋子位置和颜色并目标位置发送到博弈引擎模块。

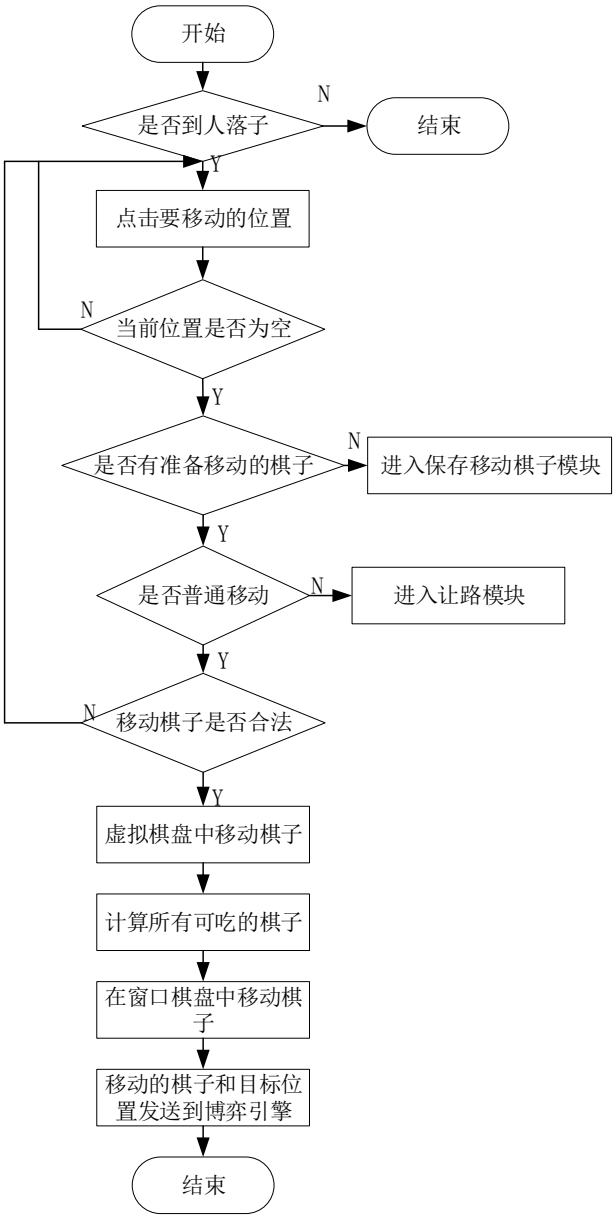


图 4-7 移动控制模块流程图

4.2.5 让路模块设计与实现

当满盘进入移局阶段时棋盘上只有一个空位置可以移子。但有时空位的周围没有己方棋子而无法移动棋子，这时对方要己方让路。同样有时己方挡住了对方移动的路线，这时己方给对方让移动的路线。让路的方法有移动让路和跳棋让路，但首先一定判断是否能移动让路，如果不能移动让路才能跳棋让路。

因让路模块是移动模块的特殊情况，因此让路前一定要将让路的棋子保存好，保存流程和方法跟移动模块的移动棋子保存方法一样。让路控制模块流程如图 4-8 所示。要让路的棋子保存完成后，在棋盘中要让路的目标位置点击并获取位置信息。然后判断当前位置是否为空，若是空位则继续判断是否能移动让路。如果能移动让路则判断是否能一步让路，然后在检查当前移动是否合法。完成之后才能在虚拟棋盘和用户界面棋盘中让路移动。最后将让路棋子的信息和目标位置信息等发送到博弈引擎模块并计算当前让路棋子能吃的所有棋子保存在吃子集合。如果不能移动让路则进行跳棋让路，这时不用检查移动是否合法直接跳棋让路，其他的步骤跟移动让路一样。

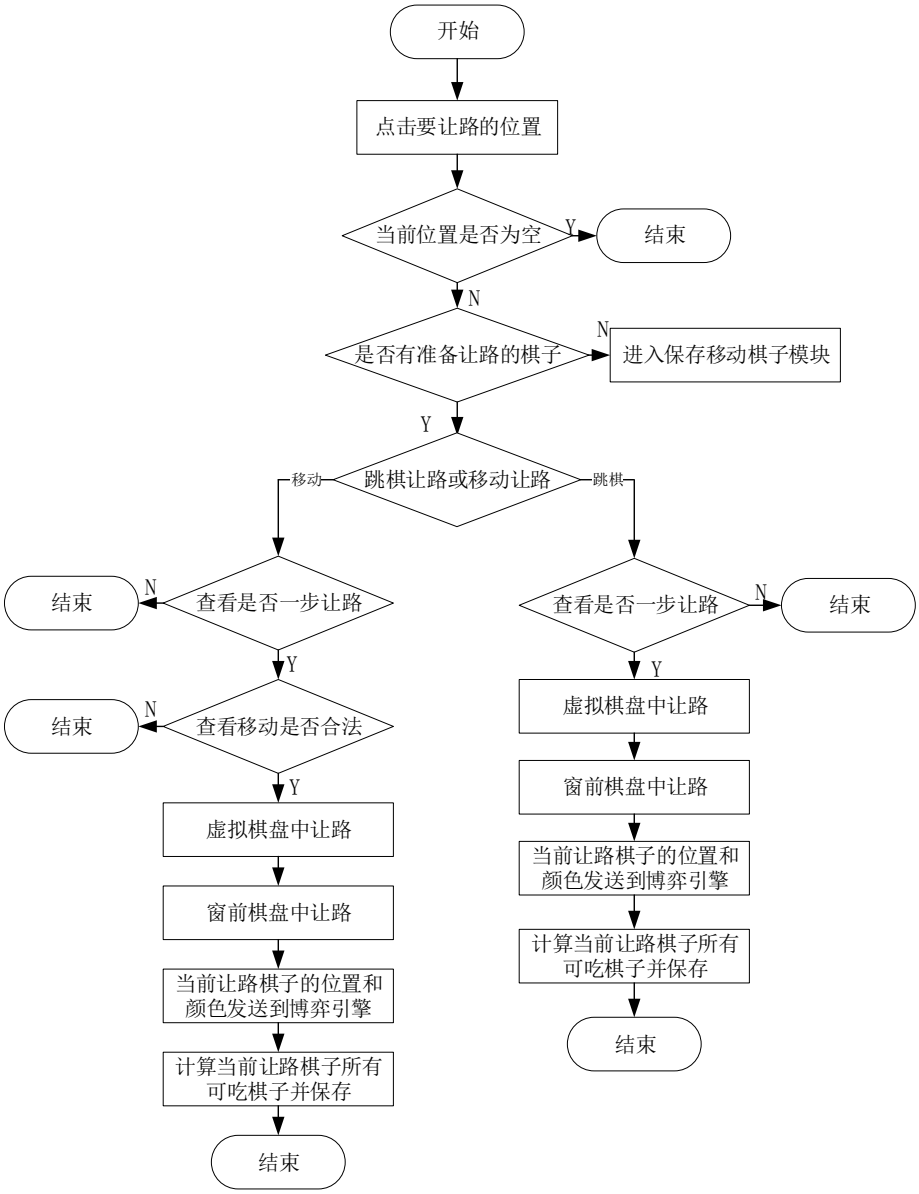


图 4-8 让路控制模块流程图

4.3 藏式夹棋博弈引擎设计与实现

4.3.1 博弈引擎总体设计

博弈引擎模块是实现如何让计算机下藏式夹棋的核心模块，是整个计算机藏式夹棋人机博弈系统的大脑。博弈引擎通过通信模块与藏式夹棋客户端界面软件通讯。博弈引擎模块的关键技术主要包括博弈树搜索相关的算法和着法生成器。博弈树搜索方法是前文提到了的蒙特卡洛树搜索算法结合静态评估的方法。着法生成器有三种，分别是落子控制模块的着法生成器和移动棋子模块的着法生成器、让路控制模块的着法生成器。随着藏式夹棋进入了不同阶段，着法生成器也不同。但每一个阶段博弈树搜索算法是一样的。计算机藏式夹棋人机博弈系统的博弈引擎模块总体流程如图 4-9 所示。

