分 类 号：　　密　级：
论文编号：　　学　号：30708120301

重庆理工大学硕士学位论文

基于智能算法的六子棋博弈行为选择的应用研究

研　究　生：　陈 光 年
指 导 教 师：　张 小 川 教 授
学 科 专 业：　计 算 机 应 用 技 术
研 究 方 向：　计 算 智 能 与 智 能 软 件
培 养 单 位：　计 算 机 科 学 与 工 程 学 院
论文完成时间：　2010 年 4 月 10 日
论文答辩日期：　2010 年 6 月 1 日

Category Number： **Level of Secrecy：**

Serial Number ： **Student Number：30708120301**

Master's Dissertation of Chongqing University of Technology

The Research and Application on the Behavior Eelection of Connect6 Based on Intelligent Algorithms

Postgraduate: **Chen Guangnian**

Supervisor： **Prof. Zhang Xiaochuan**

Specialty: **Computer Applied Technology**

Research Direction: **Artificial Intelligence**

Training Unit: **College of Computer Science and Engineering**

Thesis Deadline: **April 10, 2010**

Oral Defense Date: **June 1, 2010**

重庆理工大学

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，独立进行研究所取得的成果。除文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果、作品。对本文的研究做出重要贡献的集体和个人，均已在文中以明确方式标明。

本人承担本声明的法律后果。

作者签名：　　日期：　　年　月　日

学位论文使用授权声明

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权重庆理工大学可以将本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和汇编本学位论文。

本学位论文属于（请在以下相应方框内打“√”）：

1.保密□，在___年解密后适用本授权书。

2.不保密□。

作者签名：　　日期：　　年　月　日

导师签名：　　日期：　　年　月　日

摘　要

行为是生命体外在的表现形式，是生命体内在智能的外部表象，Tyrell 认为：“行为选择就是从一组可能的候选集中选择最适合的行为。”因此，行为选择是生命体智能的高级形式。行为选择问题是人工生命研究领域的一个核心问题，人工生命是人工智能的发展，人工生命作为信息科学、生命科学、系统科学等学科的交叉学科，它不是用分析、解剖生命体的方法来理解生命，而是用综合的方法来理解生命，强调系统性和整体性。此外，计算机博弈过程，本质上就是一个对抗性极强的、智能程度高的博弈行为的选择过程，因此，将基于计算机博弈系统的博弈机器人作为人工生命体，并利用人工智能方法来研究智能系统是可行的，也具有重要研究意义。

模仿人类的博弈行为选择过程，本文将博弈机器人划分为“大脑”、“视觉”、“记忆”、“控制”等 4 个部分，文章所依托的科研项目的最终目标是构造一个在物理棋盘上与人类对弈的博弈机器人，本文的主要工作是设计“大脑”。论文主要研究了以下 4 个方面的问题：

第一、设计实现了一套博弈系统，包括棋盘和棋子在计算机中的表示问题，走法生成，搜索技术，估值函数等。

第二、针对基于棋形的六子棋博弈系统中，棋形难以判断和统计的问题，提出并规范了“路”的思想，使对局面进行评估前的准备工作大大简化。

第三、针对静态估值函数依赖人类棋类知识和评估不够准确的问题，本文在前人研究的基础上，对遗传算法的应用进行了改进，克服了原来遗传算法过早收敛的问题，同时提高了种群的多样性，实验结果证明该方法有效。

第四、同样针对静态估值函数依赖人类棋类知识和评估不够准确的问题，提出了采用基于聚焦距离动态调整惯性权重的方法，以改进微粒群算法，以此优化相关参数。实验证明，该算法比遗传算法搜索速度更快，结果更优，是解决评估函数参数优化的有效办法。

关键词：机器博弈　六子棋　评估函数　遗传算法　微粒群算法

Abstract

Behavior is the manifestation of life in vitro, and it is the external appearance of life's intelligent. Tyrell said: "Behavior selection is that selecting the most appropriate behavior from a group of potential candidates." Therefore, behavior selection is an advanced form of intelligent life. Behavior selection is the core issue of artificial life research. Artificial life is the development of artificial intelligence. Artificial life, as an interdisciplinary subject of information science, life science and system science, doesn't understand life by analyzing and anatomizing, but uses an integrated approach to understand life. It emphasizes on the systematic and holistic. In addition, the process of computer game is essentially a process of highly antagonism and intelligent behavior. Therefore, taking the game robot which based on computer game system as artificial life, and using intelligent methods to research the intelligent systems are feasible and have important research significance.

By mimicking the process of human's behavior of the game selection, this article divided the game robot into four parts - "brain", "visual", "memory", "control". The ultimate goal of the research project was constructing a game robot which could play game on physical board with human. The main work of this paper is designing the "brains". This article addressed the following four aspects:

First, a game system has been designed in this article. It includes the expression of the board and pieces in the computer move generation, search technology and evaluation functions.

Second, for the problem that it's difficult to judge and count the chess pieces in the Connect6 game system, the thinking of "path" has been proposed and standardized. It makes the preparatory work before evaluating the situation greatly simplified.

Third, for the problem that the static evaluation function relying on human's chess knowledge and it is not accurate enough, based on previous studies, the application of genetic algorithms has been improved. It overcomes the original problem of premature convergence of genetic algorithm, and it increases the diversity of population. The experimental results show that the method is effective.

Fourth, for the problem that the static evaluation function relying on human's chess knowledge and it is not accurate enough, a new adaptive Particle Swarm Optimization algorithm which dynamically changes inertia weight based on the current rate of cluster focus distance was proposed to improve the PSO algorithm and optimize parameters.

Experiments show that the algorithm is faster than the genetic algorithm and the result is better than genetic algorithms. It is an effective way to solve the optimization of the parameters of evaluation function.

Keywords: machine game; Connect 6; evaluation function; genetic algorithm; particle swarm optimization

目录

1 绪论

1.1 引言

长期以来，研究生命现象的生命科学，是以地球上的碳水化合物为基础的生命为研究对象的，它的研究取得了令人瞩目的成就，如遗传密码、DNA 结构、蛋白质合成、克隆技术等。然而人工生命的开拓者则试图构建一些非蛋白质媒体的生命形式。早在 20 世纪 40 年代，科学通才 Von Neumann (冯·诺伊曼)提出能自我繁殖的细胞自动机，Penrose 提出用机械方法制造自繁殖的机器，1987 年，“人工生命之父”C.Langton 教授正式提出“人工生命（Artificial Life）”概念，她认为：“人工生命就是展示自然生命系统行为特征的人造系统”。 “人工生命”是“人造生命”，是对“自然生命”的模拟、研究和衍生。人工生命不仅是当前生命科学、信息科学、系统科学及工程技术科学的交叉研究的热点，也是人工智能、计算机科学、自动化科学技术的发展动向之一。因此，自 “人工生命”这个新领域于 1987 年诞生以来，许多学科的科学家们被吸引到其中。

生命的定义是什么？生命从何而来？生命可以创造吗？这些问题人们一直在探索并试图回答。人类在 20 世纪 60 年代破译了遗传密码，70 年代遗传工程有了重大突破，80 年代人类又计划测定人类基因组的碱基序列。显然，接下来人类研究的一个重要目标就是用人工的方法创造出生命。目前，以生命物质为基础创造人工生命是一条重要途径。但是，由于生命起源的机理尚未清楚，采用这样的途径前景并不乐观。如今，随着计算机硬件以及相关理论的飞速发展，可以尝试在计算机或其他媒质中创造出新形式的生命。于是在 80 年代末 90 年代初国际上兴起了一股探索用非有机物质创造新的生命形式的研究热潮，这就是“人工生命”。人工生命概念一提出，马上得到世界各地学者的热烈响应，吸引了众多学者参与到这一新兴的研究领域中。人工生命的研究进展一度成为《科学》（Science）杂志和《科学美国人》杂志报道的热点。人工生命不仅对传统生物学提出了挑战，而且也对我们最根本的社会、道德、哲学等的观点提出了挑战。基于计算机科学技术的人工生命方法是通过合成的、计算的方法去理解自然生命。

人工生命是人工智能的发展和应用；人工智能是人工生命的核心和基础。人工智能是一门具有挑战性的交叉性学科，这个领域的研究人员必须既要懂得计算机知识、心理学，还要懂得哲学及其它相关领域的知识。它的研究目标之一是模拟学习及记忆等复杂的生物过程。亚里士多德在几千年前就已经开始研究人类思维规律。在近代，帕斯卡制造世界上第一台机械式加法器；巢寄崇制成可进行四则运算的计算器，

并提出了“万能符号”和“推理计算”的思想，被后人尊为数理逻辑的第一奠基人；布尔创立了逻辑代数（即布尔代数）；巴贝奇致力于差分机和分析机的研究；图灵提出了理想计算机模型，并创立了自动机理论。这些早期的研究都为人工智能的正式出现奠定了基础。简单来说，人工智能的目的就是让某台机器如计算机能够像人类一样思考，具有类似人类的智慧。如今，我们的生活当中到处都涉及到人工智能，例如模糊控制，决策支持等等。人工智能对人类现代生活的影响越来越大，成为人们生活不可分割的一部分[1-4]。

机器博弈是人工智能学科和计算机学科的重要研究方向之一，其研究内容是利用各种智能算法、搜索策略等计算机博弈技术，在计算机上模拟人类博弈的过程，使机器具有近似于人类的智能。1997 年，IBM 公司的超级计算机“深蓝”与当时的国际象棋世界冠军卡斯帕罗夫进行了一场举世瞩目的比赛。“深蓝”成为这场比赛的胜利者，而“深蓝”计算机的取胜成为了人工智能研究领域的一个里程碑。机器博弈被称为人工智能的果蝇，因为人类对机器博弈的研究衍生了大量的研究成果，这些成果对更广泛的领域产生了重要影响，如航空调度、天气预报、资源勘探、军事博弈，在金融／经济调控等领域，也取得了大量引人瞩目的成果。

六子棋是人们较晚关注的机器博弈棋类项目之一，它继承了五子棋简单、易学的特点，取消了五子棋的禁手约束，相比五子棋来说还有以下优势：在变化复杂方面，由于除先行者第一手棋下一颗棋子外，其余行棋双方都是一次下两子，这样的规则，使得六子棋的棋形组合、变化就比五子棋多许多，棋形的复杂度已被评估为仅次于围棋及日本将棋棋种，远远高于五子棋及西洋棋，与中国象棋、国际象棋相当；而在游戏公平性方面，由于黑、白各方每次下完一手后，盘面都比对方多一子，因此赛局可自然达成平衡的状态，这使得公平性大为提升。在棋类游戏中，从象棋开始，开展机器博弈到现在，已有几十种游戏都开发了自己的博弈软件。六子棋虽然发展比较晚，但也有自己的博弈软件，而且发展相当的迅速。因此，本文选择六子棋作为对象，研究人工生命博弈行为选择的智能化问题。

1.2 课题的学术意义

1.2.1 人工生命体及其行为选择的研究价值

“人工生命”的概念最先是由计算机科学家 Christopher Langton 于 1987 年在洛斯阿拉莫斯国家实验室召开的“生成以及模拟生命系统的国际会议”上提出。到目前为止，人工生命还没有统一的定义，不同的学者有不同的看法。一种典型的定义是：人工生命是研究怎样通过抽取生物现象中的基本动力规则来理解生命，并且在物理媒

体如计算机上重建这些现象，使它们成为一种新的实验方式。人工生命具有多方面的研究意义：首先，它是生物学和生态学研究方法的补充，可以通过模拟生命来帮助人类验证生物的某些现象，充当某些生物信息学研究的“果蝇”，降低研究成本；其次，可以通过设计基于人工生命的新产品、新系统、新技术，为复杂系统的研究提供了新的思路与工具；最后，它还可以促进生命科学、信息科学、系统科学的交流与发展。因此，人工生命的研究及应用具有重大的科学意义和深远的社会影响[5]~[7]。

选择是所有生物都要面临的、共同的、常见的问题。比如，人在商场购物的时候，就存在选择什么商品的问题；驾驶员在行驶途中，面临选择什么道路的问题；人们进行股票买卖时，就面临选择什么股票，什么时候买进、卖出，以及买卖价位与数量等问题；再如，狮子在追捕鹿群过程中，面临进攻还是跟随，以及选择哪只猎物为进攻对象的问题……它们从本质上都体现了生物的智能性选择，归根结底是一个生物智能水平的问题。人工生命体的行为选择研究正是以人工生命为载体，模拟生命体的智能行为，将生物的智能注入该人工生命体，以此来研究生命体的智能结构或机制，发现生命体的内在本质。

既然人工生命的研究和应用具有重大的科学意义和深远的社会影响，而研究人工生命体所要解决的一个基本问题就是人工生命体的行为选择问题，因为不管是高等生物、低等生物、自然生命体或是人工生命体，所有这些生命体都必须在自己所处的当前环境中根据具体情况自主地选择自己的行为。只有当人工生命体对外部刺激和内部状态能够像人类或动物一样做出合理的反应时，才能称为是真正的人工生命体，因此，行为选择问题是研究智能的人工生命体所必须解决的一个重要基本问题，研究它具有重要的研究价值和前景。

1.2.2 机器博弈的研究意义

机器博弈首先是在国外发展起来的，早在计算机诞生的前夜，著名的数学家和计算机学家阿兰·图灵便设计了一个能够下国际象棋的程序[5]。如今，国内对机器博弈的研究也开始兴起，并举行了许多机器博弈大赛，很多高校学生以及部分计算机行业的工作人员参与到该项目中来，机器博弈已经逐渐渗透到我们的生活中。机器博弈是近年发展迅猛的一种人工智能的研究平台，它是以人们喜闻乐见的棋牌类活动为研究对象，以具体的博弈行为为研究内容的。人类在机器博弈的研究中衍生出大量的研究成果，机器博弈常被看作是人工智能的“果蝇”，而这些研究成果对其他领域也能产生重要影响。棋类游戏是一个标准博弈问题，下棋的双方无时不在调动自己的一切智能。所以，在人工智能领域中，始终将棋类的机器博弈作为最具挑战性的研究方向之一。

借助博弈问题来研究人工智能的一些典型问题，具有以下优点：

⑴博弈问题局限在一个小的有典型意义的范围内，研究容易深入。

⑵博弈问题非常集中的体现了人类的智能，已足以为现实世界提供新的方法和新的模型。

⑶专家经验容易获取。

⑷进展可以精确的展示和表现出来，不同方法和模型的优缺点也容易比较。

1.2.3 六子棋博弈的研究意义

六子棋是最近几年兴起的一种新的棋类游戏，属于完全信息博弈类型。虽然六子棋游戏已经逐渐兴起，并多次在国内外举办的机器博弈比赛中被列为比赛项目，但真正把六子棋引入到学术上来研究的事实相对较少。除台湾交通大学吴毅成教授给出了六子棋的公平性问题以及基于迫着(Threats)的胜利策略等外，六子棋计算机博弈问题很少被提及。六子棋计算机博弈系统比较著名的程序有 NCTU6、X6 和 EVG，但基本都采用吴毅成教授所给出的基于迫着的胜利策略，没有把计算机博弈技术完全地引入到六子棋计算机博弈系统。其他的一些棋类，比如国际象棋、围棋、中国象棋、五子棋等来说，世界各国的学者都已经成功地将一些先进的计算机博弈技术引入其中，很多博弈算法的应用也比较成熟，其计算机博弈水平也基本达到或超过了世界冠军水平。

六子棋规定先手方落 1 颗棋子，此后每方每次可同时下 2 颗棋子，这样就最大限度抵消了先手之利，提高了博弈的公平性，它不像五子棋，需要通过设置禁手来抵消先手之利(实际上，数学家已经证明，不设禁手，只要不出错误，五子棋的先手方必赢)。六子棋的特殊规则，使得其状态空间复杂度以及博弈树的复杂度成倍提高，大大高于五子棋，与中国象棋、日本将棋相当。与其他棋类不同，六子棋博弈必须综合考虑两步棋，需要综合评估两步棋的棋盘状态，在搜索算法上需要将“两步”当“一步”来处理，实现所谓的是综合搜索，这是六子棋计算机博弈的难点所在。

六子棋计算机博弈和其他棋类的计算机博弈一样，也可以采用一些基本的智能算法。比如重庆大学自动化研究所 2008 届硕士研究生张颖、研究生李果，利用智能算法做过六子棋的应用研究。本文在人工生命行为选择的理论指导下，将六子棋计算机博弈的研究目标拓展到提高博弈行为的智能性高度，尝试利用智能算法，按照仿人、仿生思路，采用离线方式，实现六子棋的人-机和机-机对抗，提高其博弈水平，这将对人们的娱乐、教学和科普等都具有重要意义。

1.3 国内外研究现状

1.3.1 人工生命行为选择的研究现状

生命源于动物，生命寓于系统，生命控于信息，生命精于智能。人工生命是展示自然生命系统行为特征的人造系统，它不仅要展示自然生命系统的外部行为特征，而且应当模拟、延伸、扩展自然生命系统的内在性能[6]。人工生命是多学科交叉的研究领域，虽然其研究历史不长，但其他相关研究领域都为它的研究奠定了理论基础，如行为学、心理学、生态学等对动物及其活动过程和性能的研究，人工智能、信息论、进化计算等对生命活动过程中信息的获取、传递和利用的过程的研究，都为人工生命的研究打下了坚实的基础。自从人工生命的概念提出以来，得到了专家学者的广泛关注，吸引了许多相关领域的专家投入研究，包括控制科学、系统科学、计算机科学、人工智能、机器人科学、生物科学、经济学、哲学和人类学等，其中美国、日本和欧洲的研究最为活跃。目前，国际上主要有三个关于人工生命的学术研讨会，一个是主要由美国承办的 ALIFE 系列国际研讨会；一个是由欧洲主办的“欧洲人工生命会议”(ECAL) ，它从 1991 年开始每两年举办一届；另一个是由日本举办的“人工生命和机器人”(AROB) 系列国际研讨会，从 1996 年开始每年举行一届。此外，IEEE 也召开了相关的国际会议——“进化计算”以及“仿真和适应行为”。国内外人工生命的研究可分为细胞级人工生命、器官级人工生命、个体级人工生命、群体级人工生命以及生态系统级人工生命等几个方面。

行为选择问题是人工生命重要的研究问题之一。对于动物来说，行为选择的最终目标是选取合适的行为，以增大后代中它的基因被复制的数量；对于机器人来说，为完成相关的任务，确立的行为选择机制必须以完成任务为目标并维护机器人的安全；对虚拟动物和自主的动画角色来说，行为选择的目标是获得令人满意的行为逼真度。在以前的行为动画工作中，有些科学家只用“刺激-驱动”行动选择过程建立行为选择模型，行为由环境条件直接激活，如 Reynolds 的 boid 群体。在一些机器人的行为选择设计中，也采取了与此类似的设计，如针对对于环境中出现的意外情况所设计的应激机制，就具有很大的优越性，但是这种机制没有考虑智能体的内部状态或内部动机，只单纯考虑“刺激-驱动”。这样不但难以处理好复杂的行为选择问题，而且也和动物行为选择的研究不符。Donnart 和 Meyer 设计了一个行为控制系统并进行了仿真，它的控制结构可以用两个产生式规则来描述：

行为规则：If <sensory information> and <current goal> then <action>

计划规则：If <sensory information> and <current goal> then <new current goal>

日本科学家也采用了相似的方法来实现机器人的行为选择，先建立情形集，然后建立基于情形基础上的行为集。行为选择的规则为：If a state then an action[7]。这两种

方法均采用行为准则来为机器人选择行为，都是根据当前的形势或条件来判定和选择相应的行为，当环境和行为复杂时，当前形势或条件的归纳就变得复杂甚至难于总结，因此这种行为选择策略难以处理复杂环境中的问题。国内的一些研究者曾用基于生长神经网络的进化机器人行为算法来实现机器人的行为选择，并进行了模拟。其特点是通过自然选择对神经网络进行训练和进化，自主实现机器人避碰、移动、复制和攻击等行为，但其进化效率并不高，需要较多的时间步后才能得到较好的效果。其他的行为选择方法中，有代表性的三种典型方法是：行为选择的分级处理方法、行为选择网、自由流动网。又如重庆大学、重庆理工大学和厦门大学的李祖枢教授、张小川教授、邵桂芳副教授等人，针对分级行为选择模型的不足，提出并建立了基于优先度的人工生命体行为选择模型[8,9]和基于评估权衡结构的行为选择模型[10,11]。