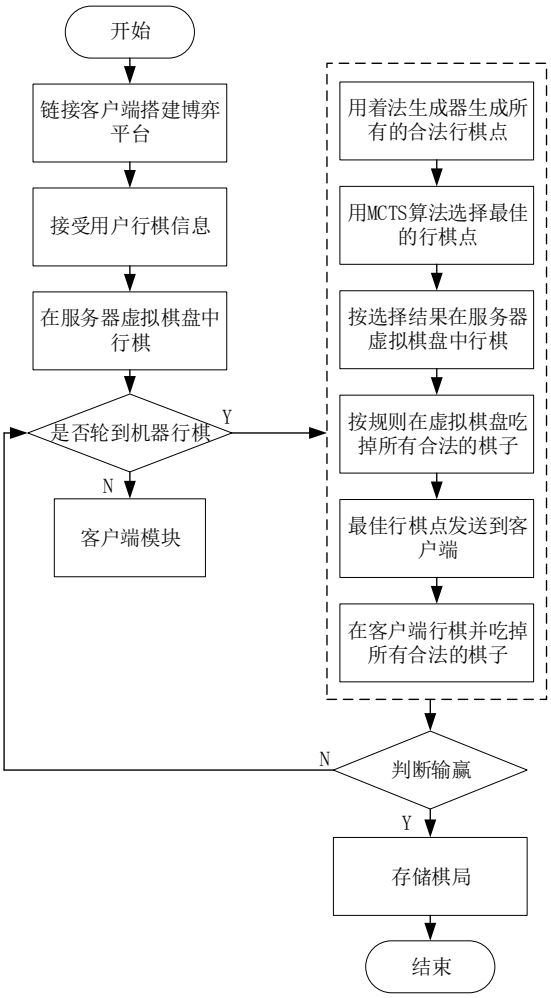


图 4-9 博弈引擎总体流程图

4.3.2 MCTS 算法模块设计

MCTS 算法模块是研究的核心部分。根据藏式夹棋的特点，应用了一种静态局面评估方法和 UCT 算法结合的 MCTS 算法。该算法的原理以及步骤在第三章已详细介绍，图 4-10 为该算法的具体流程图。

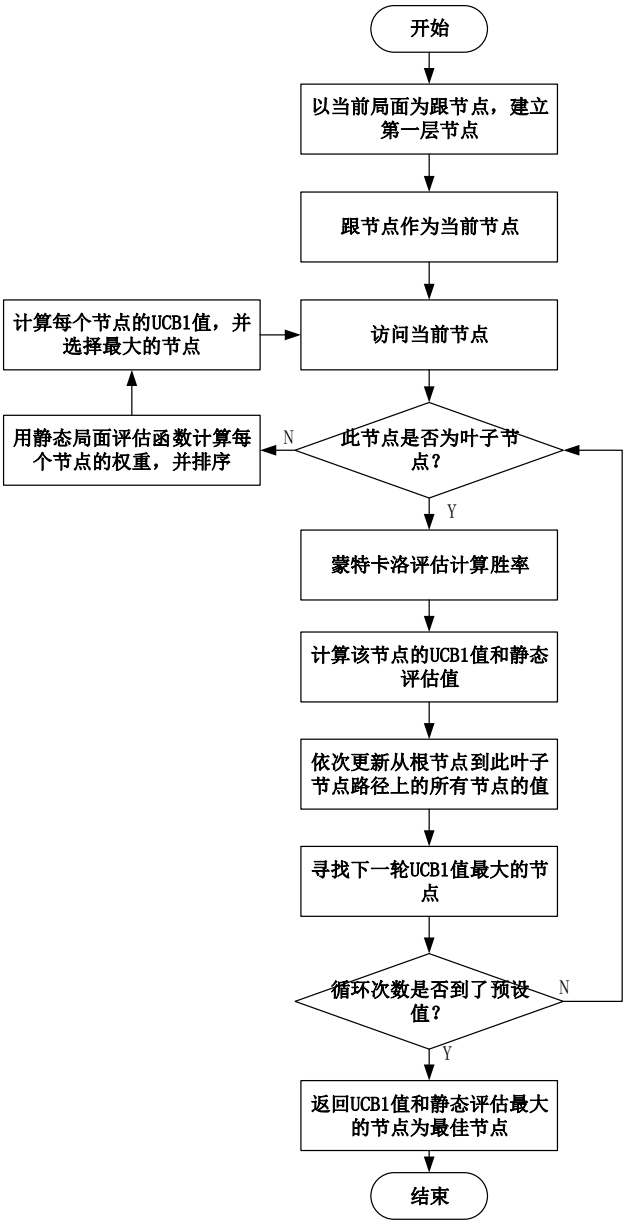


图 4-10 MCTS 算法流程图

4.3.3 机器落子控制模块

机器落子模块是主要负责在开局阶段计算机落子时选择最佳的落子点。本

模块是机器获胜的关键部分，落子的好坏直接影响到往后的移局阶段。机器落子控制模块主要有特定的着法生产器和博弈树搜索算法。本模块的着法生成器基本思路是在棋盘中扫描，找出所有的空位置。因此本模块博弈树较庞大，搜索算法运行时消耗的时间较长。机器控制模块的流程如图 4-11 所示。

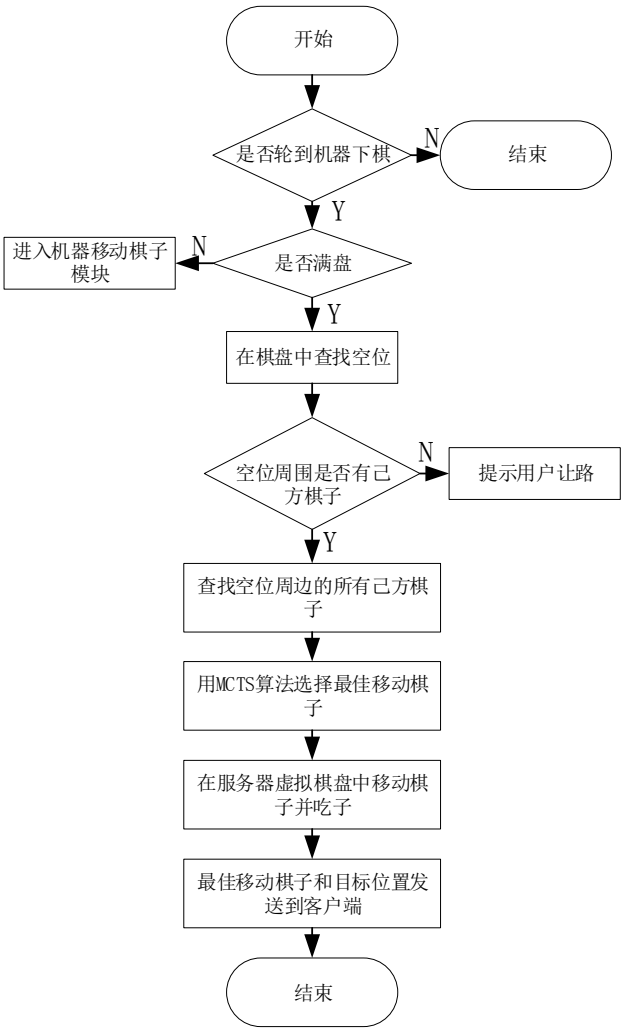


图 4-11 机器落子控制模块流程图

4.4.4 机器移动棋子控制模块

机器移动棋子控制模块是主要负责移局阶段计算机在所有能移动的棋子中选择最佳的棋子。移局阶段全盘中只有一个点可以移动棋子，因此计算机在移动棋子前首先要找的移动点，并且查找在移动点周围是否有可以移动的棋子，上述这个流程是机器移动棋子模块的着法生成器原理。机器移动棋子控制模块中因移动点只有一个，所以能移动的棋子也不多。从而博弈树的规模也小，在