1.3.2 机器博弈的研究现状

40 年代后期，计算机博弈吸引了学者们的注意力。1946 年，美籍匈牙利数学家约翰·凡·诺依曼(John von Neumann)被指派设计一台强大的计算机器用以加快工作进度的任务。1950 年，一台叫“MANIAC 一号”的巨型机被交付使用，它的内部装有数千个电子管和开关，每秒能执行 10000 条指令，它也可以编程。利用 MANIAC，科学家解决了原子能研究中的大量数学问题，但是科学家并不马上用 MANIAC 来设计核弹，而是先试验一下这台机器，因此首先做的事情之一就是编写一个下棋的程序。1950 年，C.Shannon 发表了两篇有关计算机博弈的奠基性文章——Programming A Computer for Playing Chess 和 A Chess-playing Machine，它标志着机器博弈研究领域的重大突破。1951 年，A.Turing 完成了一个叫做 Turochamp 的国际象棋程序，这个程序还不能在已有的计算机上运行。1956 年，Los Alamo 实验室的研究小组研制了一个真正能够在 MANIAC-I 机器上运行的程序(不过这个程序对棋盘、棋子、规则都进行了简化)。1957 年，Bernstein 利用深度优先搜索策略，每层选七种着法展开对局树，搜索四层，他的程序在 IBM704 机器上操作，能在标准棋盘上下出合理的着法，是第一个完整的计算机国际象棋程序[12]。实际上，在计算机博弈的研究中，影响最大、研究时间最长、投入研究精力最多的博弈项目就是国际象棋，它也成为计算机博弈发展的主线。到 1967 年，计算机界著名的美国 MIT 实验室 Greenblatt 等人在 PDP-6 机器上开发了 Mac Hack VI 程序，并参加了在麻省理工大学举办的国际象棋锦标赛，创下了计算机正式击败人类的记录。从 1970 年起，ACM(Association for Computing Machinery)开始举办每年一度的全美计算机国际象棋大赛，在 1974 年该比赛改为三年一度的世界计算机国际象棋大赛。1975 年，计算机科学家肯·汤普森 (Ken Thompson)和贝尔实验室的同事一起决定建造一台专门用途的机器，使用了价值大约 20，000 美元的几百个芯片。这台机器被叫做“尤物”(belle)，虽然它只会下国际象

棋，但是它能够每秒搜索大约 18 万个局面（而当时的超级电脑只能搜索 5000 个）。1997 年深蓝在“回敬赛”中战胜棋王卡斯帕罗夫，标志者计算机国际象棋博弈达到了一个新的里程碑。1989 年，第一届计算机奥林匹克大赛在英国伦敦正式揭幕，从此，计算机博弈在计算机界和人工智能研究领域的影响日渐广泛而深远。

20 世纪 70 年代末，台湾学者开始了中国象棋计算机博弈的研究工作，并在八十年代初开发出了苹果机（6502）和 IBM PC 机（8088）象棋程序。1983 年，南开大学开发残局程序，并在随后的几年里开发了全局程序。但是，由于中国象棋在计算机实现方面比国际象棋更加复杂，而且中国象棋博弈技术研究落后于国际象棋，所以，中国象棋软件还远未达到世界冠军水平。进入本世纪后，以东北大学徐心和教授为首的研究团队，通过短短的几年时间，开发出的“齐天大圣”，在 2005 年国内第一届机器博弈锦标赛中，打败了国内冠军，一时传为佳话。由于中国象棋冠军代表了该棋种的世界水平，因此，这标志着国际中国象棋计算机博弈水平达到世界水平。在中国象棋计算机博弈的带领下，国内已经举办了 3 届国际、国内比赛。作者在 2007 年 10 月，作为自愿者，参与了在重庆工学院新校区举办的第二届机器博弈锦标赛的服务性工作，2009 年 10 月，替导师行使领队职责，带领本科生队伍参加了在北京举办的计算机博弈国际比赛，并取得第 4 名的优异成绩。总体来看，近些年来，计算机博弈在国内得到长足发展，中国象棋的发展成熟相对好些，而六子棋项目普及程度要差许多，参与六子棋计算机博弈项目的队伍和社会普及程度都相对较差，所以关于六子棋计算机博弈的研究文献也相对较少。

1.3.3 智能算法的研究现状

智能算法是人们受自然（生物界）规律的启迪，根据其原理，通过模仿来求解问题的算法。从自然界得到启迪，模仿其结构进行发明创造，这就是仿生学。上世纪 60 年代以后，通过研究和模拟生物进化的过程，人们开发了一些求解复杂优化问题的有效算法，并受到更多学者的关注。智能算法包括人工神经网络算法、进化计算、群智能算法等。其中进化计算[13-17](Evolutionary Computation，EC)就是由此而发展起来的一种启发式算法，它包括遗传算法(Genetic Algorithms，GA)、进化规划(Evolutionary Programming，EP)、进化策略(Evolutionary Strategies，ES)和遗传编程(Genetic Programming，GP)等主要进化理论。进化计算的具体实现方法与形式称为进化算法（Evolutionary Algorithm，EA），虽然进化算法具有多个代表性的重要分支，但是它们的进化形式和原理却不尽相同。

群智能算法（Swarm Intelligence Algorithm）是最近十余年研究发展起来的一种新型的仿生类进化算法，并已经成为许多学者研究关注的焦点。群智能(Swarm Intelligence)是一种由若干无智能或低智能的个体通过任何形式的聚集协同而表现出

的较高的智能。生物学家观察探索发现，群居昆虫群体可以看作一个分布式系统，虽然系统中的每个个体都非常简单且智能较低，但是整个系统却呈现出一种高度结构化的群体组织，使个体能协同工作，完成一些远远超出单个个体能力负荷的复杂工作。群智能算法就是模拟自然界生物群体的这种行为而构造的随机搜索方法。目前，该研究领域有两种主要算法：蚁群优化算法(Ant Colony Optimization，ACO)[18]和微粒群算法(Particle Swarm Optimization，PSO)[19]。群智能算法具有较强的鲁棒性，采用分布式计算机制，算法实现简单，在函数优化、数据挖掘、组合优化、网络路由、机器人路径规划、无线传感器网络性能优化等领域获得了广泛的应用，并取得了较好的效果。已有研究结果表明群智能算法为解决许多应用问题提供了新的途径和方法。

根据六子棋计算机博弈的特点，本文将把 GA、PSO 应用于六子棋计算机博弈项目之中。

1.4 本文研究的主要内容

本课题的支撑项目来自于重庆市科委基金项目“人工生命体行为与行为选择的进化研究”，最终目标是构造能够在物理棋盘上与人类进行实物对弈的博弈机器人。因为在人工智能研究中，仿人、仿生是一种重要的研究方法，因此，本文也采用这种方法，并模仿人类的弈棋过程，将博弈机器人划分为“大脑”、“视觉”、“记忆”、“控制”等 4 个部分，而本文的主要工作是设计“大脑”，构造智能决策系统，通过智能算法的应用研究来提高博弈机器人智能水平。

要实现模拟人类博弈过程，首先需要解决棋局在机器内的表示问题。根据前人对基于棋形的棋局局面表示方法的研究，发现基于棋形的表示方法对棋形的判断和统计相对较为困难，而错误的判断和统计将导致博弈机器人下一步行动选择的错误。本文受到其他研究人员的启发，规范了基于“路”的局面表示方法。通过“路”的思想，提高对局面进行统计和判断的准确率，同时，简化评估函数。

另外，针对评估函数中，凭人工经验难以达到全局最优的问题，本文将采用智能算法来对参数进行寻优。

因此，本文所做的工作主要包括以下几个方面：

1、引入并规范了“路”的思想；

2、对评估函数进行了改进；

3、由于本系统的实现，是在导师所带领的团队搭建的博弈平台上实现的，因此，需要对现成的博弈平台进行修改，以方便采用智能算法进行优化；

4、将遗传算法引入评估函数的参数优化；

5、将微粒群算法引入评估函数的参数优化。

2 六子棋机器博弈平台的搭建

2.1 背景

2.1.1 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：

• 玩家：和五子棋、围棋相同，有黑白两方，分别持黑子和白子，黑先。

• 玩法：除了第一次黑方下一颗子外，之后黑白双方轮流每次各下两子，直的、横的、斜的连成 6 子（或以上）者获胜。

• 若全部棋盘填满仍未分出胜负，则为和局。

• 没有禁手；例如长连仍算赢。

在连珠棋规则或国际五子棋规则中，有其他额外的禁手及开局规则。六子棋中则没有。

• 棋盘：因为公平性不是问题，棋盘是可以任意地大，甚至是可以无限大的，然而为了让游戏可实质地来玩，目前棋盘采用围棋的十九路棋盘。

2.1.2 六子棋的复杂度

对于机器博弈来说，一般采用 state-space 复杂度（The state-space complexity of a game is the number of legal game positions reachable from the initial position of the game）和 game-tree 复杂度（The game-tree complexity of a game is the number of leaf nodes in the smallest full-width decision tree that establishes the value of the initial position.）[24]来衡量某种博弈游戏的复杂程度。由于六子棋具有很好的公平性，因此，理论上棋盘是可以任意大，甚至是无限大的。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$(300*300/2)^{30} \sim 10^{140}$，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$(300*300/2)^{40} \sim 10^{188}$，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。事实上，六子棋的这个 game-tree 复杂度已接近于大众认知能力的极限。当然，随着高速计算机的研制和发展，计算机的计算能力必将大幅提升，从而将极大限度地提高计算机博弈能力，这是后话。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 2.1　常见棋类状态空间和博弈树复杂度比较表

State-space complexities and game-tree complexities of various games

Id.	Game	State-space compl.	Game-tree compl.	Reference
1	Awari	10^{12}	10^{32}	[3,7]
2	Checkers	10^{21}	10^{31}	[7,94]
3	Chess	10^{46}	10^{123}	[7,29]
4	Chinese Chess	10^{48}	10^{150}	[7,113]
5	Connect-Four	10^{14}	10^{21}	[2,7]
6	Dakon-6	10^{15}	10^{33}	[62]
7	Domineering (8×8)	10^{15}	10^{27}	[20]
8	Draughts	10^{30}	10^{54}	[7]
9	Go (19×19)	10^{172}	10^{360}	[7]
10	Go-Moku (15×15)	10^{105}	10^{70}	[7]
11	Hex (11×11)	10^{57}	10^{98}	[90]
12	Kalah(6,4)	10^{13}	10^{18}	[62]
13	Nine Men's Morris	10^{10}	10^{50}	[7,44]
14	Othello	10^{28}	10^{58}	[7]
15	Pentominoes	10^{12}	10^{18}	[85]
16	Qubic	10^{30}	10^{34}	[7]
17	Renju (15×15)	10^{105}	10^{70}	[7]
18	Shogi	10^{71}	10^{226}	[76]
	(Source: [Herik et al 2002] [Wu et al 2006])			
	Connect6	10^{172}	10^{140}–10^{188}	