搜索算法选择最佳的移动棋子时消耗的时间也少。机器移动棋子控制模块流程如图 4-12 所示。

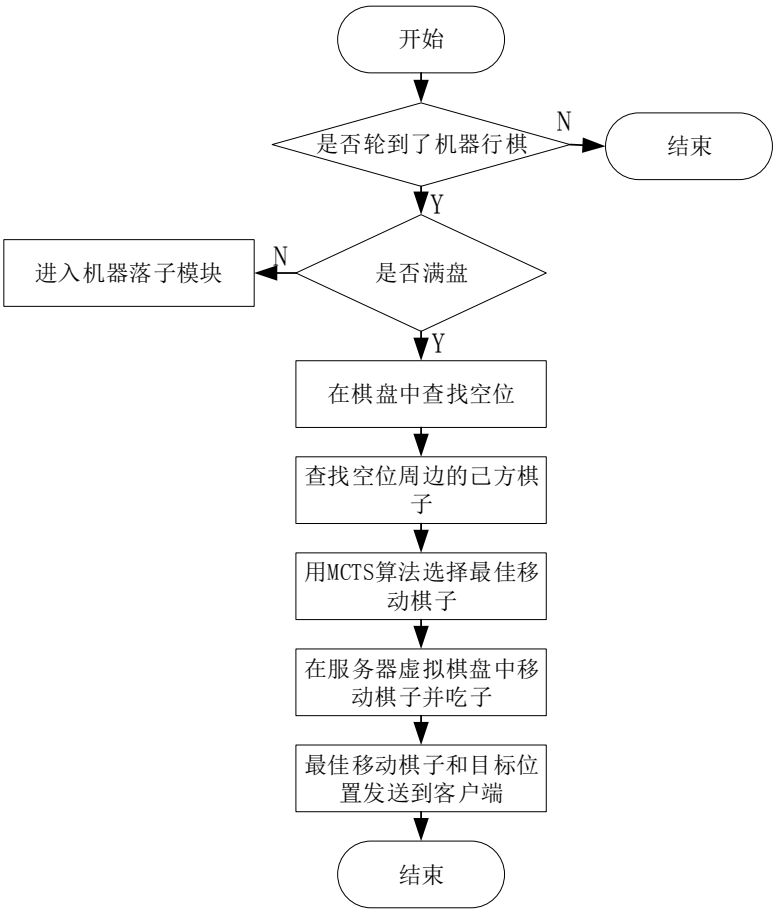


图 4-12 机器移动棋子控制模块流程图

4.3.5 机器让路控制模块

机器让路控制模块是移局阶段的一种特殊情况。让路其实也是一种移动棋子，但它移动的性质不一样。当轮到用户移动棋子是移动点周围没有己方棋子，而无法移动棋子时，又让计算机来移动棋子给用户让出一条移动路线，此时才会进入机器让路控制模块。在机器让路控制模块中有一种普通的移动让路方法和一种特殊跳棋让路方法。当机器要让路时首先判断是否移动让路，如果不能移动让路才能用跳棋的放来让路。总之机器要一步之内给用户让出一条移动的路线。因此机器让路控制模块的着法生成器有点复杂，它关系到棋盘中所有棋子和棋子的位置。搜索算法没运行前要确定能一步让路的所有棋子，但随着棋局的不同，能一步让路的棋子也不同。所以博弈树的

规模大小也不确定，从而搜索算法消耗的时间也不确定。机器让路控制模块流程如图 4-13 所示。

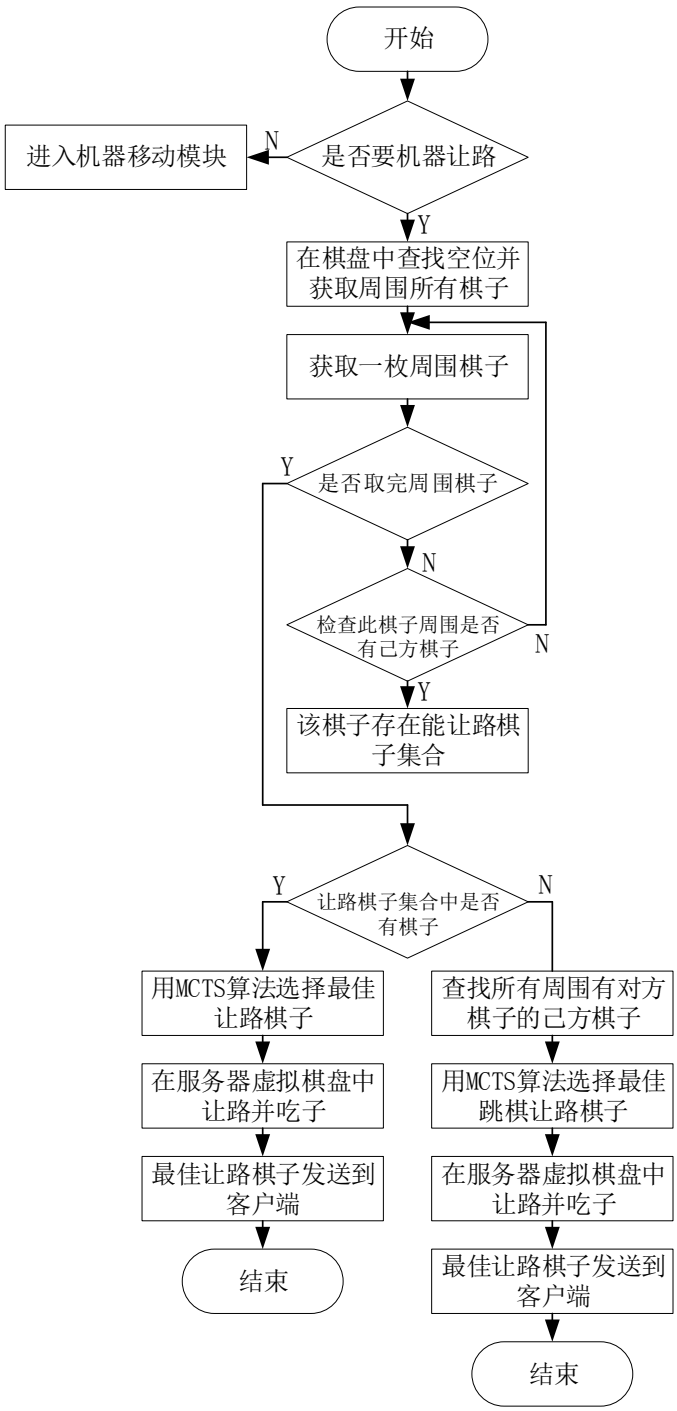


图 4-13 机器让路控制模块流程图

4.4 藏式夹棋博弈引擎的数据结构

在藏式夹棋计算机博弈系统中主要有三类数据结构：一是棋盘的相关数据，

用于记录棋盘的大小和棋盘上每个位置上的棋子数据；二是棋子的相关数据，用于记录棋子的相关属性；三是与棋局保存的相关数据。

4.4.1 棋盘表示

为了在软件运行时能够清楚地观察对弈时棋局的状态，棋盘的信息进行量化，使用一种数组来进行表示棋盘上的所有信息。表示棋盘信息的数据结构直接会影响软件运行的效率及空间复杂度。藏式夹棋是 5×5 的类似于表的网格，除了五条横线和五条竖线的交叉点以外，另有七条斜线来交叉的点组成。因此藏式夹棋的虚拟棋盘存储结构可以定义为如下：

$$Chess[i,j] \quad 0 \le i \le 8 \quad 0 \le j \le 8 \qquad (4\text{-}1)$$

其中 Chess 表示为棋子类型，具体在下面介绍。i 表示为虚拟棋盘的 X 坐标，j 表示为虚拟棋盘的 Y 坐标，因此Chess[i,j]表示棋盘上的某个点。公式(4-1)得知虚拟棋盘的大小 9 × 9 的二维数组，跟实际的藏式夹棋相比有点大。其实藏式夹棋的棋盘处了五横五竖的横纵线交叉而形成的 25 个交叉点以外，还有横纵线形成了 16 个小格子，这 16 个小格子的对角线的交叉点也是棋盘的落子点。因此藏式夹棋的虚拟棋盘的二维数组的大小定义为 9 × 9 。这样设计的目的首先很直观，其次计算机运算比较方便。从而用空间换时间的办法将虚拟棋盘设计的较大一点。图 4-14 是藏式夹棋虚拟棋盘的示意图。

00	01	02	03	04	05	06	07	08
10	11	12	13	14	15	16	17	18
20	21	22	23	24	25	26	27	28
30	31	32	33	34	35	36	37	38
40	41	42	43	44	45	46	47	48
50	51	52	53	54	55	56	57	58
60	61	62	63	64	65	66	67	68
70	71	72	73	74	75	76	77	78
80	81	82	83	84	85	86	87	88

图 4-14 藏式夹棋虚拟棋盘示意图

上图中数字表示为虚拟棋盘的下标。比如，X 坐标为 0 和 Y 坐标 4 的位置用 04 表。其中白色的元素为虚拟棋盘的落子点，黑色的元素中不能存数据，也就是在实际的藏式夹棋棋盘中没有这个点。

4.4.2 棋子的表示

在藏式夹棋中棋子的属性主要有棋子类型和棋子位置两种。其他还有棋子的序号、计算棋子在棋盘的位置、计算某一枚棋子是否在棋盘中、拷贝一个棋子的信息等等。在程序中把类 Chess 定义为一枚棋子。

在实际藏式夹棋中棋子的类型有白棋和黑棋两种，但在程序中空位也要表示，因此在程序中棋子的类型有白、黑、空等三种。可以用枚举类型来表示棋子的类型。White 表示白棋，Black 表示黑棋，Empty 表示空。

棋子的位置是指棋子在棋盘中的位置，在程序中两个变量来表示棋子的行列位置。posX 表示为棋子的行位置，posY 表示为棋子的列位置。图 4-2 为棋子的两个主要属性的伪代码：

```
/// <summary>
/// 棋子的类型
/// </summary>
public enum ChessType
{
  Empty = 0,//空子
  Black = 1,//黑子
  White = 2,//白子
}
```

```
/// <summary>
/// 棋子的横坐标
/// </summary>
public int posX;
/// <summary>
/// 棋子的纵坐标
/// </summary>
public int posY;
```

图 4-15 棋子类 Chess 的两个属性伪代码

为了将用户界面的棋盘位置与虚拟棋盘的位置相互关联，用一种计算公式来互换两个棋盘之间的位置。计算公式如下：

$$pos = \left[\frac{Point - D}{S}\right] \qquad (4-2)$$

在公式(4-2)中 S 为棋盘中两条横线之间的距离；Point 为用户在棋盘点击的坐标位置，计算 PosX 时 Point 为 X 坐标位置，计算 PosY 时 Point 为 Y 坐标位置；D 为棋盘边缘到窗口边缘之间的距离。

4.4.3 棋谱表示与存储

在计算机博弈领域各种棋的棋谱保存格式有很多种。主流的棋谱保存格式有 SGF（Smart Game Format）、BOX、MGT、GOS、NGF 等等很多种，但在这里按藏式

夹棋的特点用了一种新的棋谱存储格式。用 XML 文件格式来存储藏式夹棋谱，XML 文件具有较强的数据存储能力和分析能力，它的简单使其易于在应用程序中读写数据。以下是用 XML 文件存储藏式夹棋棋谱的表示方式：

表 4-1 棋谱标签表示表格

标签	表示	标签	表示
MG	表示棋类	CM	表示移动阶段记录
SZ	表示棋盘大小	CR	表示步骤
UN	表示程序开发单位	CC	棋子颜色
DT	表示比赛时间	CL	表示落子位置
PB	表示黑方名称	CJ	表示移动目标位置
PW	表示白方名称	ER	表示赢棋方
PC	表示布局阶段	TibetanGo	表示一个棋谱

图 4-16 藏式夹棋棋谱

4.5 博弈程序通信模块设计

4.5.1 计算机藏式夹棋通信协议

计算机藏式夹博弈引擎与程序客户端是在一个局域网网络环境中对弈，在对弈过程中博弈引擎与客户端之间要交换信息。因此，博弈引擎程序和客户端程序都需要一个通信接口模块负责信息的接受与发送。

目前，在计算机围棋等棋类项目的对弈过程中使用的是一种基于文本的协议 GTP。该协议是用于规范计算机围棋博弈程序对外的接口，为了使不同的围棋程序之间通过一个统一的接口来实现对弈。通过藏式夹棋的特殊情况，定义了一种藏式夹棋的通信协议。该协议将藏式夹棋博弈引擎与客户端在信息收发的过程中，为了保证能够读懂传送过来的数据信息，就必须对传送的数据格式进行约定，之后双方按照统一的格式约定进行数据的接受与发送。如表 4-2 为藏式夹棋通信协议命令。

表 4-2 藏式夹棋通信协议命令

功能	请求	发送数据格式	返回	返回数据存送格式
登录	Login	Login,用户名	Tables	Tables,房间个数
选座	SitDown	SitDown，房间号	SitDown	SitDown，提示信息
开局	Ready	Ready,房间号,棋子颜色,用户名	Start	Start,棋子颜色,提示信息
认输	Lose	Lose，房间号	Lose	Lose，提示信息
落子	SetDot	SetDot,房间号,棋子颜色,位置	SetDot	SetDot，提示信息
计算机落子	Switcher	Switcher,房间号,棋子颜色，输赢	SetDot	SetDot,房间号,棋子颜色,位置,输赢
吃子	Color	Color，房间号,棋子颜色,位置	Color	Color，提示信息
计算机动棋	Move_chess	Move_chess,房间号,棋子颜色	SetDotD	SetDotD，房间号,棋子颜色,原位置，目标位置
动棋	SetDotD	SetDotD，房间号,棋子颜色,原位置，目标位置	SetDotD	SetDotD，提示信息
计算机让路	Give_way	Give_way,房间号,棋子颜色	SetDotD	SetDotD，房间号,棋子颜色,原位置，目标位置
悔棋	Repent	Repent,房间号，位置	Repent	Repent，提示信息