2.1.3　六子棋的发展展望

机器博弈作为一个竞技项目，具有无穷的魅力，世界各国每年都要举办许多机器博弈比赛。但是就目前而言，国内外都很少进行博弈机器人的研究，一般都是编写计算机博弈软件，如国内著名的计算机围棋软件“手谈”，曾经在世界计算机围棋比赛中多次获得第一。

事实上，博弈机器人的研制具有重要的意义，它不仅涉及到计算机软件、智能机器人研究领域的知识，同时也涉及到图像识别、图像矫正、机器视觉、自动控制以及智能算法等一系列学科。因此，六子棋以后的发展，必将提升到一个新的高度，而不仅仅只是一套软件，而是向六子棋博弈机器人的方向发展。六子棋博弈机器人的基本结构应该包括机器人大脑（博弈算法和记忆存储部分）、视觉部分、控制部分和机械部分，如图 2.1 所示。事实上，博弈机器人就是一个人工生命体，博弈机器人在下棋过程中，对落子位置的选择、进攻和防守之间的选择、如何进攻或者如何防守的选择，实际上就是一个人工生命体行为选择的过程，本文主要针对博弈过程中的行为选择问题，研究其中智能算法及其应用问题。

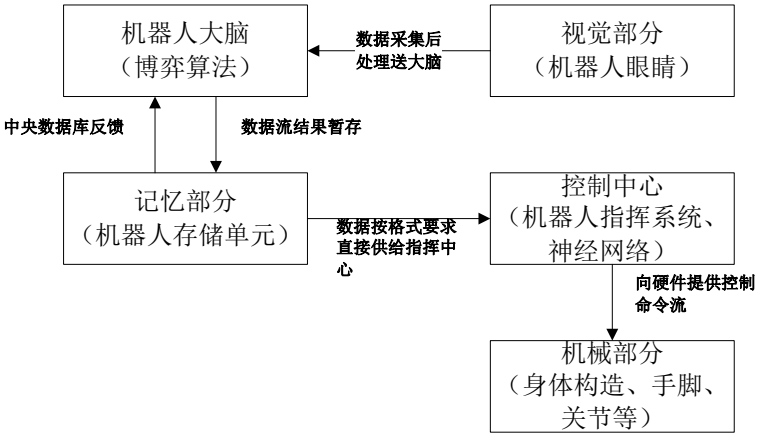


图 2.1　博弈机器人的体系结构

2.2　人-机界面

六子棋博弈属于完全信息博弈。本文的主要工作是构造机器人的“大脑”，即机器人的思维过程，其他诸如视觉、控制、机械等部分不属于本文研究范围。作为一个人-机或机-机博弈系统，需要一个博弈双方的信息交换界面，并通过这个界面实现双方的信息交换和行棋过程控制。

人-机界面是用户和计算机系统交互的纽带。一个优秀的用户界面应该简洁美观大方，色调搭配合理；同时要方便用户使用和理解其用途；最后要便于扩展功能。本系统主界面的色调搭配采用真实环境中常采用的浅黄色木质棋盘，给人以强烈的真实感，加上简单菜单和右面的面板进行导行，使整个界面显得美观大方，浅显易懂，便于操作，另外每次鼠标移动后所产生的小矩形，可以提前提示玩家所要走棋的具体位置，从而更加的人性化，参见图 2.2。

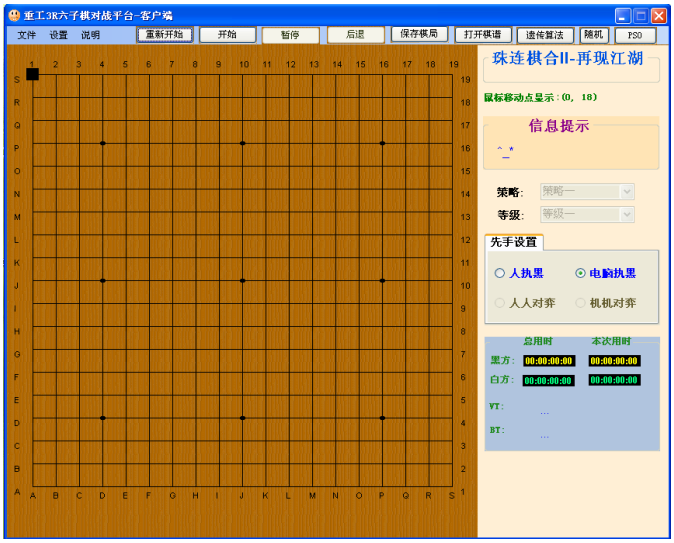


图 2.2　人-机界面

2.3 棋盘表示与状态分析

人类在下棋过程中，首先需要通过眼睛观察对手的落子位置，观察当前的局面，而在不完全信息博弈过程中，玩家甚至还要观察对手的表情来对当前的局面进行评估。同样的，作为一部机器博弈机器人，首先需要实现的是，像人类一样把当前的棋局以及对手的落子位置等相关信息以一种数据结构存储在“脑子”里。

根据机器博弈的发展现状来看，棋盘的表示方法主要有比特表示（位棋盘）[20]、矩阵表示以及用哈希表来表示。位棋盘和矩阵表示这两种表示方法存在着一个非常明显的缺点就是要了解棋盘状态，总是需要遍历整个棋盘的所有落子点，相对来说比用哈希表来表示要费时和复杂。因此，本文将采用哈希表来表示棋盘，用哈稀表中的键值对分别表示棋盘上棋子的坐标和棋子的颜色，这样特定的 key 就对应了一个 value 值。例如：如果用 true 表示白子，false 表示黑子，那么可以用 hashtable[point[9, 10], true]来表示棋盘[9, 10]处有一粒白子，用 hashtable[point[9，9]，false]来表示棋盘[9, 9]处有一粒黑子，如果哈希表中不含有[9, 9]这个键，则说明该位子无子。

哈希表最大的优点，就是把数据的存储和查找消耗的时间大大降低，几乎可以看成是常数时间，而代价仅仅是消耗比较多的内存。在当前计算机硬件配置越来越高的情况下，用空间换时间的做法是值得的，这一点对时间要求本来就比较高的机器博弈系统更为显得重要。这样表示可以很方便的的找到棋盘上面的棋子状态，只需要遍历哈稀表，而不需要遍历整个棋盘，节约了时间。例如：表 2.2 就对应了图 2.3 所示的棋局状态信息，其中 point 数组表示棋盘上的坐标。

表 2.2　某一时刻哈稀表中健值对应情况

key	value
point[10，8]	False
point[10，9]	True
point[10，10]	False
point[11，7]	False
point[11，8]	False
point[11，9]	False
point[11，10]	True
point[12，6]	True
point[12，8]	True
point[12，10]	True
point[13，9]	True

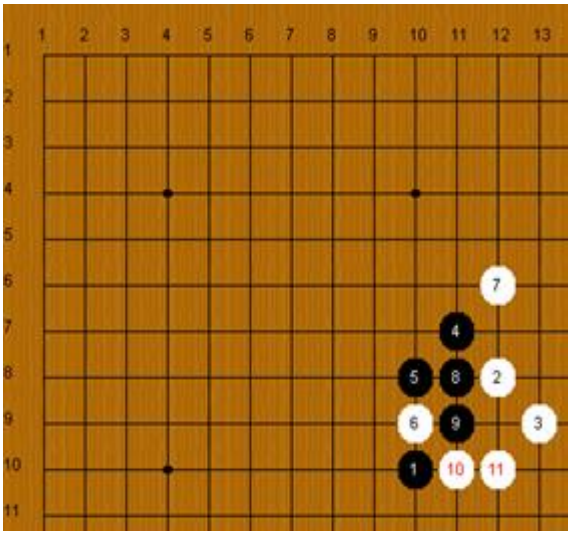


图 2.3　某一时刻棋局的状态信息

2.4 走法产生

2.4.1 如何产生

当人类通过观察当前的棋局，并将当前的局面反映到大脑以后，接下来人类棋手将进行思考，通过自己的经验、当前的局面以及对手落子位置的意图进行揣摩、判断和分析，最后选择一种自己认为最优的走法。那么，博弈机器人在将当前局面的信息收集起来保存到自己的“大脑”以后，下一步同样要进行“思考”。走法产生是指将一个局面所有合理的走法罗列出来的那一部分程序，也就是用来告诉其他部分下一步可以往哪里走的模块。各种棋类随规则的不同，走法产生的复杂程度也有很大的区别。对于六子棋的棋盘上的任意空白点都是合法的下一步，而在 19 路棋盘上，空白的点是相当多的。因此，在六子棋博弈过程中，走法的产生需要与博弈树的剪枝搜索技术相结合。图 2.4 展示了走法产生与博弈树展开而产生棋局状态演化的过程。

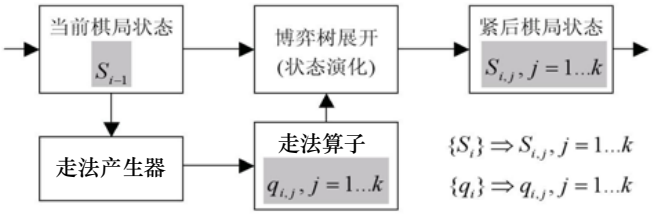


图 2.4 棋局状态伴随着法生成演化过程

2.4.2 逐个生成 VS 全部生成

走法产生往往伴随着搜索的进行。对于某一局面的所有直接后继，可以有两种选择：一种是一次产生一种走法然后搜索之；另一种是一次产生其所有走法然后搜索之。由于在对博弈树搜索的过程中会用到剪枝算法，对一个局面的某一走法搜索之后往往可能不再需要搜索其他后继，一次产生一种走法看起来似乎更有效率。但是，由于剪枝算法的剪枝效率很大程度上依赖于节点的排列顺序，往往一次产生所有节点，然后以某种方法调整其排列顺序会使搜索效率大大提高。

所以，在实际使用中，绝大部分程序都是一次产生一个局面的全部走法，然后调整其搜索顺序。

2.5 博弈搜索引擎

2.5.1 博弈树

几乎所有的棋类问题，都可以用博弈树来描述。博弈树是把计算机和用户所有可

能的走法和局面罗列出来的一颗树[21]。黑白双方交替地按合理走法把树展开，树的每一个节点都表示某一个特定局面。根节点表示的是当前需要计算的局面，中间节点表示的是对弈过程中根据当前局面可能衍生出的某一局面，叶子节点是树的最底端，表示可以推导的局面。叶子节点和根节点之间的最大距离，称为搜索深度。整个博弈树描述的是从当前局面出发，包含所有可能的对弈过程的搜索树。六子棋计算机博弈问题也就转化为寻求最佳路径的问题[22]。