4.5.2 系统通信模块设计

在藏式夹棋博弈系统中通信双方分别是客户端和博弈引擎，一旦通信连接建立之后，客户端会向博弈引擎发送请求命令，然后博弈引擎会对命令而做相应的操作后做相应的答复。通信双方在通信的过程中要服从藏式夹棋通信协议命令，按协议命令来做相应的操作并同步通信双方的棋盘信息。具体的步骤为：

首先用户向博弈引擎申请房间，博弈引擎接到用户请求后建立对弈空间并答复用户。建立了对弈空间后正式进入了的对弈状态，引擎会按用户发送的请求命令来执行相应的操作直到对弈结束。通信模块的流程如图 4-17 所示。

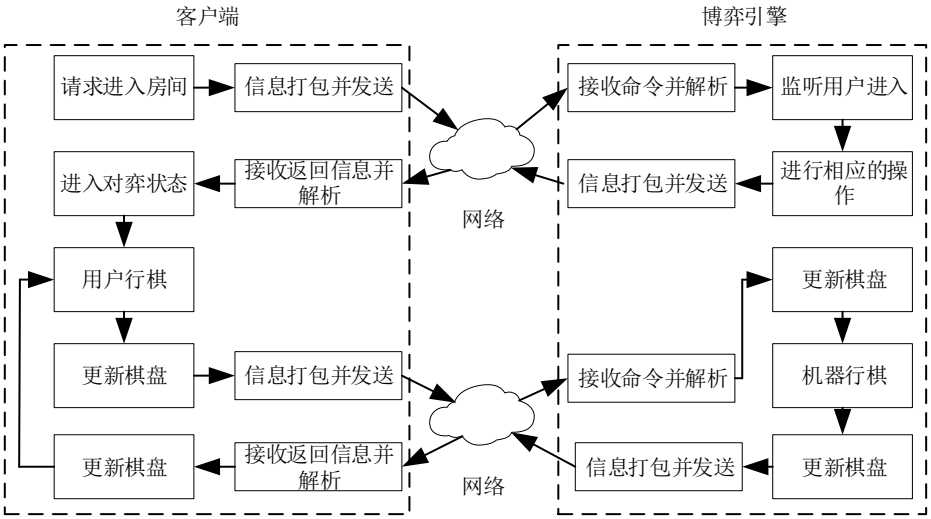


图 4-17 博弈通信模块流程图

4.6 计算机藏式夹棋人机博弈系统的实现

采用 Visual Studio 2017 编辑工具，用 C#语言实现藏式夹棋人机博弈系统“TibetanGo”。根据以上的系统结构的设计，实现了藏式夹棋人机博弈系统的登录、客户端棋盘、博弈引擎和参数设置模块等可视化界面。