图 2.5 是一棵博弈树的图示。除了极少数非常简单的棋类游戏以外，大多数棋类游戏（六子棋）都不可能建立整棵博弈树来进行搜索。假设博弈树的分枝数为 b，深度为 d，则有 $1 + b + b^2 + b^3 + \bullet\bullet\bullet + b^d = (b^{d+1} - b) / (b - 1)$个结点[23]，以六子棋的复杂度来看，即使生成博弈树的一个节点仅需 10^{-8} 秒，生成这棵树也要 10^{80} 年以上，这样显然是不切实际的。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，但是并不是把所有可能的走法都想到，那些明显不利于自己的落子位置，人类棋手将会直接将它过滤掉，不予考虑。那么博弈机器人同样可以做到，这就是博弈树的搜索算法和剪枝技术。

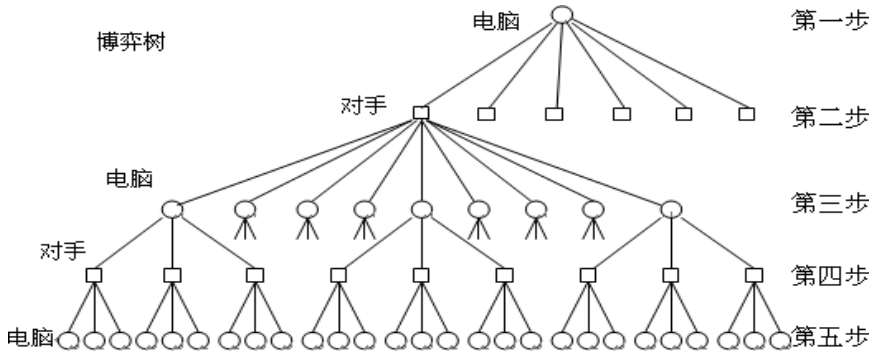


图 2.5　博弈树

2.5.2　极大极小搜索算法

香侬(Claude Shannon)教授早在 1950 年首先提出了极大-极小算法（Minimax Algorithm），从而奠定了计算机博弈的理论基础[23-27]。

博弈树搜索属于对抗性搜索 (Adversarial search)。在博弈过程中，任何一方都希望自己取得胜利。在某一方进行博弈策略选择时，他总是挑选对自己最为有利而对对方最为不利的那个行动方案。因此，某一方在走棋时必须遵守如下原则：考虑到对自己最不利的情况，从最坏的可能中选择最好的；并假定对手不会犯错误，即对手总是选择对他最有利的方法走，所以不能采取任何冒险行动。这个原则就叫极小极大原则，用该原则指导搜索的算法就叫作极小极大值算法[28-30]。

极大极小搜索策略是考虑双方在对弈若干步后，从可能的着法之中选择一个相对较好的着法，即在有限的搜索范围内求解。假设博弈双方分别为 *MAX* 和 *MIN*，要为

MAX 找出最佳的着法，并设 *MAX* 为先手方。因此，偶数层的节点对应于 *MAX* 的后续着法节点，称为 *MIN* 节点；其余的节点为 *MIN* 的后续着法节点，称为 *MAX* 节点。通常用层数表示博弈树的搜索深度，以此来量化向前预测 *MAX* 和 *MIN* 交替运动的回合数。对六子棋机器博弈，由于比赛时间限制以及合理性，不可能实现完全搜索，而通常采用在有限范围内搜索的方法，因此，寻求能缩减搜索深度的策略就尤其重要，而启发式搜索是正是这样的一种搜索策略。

启发式搜索的关键是要确定下一个着法所对应的节点位置。通常先要估算出各节点的重要性，以便搜索算法能以较快速度选择高效的搜索节点，节省时间。为此，需要定义一个静态估值函数 f，并规定：若有利于 MAX 方 f 取正值；若有利于 MIN 方时则 f 取负值；势均力敌的 f 取 0。图 2.6 是一个搜索两步棋的例子，假设图中节点旁给出的数字是用静态估值函数 f 计算得到的，其他节点就不需要用 f 估值，而用倒推方法求得，如 A、B、C 是 MIN 的着法节点，MAX 应考虑最坏情况，其估值分别取该节点 f 估值的最小者， 但 S 是 MAX 的着法节点，可考虑最好情况，因此 f 估值应取 A、B、C 值中最大者，这样得 f(S)= - 4，依此类推确定出相对较优的着法应是 B，因为从 B 出发，对手不可能产生比 f(S)=-2 更差的效果。事实上，如果比赛时间许可的话，还可以进行更深的搜索，从而得到更精确的 f 值，产生更佳的 MAX 着法。综上所述，该搜索算法步骤如下：

① 生成整个博弈树，即扩展树的每个节点；

② 用静态估值函数 f 对每个叶节点进行估值，得出每个终节点的评价值；

③ 用终节点的估值得到其搜索树上一层节点的估值；

④ 重复③ 过程在 MAX 层取其分支的最大值，MIN 层取其分支的最小值，一直回溯到根结点。

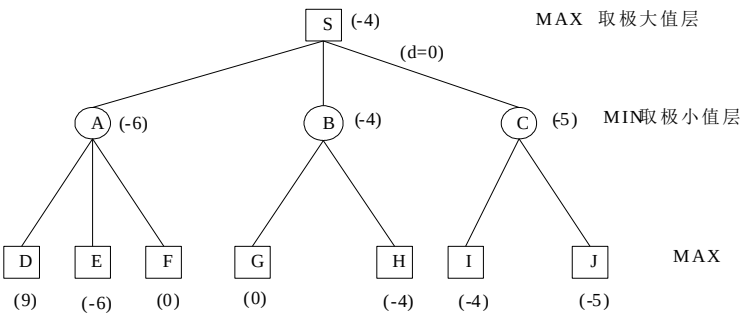


图 2.6 f(s)的求值过程

2.5.3 负极大值搜索

在使用极小极大值算法进行搜索时，往往要考虑是使用广度优先 (Breadth First Search)还是深度优先 (Deep First Search)，广度优先通常因为要存放大量生成的局面

而要使用较大内存，因此选择深度优先搜索。深度优先搜索任何时候只保存与搜索层数相等的节点个数。

冯·诺依曼（John Von Neumann）最早提出极小极大值算法的思想，他于 1928 年提出“二人零和博弈的极小极大解决方法（The minmax solution for a two- person zero-sun game）。而后的大约 50 年里极小极大值算法没有什么变化和改进，直到 1975 年 Knuth 和 Moore 提出了负极大值（Negamax）方法[31]，该方法可以兼顾双方的立场，消除与节点和或节点的差别，以统一的方式进行处理。

负极大值方法的原理是：对于没有后代或搜索层次达到极限的节点，仍采用静态估值函数来对它们的进行估值，但要站在该节点走棋方的立场上来估值；对于其他节点，均令父节点的估计值为诸子节点的估计值的负数的极大值[32]。它是极小极大算法一种更好的表示形式。

2.5.4 alpha-beta 裁减

Bruno 在 1963 年首先提出了 α - β 算法[22]，1975 年 Knuth 和 Moore 给出了 α - β 算法的数学正确性证明[31]。可以说 α - β 剪枝算法是博弈树搜索技术的基础，是到目前为止使用最广泛的博弈搜索算法之一。但 α - β 剪枝也有其弱点，需要通过相应的策略来改进、增强 α - β 剪枝算法。

极大极小搜索算法把搜索树的生成和估值两个过程分开进行，即先生成全部搜索树，然后再进行节点静态估值和倒推值来计算，这是存在计算冗余的。如图 2.7 为一棵极大极小树的片段，假设 A 为极大值节点，B 的估值为 2，节点 D 的估值为 0，由此可以判断节点 A 值≥2，而节点 C 值≤0，所以，实际计算就没有必要估计其它节点的值，这样就可以将其 D 节点的后继节点裁减掉，这就是 α 减枝。同样，图 2.8 也是一棵极大极小树的片段，假设 A 为极小值节点，B 的估值为-2，节点 D 的估值为 0，由此可判断节点 A 值≤2，节点 C 值≥0，所以，再估计其它节点的值就没有必要了，从而将 D 节点的后继兄弟节点裁减掉，这就是 β 减枝[31]。图 2.9 是一个深度为 4 的博弈树，其中带括号的数字表示求静态估值及倒推值过程的次序编号。该图展现了 α-β 裁减的细节。

可以证明，当搜索树的深度为 n，分枝数为 B 时，如果不采取 α-β 技术，搜索树的端节点数 $N_n=B^n$；若使用 α-β 技术，理想条件下生成的端节点数最少有[33]：

$$N_n=B^{n/2}-1 \qquad (n \text{ 为偶数})$$

$$N_n=B^{(n+1)/2}+B^{(n-1)/2}-1 \qquad (n \text{ 为奇数})$$

这就表明了，在一般的条件下，使用 α-β 裁剪搜索技术，在同样的资源限制下可以向前考虑更多的走步数，这样将带来更大的优势。

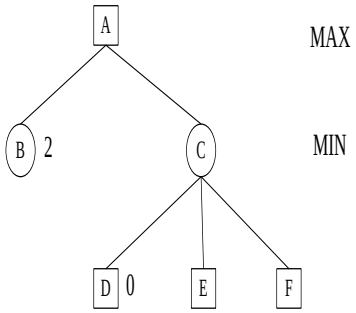


图 2.7 α 裁剪图

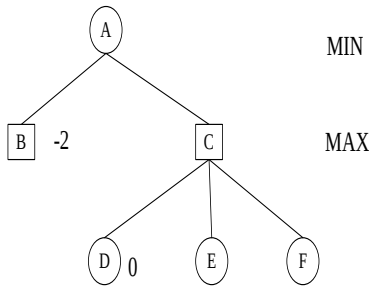


图 2.8 β 裁剪图

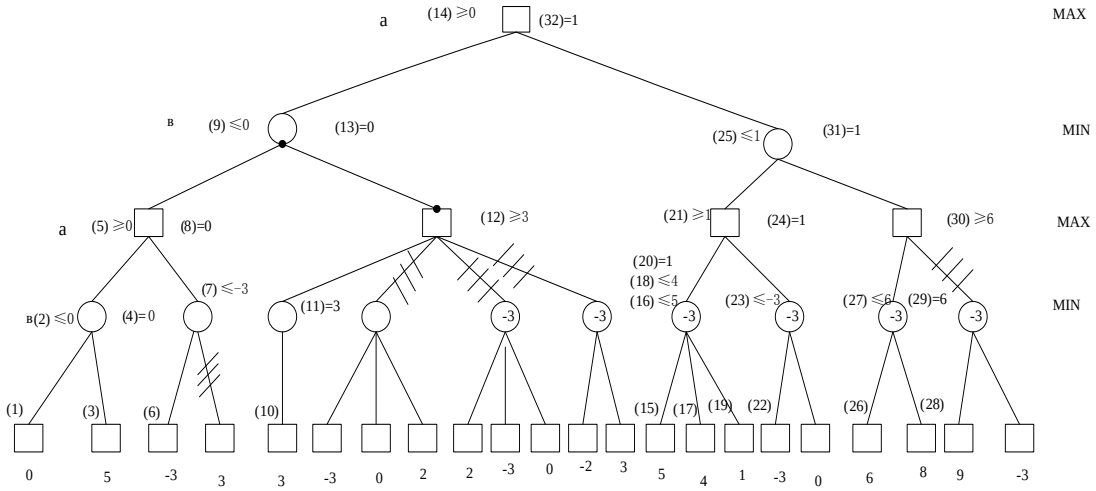


图 2.9 α-β 裁剪过程图

2.6 棋局评估函数

对于人类棋手来说，人们往往会根据当前的棋局，在脑子里往后推演若干步，考虑可能出现的局面，然后根据可能出现的局面对自己的利弊来选择下一步合适的落子位置。显然，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，这是一个模糊的而又最需要智能来判断的。同样对于博弈机器人来说，也需要这样的一个思维过程，这就是棋局的评估函数[34-35]。

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。计算机虽然计算能力极强，但是由于现在的计算机是数字化的，一切计算都以 0、1 代码为基础，对模糊的内容，现在的计算机赶不上人类的计算能力，因此在做比较时往往要有量化指标才能完成对比。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”。

事实上，博弈机器人仍然是通过数字化的计算机来实现的，所以不管多么复杂的评估函数，都必须表示为一个多项式，以此来把局面的好坏进行量化。评估函数一般

来说必须包括 5 个方面的要素[36]，分别是：

1、固定子力值：例如在中国象棋中，车、马、炮根据其作用大小给定固定子力值。但对于六子棋而言，子力没有作用，因为六子棋的棋子不像象棋中的车马炮有各自走法的特殊性。

2、棋子位置值：对于同样的一个棋子，处在棋盘的不同位置上其价值也是不相同的。比如，中国象棋中的兵，在开局阶段，它的作用并不是很明显，位置值比较低，但是如果在中局或者残局的时候，兵过河，那么它越靠近九宫，它的位置值应该越大；在五子棋和六子棋中，位于棋盘中心的棋子位置值比其他位置的大。

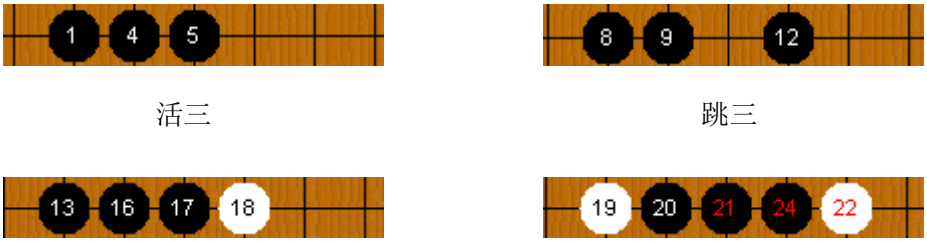
3、棋子灵活度值：在象棋中，车可能被己方棋子挡住路线，马可能被撇脚。这时要考虑灵活度。

4、威胁与保护值：例如己方的车保护马不被对方吃到，或己方的炮下一步可以吃掉对方的子，这就是威胁和保护。

5、动态调整值。

每一方面的值又是由许多参数值构成的。将这些值线性地组合在一起便得到最终的评估值。

根据分析，六子棋的棋形可以划分成 19 种：连六，长连，活五，眠五，死五，跳五，活四，眠四，死四，跳四，活三，眠三，死三，跳三，活二，眠二，死二，活一，死一，如图 2.10 所示。先对棋局上的局面进行分析，获得局面上各种棋形的数量，根据上述棋形对应的分值，加上对空棋盘每个空位的初始估值，进行线性组合，即得到局面的分值。



活三　　跳三

眠三　　死三

图 2.10　六子棋的棋形

2.7　六子棋机器博弈策略框架

根据上面分析，可以给出图 2.11 所示的整个机器博弈机器人“脑子”的系统策略部分的框架简图。

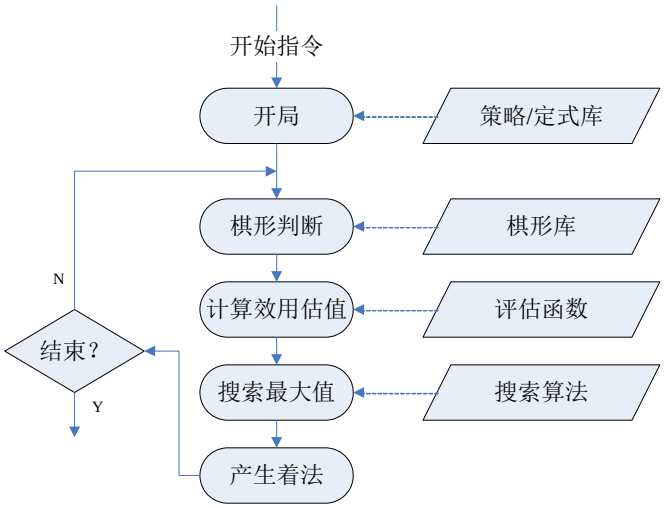


图 2.11 六子棋博弈系统的博弈流程图

3 基于“路”的六子棋机器博弈策略

3.1 完全信息博弈与不完全信息博弈

机器博弈是博弈论的应用之一。所谓博弈论，简单来说，就是研究多人谋略和决策问题的理论。显然，一个博弈问题必须至少有两个参与博弈的主体，每一个主体都希望尽可能提高自己的利益所得。由于参与博弈的主体是相辅相成、相互制约而又互相竞争的，因此，每一个主体都必须捕捉各种信息，审时度势，在适当的时候选择一个最佳的行动。

在博弈论中有这样几个定义：

⑴ 参与者。参与者是一个博弈中的决策主体。在六子棋博弈中，参与者就是指下棋的选手或计算机。

⑵ 信息。信息时参与者在博弈过程中能了解到和观察到的知识。信息对参与者至关重要，只有准确的把握信息时自己做出最佳选择的基础。

⑶ 战略。战略是参与者如何对其他参与者的行动作出反应的行动规则，它规定参与者在什么时候该选择什么行动。

⑷ 收益。收益是在一个特定的战略组合下参与者得到的确定的效用或期望效用。

根据参与者对博弈信息的了解程度，博弈可以分为完全信息博弈和不完全信息博弈。

3.1.1 完全信息博弈

完全信息博弈是指每一个参与者对自己以及其他参与者的行动，以及各参与者选择的行动组合产生的效益等知识有完全的了解。完全信息博弈又分为完全信息静态博弈和完全信息动态博弈。所谓静态博弈是指博弈的所有参与者同时选择各自的行动，如果选择行动有先后的话，那么后行动者也不知道先行动者采取了什么行动。例如人们熟知的“石头、剪子、布”游戏，就属于静态博弈。而在动态博弈中，各参与者的行动有先后顺序，而且后行动者在自己行动之前能观测到先行动者的行动。六子棋就属于动态博弈。在动态博弈过程中，参与者可以观测到其他参与者的行动，于是可以理性的思考，后行动的参与者将根据先行动的参与者可能采用的战略采取针对性的措施，同时也会受到先行动的参与者行动的制约。在实际生活当中，完全信息动态博弈也得到了广泛的使用，例如斯坦克尔伯格(Stackelberg)双寡头竞争模型、里昂惕夫(Leontief)工会与企业的工资和就业模型、罗冰斯坦(Robinstein)轮流出价的讨价还价模型等等。

在机器博弈项目中，属于完全信息动态博弈的包括：中国象棋，国际象棋，围棋，六子棋等。

3.1.2 不完全信息博弈

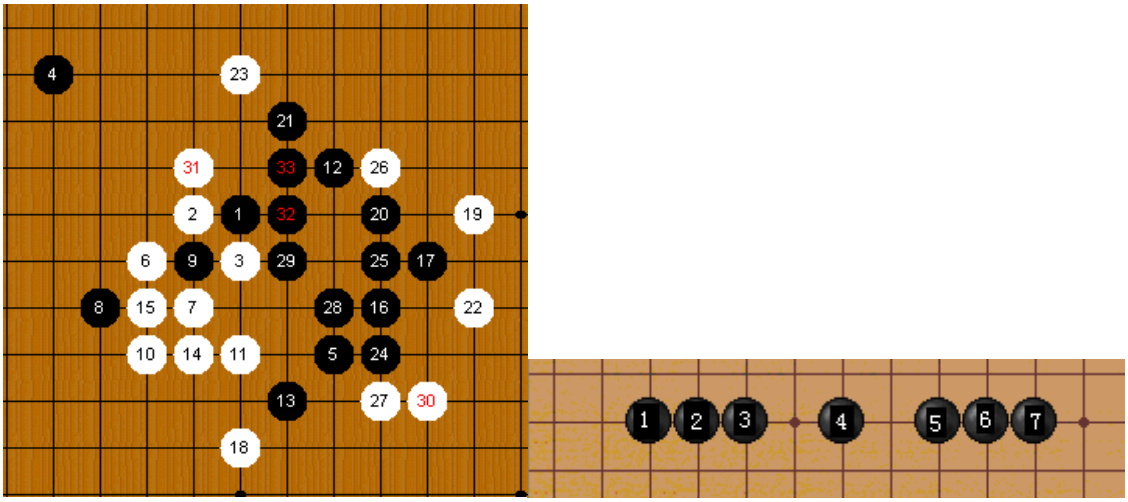
与完全信息博弈相反，所谓不完全信息博弈就是指在博弈过程中，每一位参与人对其他参与人的特征、策略空间及收益函数没有准确的信息，而只有部分信息，参与人并不完全清楚有关博弈的信息。