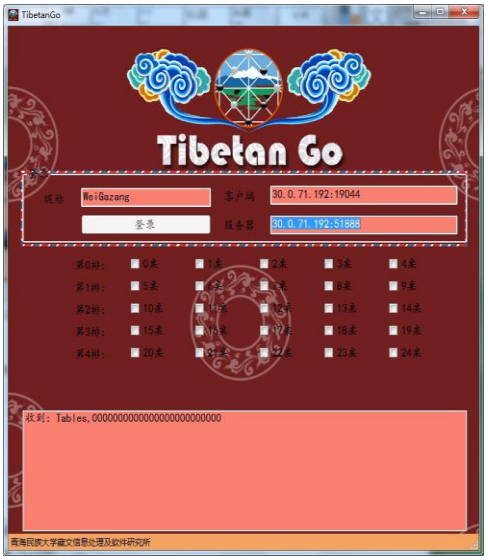


图 4-18 登录界面

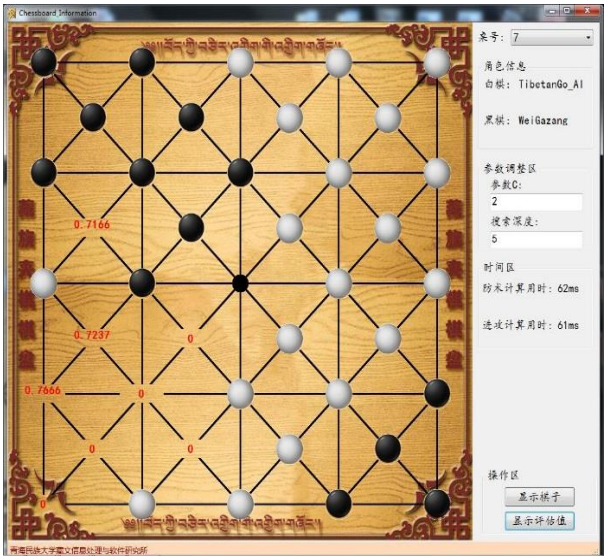


图 4-19 参数调整界面

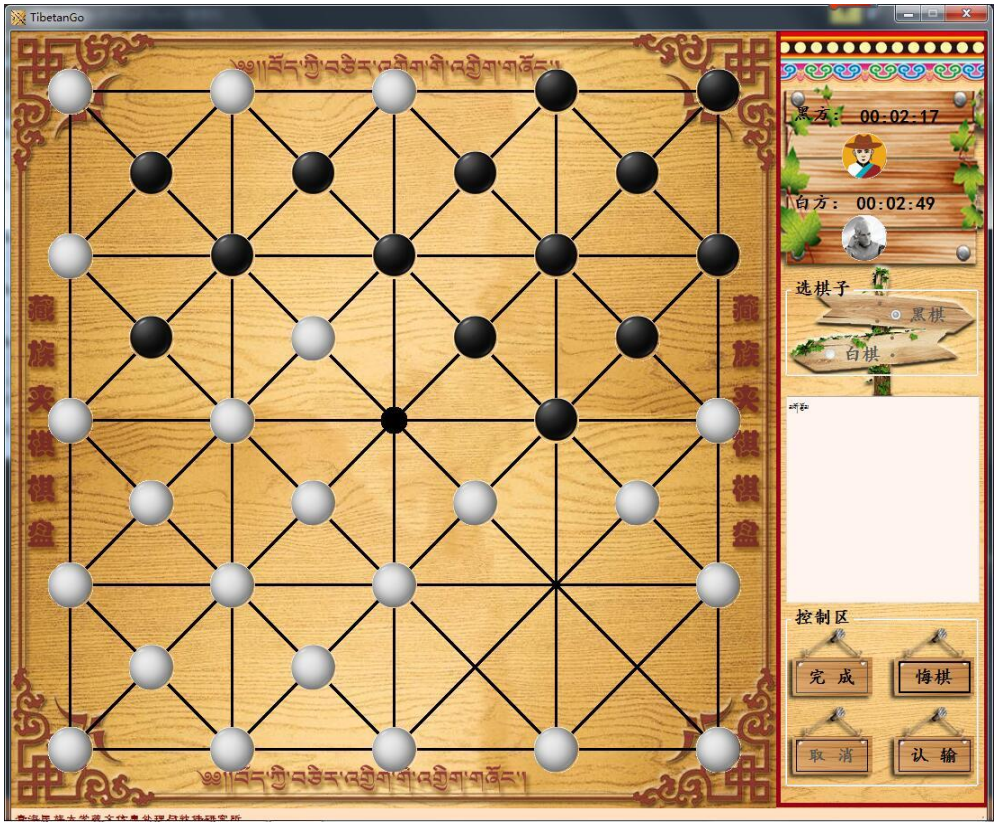


图 4-20 客户端棋盘界面

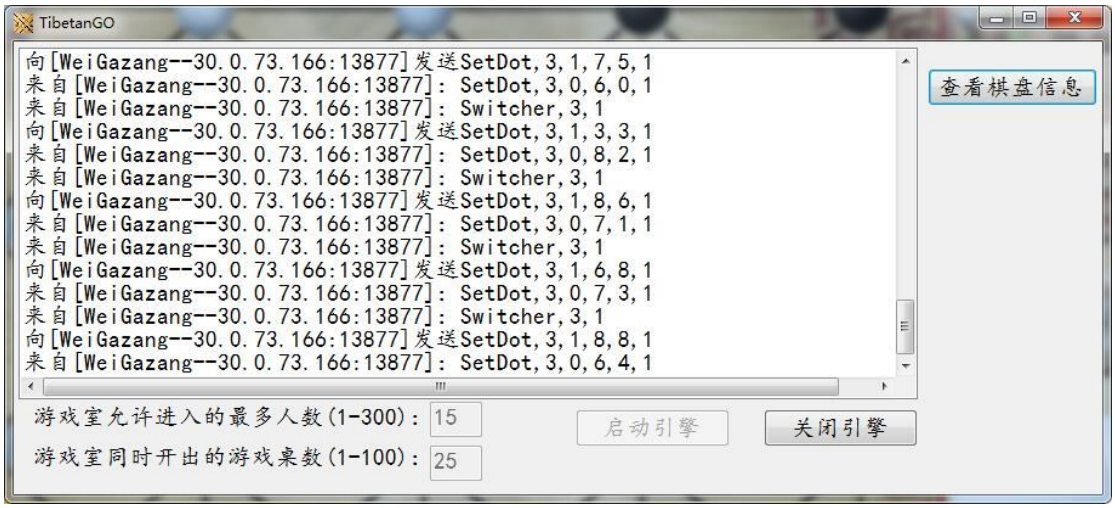


图 4-21 博弈引擎界面

4.7 本章小结

在本章主要介绍了藏式夹棋人机博弈系统的设计与实现，首先对系统几个主要的数据结构的进行了说明，然后详细地分析和设计了系统的总体结构图和系统的流程图；其次详细地介绍各模块的流程设计；最后展示了系统实现后的界面。

第5章 实验评测与结果分析

影响一个计算机博弈软件棋力的元素非常多，但一般博弈软件的好坏主要检查博弈效率和博弈水平。因此这里也从这种方面对计算机藏式夹棋人机博弈系统的性能进行测试和分析。

5.1 实验环境

本章将对第三章提出的原始 UCT 算法与优化后的基于静态评估的 UCT 算法进行对比实验。计算机博弈系统对计算机硬件的配置要求一般比较高，因为当面对一些博弈树庞大的棋类，它的搜索空间大，因此会占很大的内存空间和 CPU 空间。虽然计算机藏式夹棋的博弈树不像围棋那样非常庞大，但相比其他一些博弈树的搜索空间大。因此为了结果更具有代表性，系统的实验环境如表 5-1 所示

表 5-1 实验环境

配置名称	配置型号	配置属性
CPU	Intel(R) Core(TM) i5-4590CPU	3.3GHz
内存	DDR2 800MHz	4GB
操作系统	Window7	64-bit
屏幕分辨率	NVIDIA GeForce GT 705	1440×900

5.2 搜索效率测试

5.2.1 实验方案

在本节为了实验测试计算机藏式夹棋人机博弈系统的搜索效率，将原始 UCT 算法和基于静态评估的 UCT 算法在搜索深度相同的前提下比较搜索效率。藏式夹棋在开局阶段搜索树较大，因此测试搜索效率时只有开局阶段的二十步。两种算法分别进行 50 局测试，并统计了开局阶段前二十步的平均时间，作为衡量两种算法在相同搜索深度中的搜索效率。测试结果为图 5-1 所示。

5.2.2 实验结果

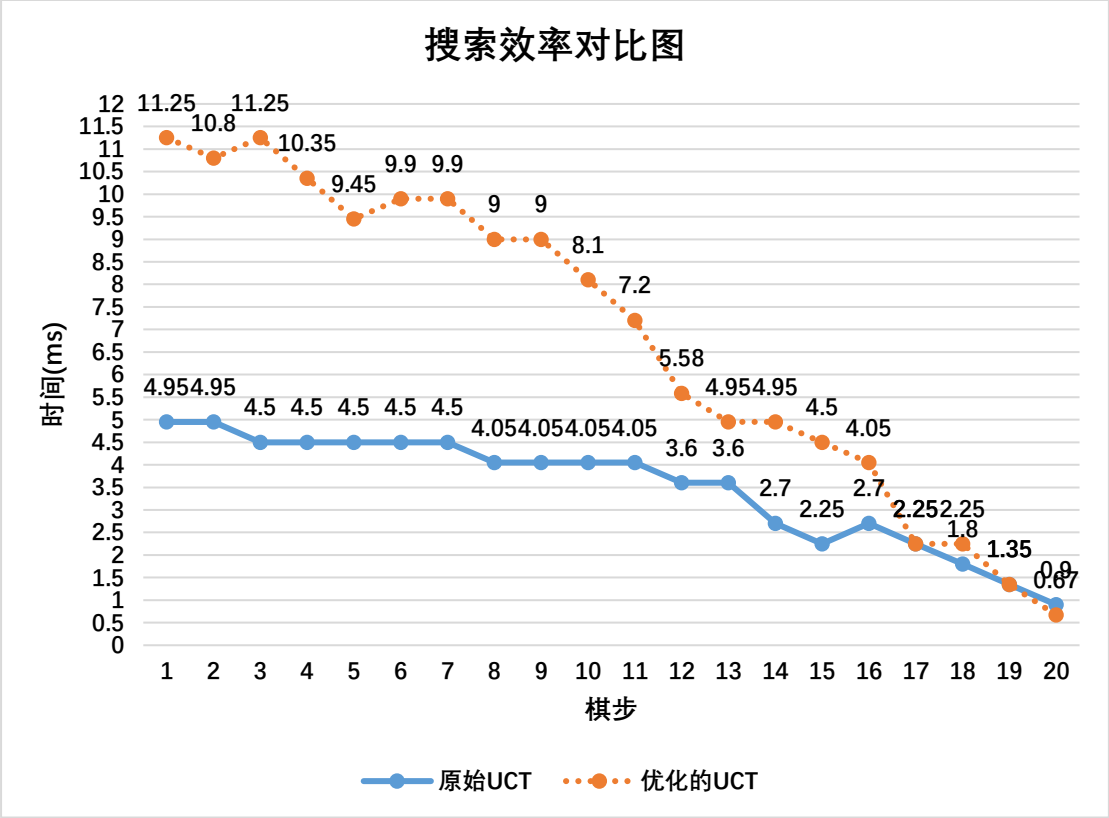


图 5-1 搜索效率对比图

5.2.3 实验结果分析

由图 5-1 所示，两种算法都随着棋步的增加，棋盘中可落子点不断减少，对应博弈树的大小也不断缩小，所以每步所用的时间也少。但原始 UCT 算法相比优化的 UCT 算法每步所用的时间几乎多一倍。原因是优化的 UCT 算法在静态评估和节点排序方面所用的时间多，而原始 UCT 算法没有离线知识的引导，只有随机的模拟评估，因此所用的时间相对少。