大多数纸牌游戏是不完全信息博弈。例如，在桥牌里，我们并不知道我们同伴手中的牌，也并不知道坐在左右两位对手手里的牌。在作决策的过程中，我们必须根据手上的牌等已知信息，对其他三位手中的牌作一个估计，并没有准确的信息。在股票交易过程中，买卖股票的人只知道自己买入股票的价位，股票到目前为止的走势，大盘情况以及该股的业绩等等，但是并不清楚这支股票是会涨还是会跌，涨会涨到什么价位，跌又会跌到什么价位，即使有资深的股票分析师进行了分析，也不能百分之百确定。当你与一位新同事打交道时，你并不知道他的性格特征。事实上，即使是你的老同事，也很难说你对他有完全的了解；当你在市场上购物时，你并不知道卖主愿意脱手的最低价格是多少，或买主愿意出的最高价格是多少。如此等等，这样的例子举不胜举。类似上述这些不满足完全信息假设的博弈都是不完全信息博弈。

当然，如果对博弈对手一无所知，那么，也就无从博弈。现实生活中，大多数情况下，虽然对于对手的一些特征不完全了解，但总不至于一无所知。例如，打牌时，虽然不知道对手具体拿什么牌，但根据自己的牌，还是可以对手的牌有一个估计的，而且，随着牌局的展开，人们会不断改变这些估计。这些估计，可以用数学上的“概率分布”来表示。

3.2 基于棋形的缺陷

在对局面进行估值的时候，通常的做法是通过给定各种棋形一个能够反映该棋形利弊的分值，然后分析局面中各种棋形的个数，最后通过线性组合来获得局面的分值。事实上，目前在国内外，大部分六子棋博弈系统都是采用常规的以棋形分类为基础的结构。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。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。



(a) (b)

图 3.1 交叉棋形

在图 3.1(a)中，黑白棋子交错分布，各种棋形的组合又错综复杂，难以分。又如在图 3.1(b)中，黑子 1、2、3 可以划分为连三，黑子 1、2、3、4 又可以划分为跳四，黑子 1、2、3、4、5 又可以划分为跳五，同样的，黑子 5、6、7，黑子 4、5、6、7，黑子 3、4、5、6、7 也可以划分成不同的棋形，像这样多种棋形交叉的情况很难划分。因此，基于棋形的思想对于准确判断棋形的要求非常高，是否能够有效准确地判断棋形成为衡量系统棋力的一个决定性因素。而如何有效和准确的判断、搜索目标棋形，已经成为机器博弈系统开发者的一个难题。

3.3 “路”的定义

在博弈当中，对信息的把握尤为重要，特别是对于完全信息博弈来说，如果不能准确把握对手的信息，而自己的信息又完全暴露在对方的视野下，显然对自己是相当不利的。而基于棋形的策略难以准确评估当前的信息，为破解这个难题，受到其他学者的启发，本文规范了“路”的思想，以此来构建六子棋计算机博弈的决策模型。

定义 1 路：在棋盘上连续 6 个点能够连成一条直线，则称为 1“路”。

定义 2 路的颜色：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。路的颜色用 c 表示；

定义 3 路的长度：路上已经含有棋子的个数称为路的长度。用 n 表示；

定义 4 路的有效性：如果某条路可能形成 6 个同色棋子连成一线的终局局面，则该路是有效的。路的有效性用 a 表示；

定义 5 路的分值：该路对当前局面影响大小的反映，用 s 表示。

按照横向、纵向、左斜、右斜 4 个方向的特点和 “路”的定义（图 3.2），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)。

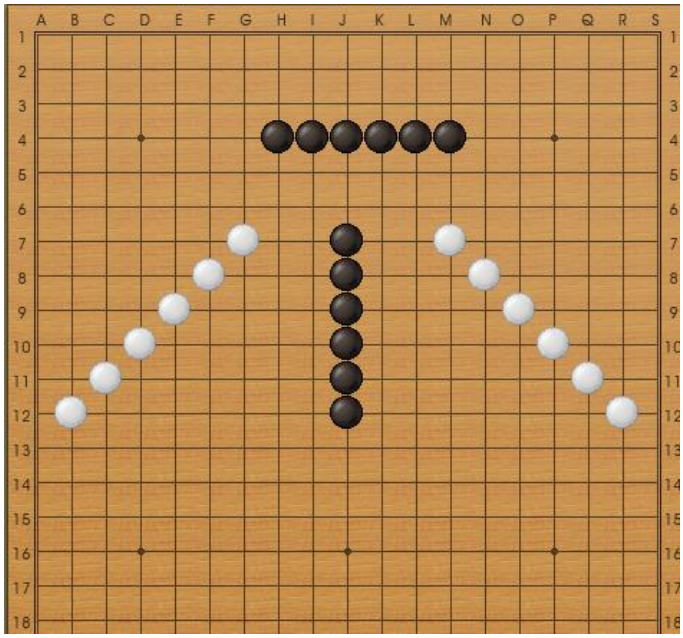


图 3.2 “路”的 4 个方向

用状态空间法来描述，一个“路”R 的状态（state）由一组变量的有序集合来表示，其矢量形式如下：

$$Q=[n,c,a,s]^T$$

n , c ,a , s 表示“路”的 4 个属性，用 N , C , A , S 分别来表示 n , c ,a , s 的值域，则有：

$$n\in N=\{0,1,2,3,4,5,6\}$$

n 表示该条路的长度；

$$c\in C=\{black,white,null\}$$

c 表示该条路的颜色，如果该路上所有棋子都为黑子，则 c = *black*；如果该路上所有棋子都为白色，则 c = *white*；如果该路上没有棋子或同时含有 2 种颜色的棋子，则 c = *null*；

$$a\in A=\{true,\ false\}$$

a 表示该条路的有效性，路的有效性的判断条件是：

$$if \quad (n=0)\cup(c\neq null) \quad then \quad a=true \quad else \quad a=false;$$

当 n = 0 时，表示该“路”上没有棋子，因此任何一方都可以在该路上形成六子相连的胜利局面，即该路是有效的；当 c ≠ null 时，表示该路上只还有白色或黑子的棋子，那么白方或黑方可以在该路上形成六子相连的胜利局面，因此该路是有效的。

$$\begin{aligned} s\in S=\ & \{score\_of\_path[0], score\_of\_path[1], \\ & score\_of\_path[2], score\_of\_path[3], \\ & score\_of\_path[4], score\_of\_path[5], \\ & score\_of\_path[6]\} \end{aligned}$$

s 表示该路的分值，类似于在基于棋形的思想中，我们需要给不同的棋形，如活四、活三，不同的分值，该分值代表了该种棋形对己方的有利程度。在路的思想中，不同路的分值大小由该路上所含有的同色棋子的个数决定，$score\_of\_path[0]$ —— $score\_of\_path[6]$ 分别表示 $n = 0, 1, 2, 3, 4, 5, 6$ 时该路的分值。分值越大，代表这条路的作用越大，对对方的威胁也越大；相反，分值越小，代表这条路对对方的威胁也越小。当某条路为无效路，即该路的状态为 $Q'=[*, *, false, *]$ 时，因为该路已经失效，该路上已经同时存在 2 种颜色的棋子，那么在这条路上就不可能形成 6 颗同色棋子连成一条线的胜利局面了，因此该路对敌方和己方的威胁已经消失，其分值 $s = 0$，即 Q'=[* ， * ， $false$ ， 0]。

采用“路”的思想以后，这样棋形的判断和统计就变得非常的简单，19×19 的棋盘经计算一共可划成 924“路”，那么问题的状态空间可以用 Q 表示，924 条“路”的状态（state）用 q_k 表示，则有

$$Q=\{q \mid q=[n,c,a,s]^T, n\in N; c\in C; a\in A; s\in S\}$$

$q_k \in Q$ 且 $k=[1,924]$

这样可以建立模型如下：

初始棋局：$\forall k\in[1,924], q_k=[0, null, true, score\_of\_path[0]]$

目标棋局：$\exists k\in[1,924], q_k=\{[6,c,true,s] \mid c\in\{black, white\}, s=score\_of\_path[6]\}$

这样，在分析当前局面的时候，避免了对棋形判断的复杂性，取而代之的是简单的对 924 条路进行扫描和统计。例如，1 条路里面有 4 个黑子，则该路的状态为 $[4, black, true, score\_of\_path[4]]$，只需要知道该路的长度以及颜色，无需去判定到底是活四、眠四或者跳四。之所以可以这样简化，是因为“路”巧妙的设计思想。

3.4 基于“路”的评估函数

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。在本文中将建立的评估函数由EvaluateChessStatus(EvealuteColor) 和 CheckChessStatus()两个部分组成：

⑴ CheckChessStatus()为初始化函数。该函数的作用是在估值前先对当前局面进行状态扫描，洞察当前局面中含有 0～6 颗棋子的、本方的有效“路”数，假设用NumberOfMyPath[0]～NumberOfMyPath[6]表示之，而对方的有效“路”数假设用NumberOfEnemyPath[0]～NumberOfEnemyPath[6]表示之。

⑵ EvaluateChessStatus(EvealuteColor)为估值函数。19 路棋盘上总共有 924 条“路”，可以被划分为 7 种状态，即在“路”上分别有 0～6 颗棋子的 7 种状态。假设分别用 ScoreOfPath[0]～ScoreOfPath[6]表示不同状态估值，该值越大，则所对应的“路”的作用就越大。采用“路”思想，行棋中只要关注某路上的棋子数，而不需要关注这些棋子的棋形排列情况。比如，某条路存在 4 颗棋子，它可能是活四或跳四，但采用基于“路”的搜索策略，就不用关心是活四，还是跳四，只关心还差 2 颗棋子分胜负，且在该路上只要对方再下一棋子，该路就不可能再出现连 6 局面，按照路的定义，这一路将失效，自动被排除在搜索范围外。

根据上述思想，对具体棋局的估值 *Score* 可由式（3.1）计算获得。

$$Score=\sum_{i=0}^{6}(NumberOfMyPath[i]*ScoreOfPath[i])-\sum_{i=0}^{6}(NumberOfEnemyPath[i]*ScoreOfPath[i]) \tag{3.1}$$

由式（3.1）可知，$ScoreOfPath[i](i=0,1,\ldots\ldots,6)$ 就是 7 种状态“路”的分量值。因此，评估棋局状态就是分析这些分量值。只有式（3.1）中各参数准确，才能正确反映棋局状态。假设“路”中无棋子，则分量值 ScoreOfPath[0]=0；假设 “路”中存在 6 颗棋子，则分量值 ScoreOfPath[6]=10000；假设“路”中存在 1～5 颗棋子，则分量值 ScoreOfPath[1]～ScoreOfPath[5]应该是多少呢？我们曾采用人工方式赋值，当然，采用经验赋值也是可行的，但主观色彩就太浓。因此，本文将采用智能算法进行离线寻优。