5.3 博弈水平测试

5.3.1 实验方案

在本节为了实验测试计算机藏式夹棋人机博弈系统的博弈胜率，将原始 UCT

算法和基于静态评估的 UCT 算法在搜索深度相同的前提下比较博弈胜率。两种算法分别进行 100 局测试，统计胜负次数并计算胜率，作为衡量两种算法在相同搜索深度中的博弈水平。测试结果为图 5-2 所示。

5.3.2 实验结果

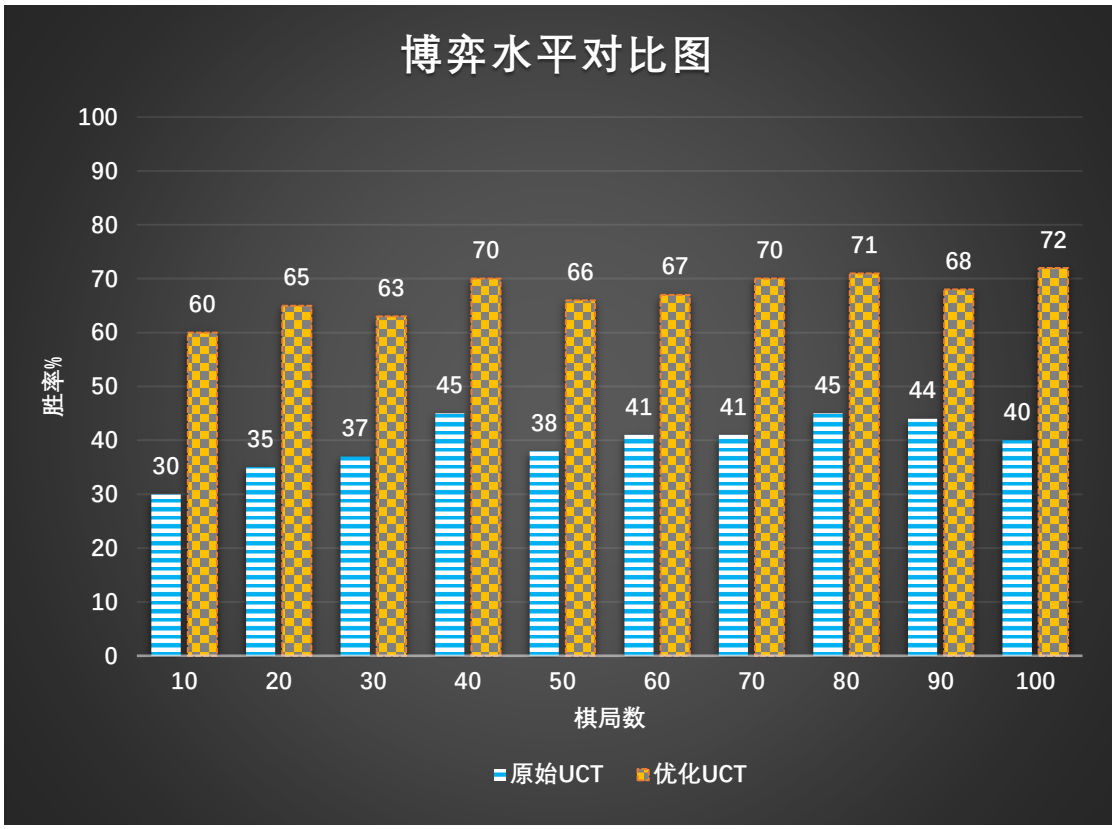


图 5-2 博弈水平对比图

5.3.3 实验结果分析

由图 5-2 所示，两种算法随着对弈的局数增加胜率逐渐平稳。两种算法中优化后的 UCT 算法胜率达到了 67.2%，而原始的 UCT 算法胜率只有 39.6%。由此可以证明原始 UCT 算法和静态评估结合将落子点的排序方法是可行并且非常有效，该方法明显提高了博弈的获胜率。但同时可以看出该方法是一种效率来换取胜率的方法，对弈时所用的时间相对多。

5.4 本章小结

本章将本课研究的核心问题藏式夹棋人机博弈系统的搜索效率和胜率进行了实验。同时为了证明优化的算法可行性，进行对比测试并将测试结果用图形来展示。

第6章 总结与展望

6.1 总结

本文对藏式夹棋的着法规则做了全面的探讨，并在此基础上学习和研究了国内外对计算机博弈方面的领先技术。借鉴一些目前较为成功的计算机围棋程序，将计算机藏式夹的博弈树搜索和局面评估作为核心的研究对象，最终实现了一种基于 MCTS 和静态评估结合的藏式夹棋人机博弈系统。

具体工作内容包括以下几个方面：

（1）起初主要学习和研究国内外对计算机博弈相关知识，进而研究目前常用的博弈树搜索方法和局面评估方法。并进一步分析了不同方法的优点和缺点。

（2）其次开始核心算法 MCTS 的研究工作，探讨了关于蒙特卡洛方法、蒙特卡洛评估、蒙特卡洛搜索、UCB 策略、UCT 算法等的原理以及各个之间关系。同时研究了每个策略应用在藏式夹棋中的方法，并分析了每个策略对计算机藏式夹棋优缺点。最终按藏式夹棋的着法特点，提出了一种藏式夹棋静态评估和 UCT 算法结合的方法。

（3）对计算机藏式夹棋的理论知识掌握以后，进而开始设计并实现了能够在局域网内进行人机对弈的计算机藏夹棋博弈平台。通过设计和分析系统的整体框架，将系统分为博弈引擎、客户端和通信模块等三个大模块。其次将每个模块进行细化，设计和分析了各个模块的功能。

（4）经过博弈系统的反复对弈，测试了模块与模块之间协调性。并且经过多次的对弈测试看出藏式夹棋博弈系统的博弈水平有明显提高，从而证明了提出的方法的有效性和可用性。

通过本文的研究，本人从中受益匪浅。一方面使本人了解人工智能的魅力，同时也了解人工智能的复杂性。特别在计算机藏式夹棋博弈系统实现的过程遇到了种种困难，解决每一次困难时对自己的专业知识的提高也有非常大的帮助。另一方面在探讨和研究藏式夹棋的着法规则时，本人体会到了藏棋文化的奥妙与独特性，从而对设计和开发藏式夹棋博弈系统更有信心和责任感。望本人开发的藏式夹人机博弈系统对藏棋文化传承和发展起到微薄之力。

6.2 展望

目前，在国内外研究关于计算机藏式夹棋的较少，本文借鉴一些其他棋类的研究成果，结合藏式夹棋的特点开发了一款藏式夹棋人机博弈系统，并获得了一些初步成果。但由于计算机藏式夹棋的博弈技术的复杂度高，而有些模块的效果不太理想。比如，静态评估和 UCT 结合的方法，虽然提高了博弈水平，但博弈效率一定程度上变弱。

在未来的工作中要解决的问题还众多，主要问题还是计算机藏式夹棋博弈效率和水平。可以通过引入目前更先进的技术遗传算法、神经网络等机器学习的方法来提高博弈效率和水平。也可以用提高静态评估的性能，并用多线程并行的方法来提高博弈效率和水平。总之，在计算机藏式夹棋博弈方面还有很大的探讨和研究的空间。

参考文献

[1]刘强.藏族传统棋艺现状及推广价值[J].民族传统体育.2012

[2]王昕杨.藏式围棋博弈软件及其教育应用技术研究[D].中央民族大学.2016

[3]张柳.基于极大极小搜索算法的亚马逊棋博弈系统的研究[D].东北大学.2010

[4]刘知青.李文峰.现代计算机围棋基础[M].北京邮电大学出版社.2011

[5]Schaeffer J.Burch N,Bjornsson Y,et al.Checkers is solved[J]. Science ,2007

[6]DavidN.L Levy.eds.Computer Gamese[M],NewYok:Springer NewYok Inc, 1988

[7]Siver D,Schrittwieser J,Simonyon K,et al.Mastering the game of Go without human knomledge[J].Nature,2017

[8]周明明.基于专家系统和蒙特卡罗方法的计算机围棋博弈的研究[D].南京航空航天大学.2012

[9]张玉琪.基于静态评估的计算机围棋UCT算法改进研究[D].南昌航空大学.2015

[10]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

[11]宋兴亮.中国象棋博弈树搜索算法研究与实现[D]. 沈阳工业大学.2012

[12]唐艳.围棋博弈机器学习算法的研究及应用[D].重庆理工大学.2012

[13]王骄.徐心和.计算机博弈：人工智能的前沿领域——全国大学生计算机博弈大赛[J].计算机教育.2012

[14]刘洋.点格棋博弈中UCT算法的研究与实现[D].安徽大学.2016

[15]马小明.藏棋_密芒_与_久_的比较研究[J].青海民族大学学报.2017

[16]更堆.淡西藏_密芒_围棋的发现和相关传统藏棋种类[J].西藏大学学报.2003

[17]韩海兰，尚涛.“棋”乐无穷——揭秘藏棋文化的前世今生[N]. 西藏日报.2016-10-24

[18]裴生雷. 王宫双门棋博弈系统中的关键问题研究[J].微计算机信息.2012

[19]仁增多杰.安见才让.“裕链”藏棋的设计与实现[J]. 信息与电脑（理论

版）.2014

[20]才让拉毛.安见才让.“鱼儿”藏棋的设计与实现[J].电脑知识与技术，2016

[21]仁青东主.安见才让.藏族王棋游戏的设计与实现[J].信息与电脑(理论版).2016

[22]邓颂庭.久棋博弈原型系统及多媒体课件设计与实现[D].中央民族大学.2017

[23]李霞丽.基于棋型的藏族_久_棋计算机博弈研究[J].智能系统学报.2018

[24]Time Hoff Kjeldsen.John von Neumann's Conception of the Min-max Theorem[J],A Journey Through Different Mathematical Contexts Contexts Exact Sci,2001,1(56): 39-68

[25]曹一鸣.基于蒙特卡罗树搜索的计算机扑克程序[D].北京邮电大学.2014

[26]于永波.基于蒙特卡洛树搜索的计算机围棋博弈研究[D].大连海事大学.2015

[27]刘宇.Monte_carlo方法在计算机围棋中的应用[D].电子科技大学.2011

[28]杨周凤.国际跳棋完备信息博弈关键技术研究与系统设计[D].沈阳航空航天大学.2018

[29]陈钧.中国象棋人机博弈系统的设计与实现[D].厦门大学.2013

[30]高春生.计算机围棋中落子预测与死活问题的研究[D].昆明理工大学.2017

[31]孙若莹.一种新的博弈树迭代向前剪枝搜索[J]. 沈阳工业大学学报.2017

[32]季辉.双人博弈问题中的蒙特卡洛树搜索算法的改进[J].计算机科学.2018

[33]马小明.围棋与藏棋_密芒_的比较研究[J].青海师范大学学报.2017

[34]刘子正.基于蒙特卡罗树搜索的_2048_游戏优化算法[J].控制工程.2016

[35]马少龙.藏棋文化的传承和保护[J].西藏研究.2017

[36]范忠雄.对弈熵率在藏棋_杰布杰曾_上的应用[J].西藏大学学报.2014

[37]吕艳辉.计算机博弈中估值算法与博弈训练的研究[J].计算机工程.2012

[38]徐心和.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

[39]张聪品.博弈树启发式搜索的剪枝技术研究[J].计算机工程及应用.2008

[40]王亚杰.计算机博弈的研究与发展[J].智能系统学报.2016

[41]王建雄.博弈树启发搜索算法在五子棋游戏中的应用研究[J].科技情报开发

与经济.2011

[42]邢森.五子棋智能博弈的研究与设计[J].电脑知识与技术.2010

[43]张利群.曹杨.计算机博弈平台构建研究[M].中国石化出版社.2017

致谢

岁月如梭，如歌。转眼间，三年的研究生求学生活即将结束，站在毕业的门槛上，回首往昔，奋斗和辛劳成为丝丝的记忆，甜美与欢笑也都尘埃落定。青海民族大学以其“进德修业，自强不息”的校训教我求学、育我成长。值此毕业论文完成之际，我谨向所有关心、爱护、帮助我的人们表示最真诚的感谢与最美好的祝愿。

本论文是在安见才让导师的悉心指导之下完成的。三年来，安见才让导师渊博的专业知识，严谨的治学态度，精益求精的工作作风，诲人不倦的高尚师德，朴实无华、平易近人的人格魅力对我影响深远。安见才让导师不仅授我以文，而且教我做人，虽历时三载，却赋予我终生受益无穷之道。本论文从选题到完成，几易其稿，每一步都是在安见才让导师的指导下完成的，倾注了导师大量心血，在此我向我的导师表示深切的谢意与祝福！

本论文的完成也离不开计算机学院的各位领导和各位老师的准确教导下，本论文才得以圆满落幕。由衷的感谢您们在这三年期间给予我帮助和指导，让我在学习和生活，思想上都有所收获。在这三年的时光里，我的成长自然也离不开同学们的帮助和陪伴。真的很庆幸遇见公保杰、德格加、多拉、道吉仁青、相毛吉、多杰措这几位同窗三年的同学，他们是一群总在你需要帮助时伸出援手的朋友，同时也要感谢学长学姐和学弟学妹给予的关心和支持。因为你们的存在，让我的读研生活充满了意义。

每一段路，每一个旅程，都离不开至亲好友的拥护。感谢每一次成长过程中母亲含辛茹苦的付出；感谢家人一直以来的支持与鼓励；感谢朋友一直并肩同行。

攻读硕士学位期间发表的学术论文目录

学术论文

[1]尕藏扎西，安见才让.基于 CYK 的藏语句法分析器研究与实现.计算机时代，2018，第 6 期

[2]尕藏扎西，安见才让.计算机藏式夹棋博弈系统中局面估值方法的研究.计算机时代(已录用)

软件著作权

[1]青海民族大学.基于 CYK 藏文句法分析器软件.1.0 版本.2018.6.1

[2]青海民族大学.网络版藏式夹棋游戏软件.1.0 版本.2018.6.1

[3]青海民族大学.网络版藏族七方棋游戏软件.1.0 版本.2018.6.1

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：

日　　期：2019 年 5 月 27 日

学位论文版权使用授权书

本学位论文作者完全了解青海民族大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权青海民族大学研究生工作部可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书，本论文：□不保密，□保密期限至　　年　　月止）。

学位论文作者签名：　　导师签名：

签字日期：2019 年 5 月 27 日　　签字日期：2019 年 5 月 27 日

学位论文作者毕业后去向：

工作单位：　　电话：（　）

通讯地址：

邮编：