3.5 基于“路”的博弈策略

采用“路”的决策思想，博弈机器人的“脑子”就不需要对复杂的棋形进行统计和区别，而只需要根据“路”来估值，计算出棋盘的状态值。基于“路”的博弈策略

框架如图 3.3 所示。

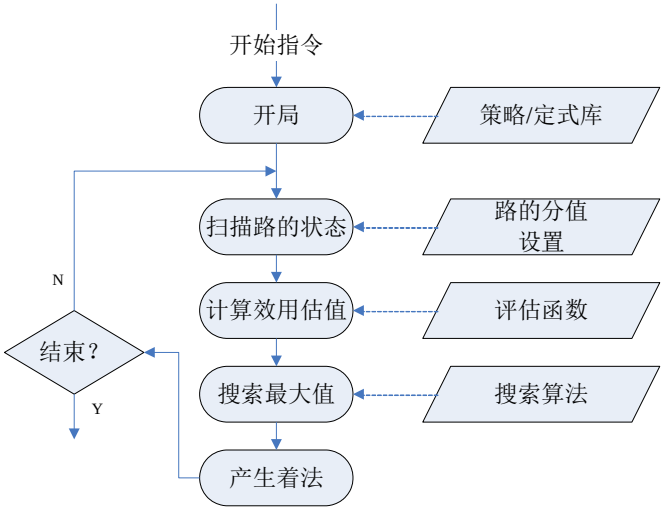


图 3.3　基于路的策略框架

首先计算机接受开始指令，在本系统中是点击界面上的“开始”按钮。这时，计算机首先判断自己是黑方还是白方，如果是黑方，表示计算机是先手，那么计算机根据自己的开局库来进行开局。这些局面往往是事先确定下来的对己方比较有利的开局。开局结束以后，计算机根据对手应对的落子位置，扫描当前局面，对当前有效路的状态进行判断和统计。接着根据有效路的统计结果，通过评估函数和搜索算法相互配合对博弈树进行搜索，获得当前最有利的落子位置，从而产生着法。

4 基于遗传算法评估函数的构造

4.1 评估函数参数优化问题

4.1.1 评估函数参数设置问题

学者已经指出：不管多么复杂的评估函数，都可以表示为一个多项式。在机器博弈中，评估函数一般来说必须包括 5 个方面的要素[36]，分别是固定子力值、棋子位置值、棋子灵活度值、威胁与保护值、动态调整值，将这些值线性地组合在一起得到最终的评估值。在本文所设计的六子棋机器博弈系统当中，路的不同状态需要给定不同的分值，也就是需要设置的参数。这个分值要反映不同路对己方的利弊，分值的大小对该路的优劣程度反映得越准确，评估函数的估值就越准确，错误的估值将会导致落子位置的选择错误。事实上，对于机器博弈来说，设计一个绝对准确的评估函数是不可能的，假设有一个评估函数是准确客观的反映局势的，称这个估值为理想评估函数，我们只可能逼近这个理想评估函数，而不可能到达，这是问题的固有难度所决定的[38]。

4.1.2 评估函数参数设置的 2 种方法

4.1.2.1 手工调整

以前的机器博弈系统通常采用人工参与的方法来调整参数，即利用大量的测试局面，通过人为的观察和经验来手工调整和优化参数。本文中设计的系统曾经也是采用这种方式来设置参数，比如，从经验上可以知道，一个长度为 1 的有效路的价值要比一个长度为 4 的有效路要小，于是给长度为 4 的有效路更大的数值；但是对于六子棋来说，长度为 3 的有效路与长度为 4 的有效路之间，长度为 4 的有效路和长度为 5 的有效路之间到底孰优孰劣？或者它们之间的分值差值到底该设置为多大？这些都需要放到评估函数中反复对弈，然后不断修正参数，找出一组性能较高的参数。

手工调整参数的方法主要一下几种[39]：

① 规格化(Normalize)

如果只是关心评价的顺序，而不怎么关心评价值，那么可以把每一项都乘以同样的常数。这就意味着，对某个特定的模块，例如长度为 1 的有效路的价值，可以硬性设一个值，其他值就可以表示成它们相当于多少个长度为 1 的有效路。这个做法可以减少一个需要设定的参数。

② 约束法（Deduce Constraints）

通过考虑希望计算机做出怎样的判断，就可以确定一些参数。例如在对弈中，可以通过一个子使自己构成一个活四，也可以通过这个子破坏对方的一个活三，这个时候也许选择使自己构成活四的走法更优，但是如果通过这个子可以破坏对方的两个活三的话，也许破坏对方活三的走法更优。这样的条件越多，合适的权重组合就越少。在开始设定权重值的时候，这个方法通常可以得到合适的值，但是在后面仍然需要做一些调整。

(3)交手法 (Hand Tweaking)

这是调整评估函数时非常常用的方法，通过让程序对弈，来找到程序的缺陷，猜测怎样调整参数会让程序更好，然后挑选新的参数。这个方法可以很快得到合理的结果，但是需要对棋类知识有足够的了解，便于根据程序的对弈情况来做分析，从而知道程序的问题在哪里。

手工调整是各种机器博弈系统调整估值参数的主要手段之一，把人类的知识和经验尽量准确客观地“教授”给计算机，是提高棋力的普遍做法，虽然比较费时，但是很有效。通过不断的试验、修改参数值，可以得到不错的结果。但是人类的知识毕竟具有一定的局限性，评估函数也不能包含所有情况，参数之间往往又互相联系，调整某个参数可能解决了某类问题，但又可能给其它问题的解决带来麻烦，在这种情况下很难实现全局下的最优组合。

4.1.2.2 机器自学习

手工调整法需要人为干预，而且往往需要调整者有丰富的棋类知识，这样参数调整的优劣跟调整者的博弈水平有很大的关系，并且当参数很多的时候，很难达到全局最优。利用智能算法来调整参数相对来说更方便，并能更好的达到全局最优。

① 爬山法(hill - climbing)

爬山法[40]通过对参数进行小范围的调整来寻找最优解，一次只能对参数作一点小改变，然后测试博弈程序的性能是否提高了，只有性能有所提高时才采纳这个调整，需要重复很多次。这种方法很慢，并且受初始采样值的限制，很容易陷入局部最优，即评价可能很差，但是任何很小的改变都会使评价更差。

② 模拟退火法(Simulated Annealing)

类似于爬山法，也是对权重做调整来寻优。不同的是如果调整没有达到更优，也随机的有一定几率接受调整，以试图跳出局部最优。这个方法需要给定一些几率，从几率高、梯度大的条件开始，然后逐渐减小。模拟退火法比爬山法更慢，但是最终可能得到比较好的值。

③ 遗传算法(Genetic Algorithms)

爬山法和模拟退火法可以得到一组好的权重，它们是逐渐变化的。遗传算法则不

同，它可以得到几组不同的好的权重，不断增加新的组合跟原来的做比较(取用某组中的某个权重，另一组中的另一个权重，互相交换得到新的)，通过淘汰坏的组合来控制种群的数量。

④ 神经网络(Neural Networks)

神经网络与上面的算法不同是，它更利于构造评估函数，而不是用来选择权重。神经元是阈值(输入权重的和)的函数，第一层神经元输入的关于局面的特征(例如局面中有效路的数量等)就可以构造网络，然后前一层的结果输入到后一层。用这种方法作为评估函数容易实现(只要根据输入的改变来重新计算神经元的输出就可以了)，问题是神经网络中权重的设置问题。除了前面的方法外，针对神经网络还发展出一些方法，例如“即时差分学习”(Temporal Difference Learning)。其基本思想是确定网络何时会做出坏的评价，并且让每个权重增加或减小看是否会评价得更好，这很类似于爬山法。跟其他自学习的方法相比，神经网络的好处在于它不需要很多人类的智慧。但是根据目前我们掌握的情况，根据自己的智慧来做评估函数，要比机器学习做得好，并且做得快。

遗传算法可以同时维护多组较好的参数，通过向其中添加一组新参数，通常是将从几组老参数中选出的值组合在一起作微小的变化，然后同几组老的参数比较孰优孰劣，将最坏的一组去除。遗传算法是从几组参数开始，而不是一组参数，具有很好的全局搜索能力，搜索速度也很快，而且此算法能将搜索重点集中于性能高的部分，能较快地求出最佳参数，鲁棒性也明显优于爬山法和模拟退火法，所以在计算机博弈中最有可能取得成功。

4.2 基本遗传算法

4.2.1 遗传算法的基本思想

遗传算法(Genetic Algorithm , GA) [41] 是近些年发展起来的一种较新的全局优化算法，最先由 Holland 教授提出[42]。它是从代表问题潜在解集的一个初始种群开始的，一个种群则由经过编码的一定数目的个体组成。每个个体实际上是染色体带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体形状的外部表现。初代种群产生之后，按照优胜劣汰适者生存的原则，逐代进化产生出更好的近似解。在进化过程中，根据个体的适应度大小随机挑选个体，并借助于自然遗传学的遗传算子进行组合、交叉和变异，产生出代表新的潜在解集的种群。这一过程将使后生代种群比前代整体适应度更高，末代种群中的最优个体经过解码，可以作为问题的最优解。

4.2.2 遗传算法的基本流程

遗传算法采用了自然进化模型，如选择、交叉、变异、迁移等。计算开始时，随机地初始化含有 N 个个体的初始种群，计算每个个体的适应度，第一代即初始代就产生了。在进化过程中，如果当前种群不满足优化准则，则进化开始产生新一代种群。在产生下一代时，根据个体的适应度采用合适的选择算子选择个体，通过合适的交叉算子对选择的父代进行基因重组产生子代。所有的子代按一定概率变异。然后子代的适应度又被重新计算，子代插入到种群中取代父代，构成新的一代。循环执行这一过程，直到满足优化准则为止。

遗传算法的一般流程如图 4.1 所示：

第 1 步：随机产生一定种群大小的初始种群，每个个体进行染色体的基因编码；

第 2 步：计算个体适应度，判断是否符合优化准则，若符合，输出最佳个体及其代表的最优解，结束计算；否则转向第 3 步；

第 3 步：依据适应度选择再生个体，适应度高的个体被选中的概率高，适应度低的个体可能被淘汰；

第 4 步：按照一定的交叉概率和交叉方法，生成新的个体；

第 5 步：按照一定的变异概率和变异方法，生成新的个体；

第 6 步：由交叉和变异产生新一代的种群，返回到第 2 步。

遗传算法中的优化准则根据问题的不同也有所不同。例如，可以采用以下的准则之一作为判断条件：

① 种群中个体的最大适应度超过预先设定值；

② 种群中个体的平均适应度超过预先设定值；

③ 世代数超过预先设定值。

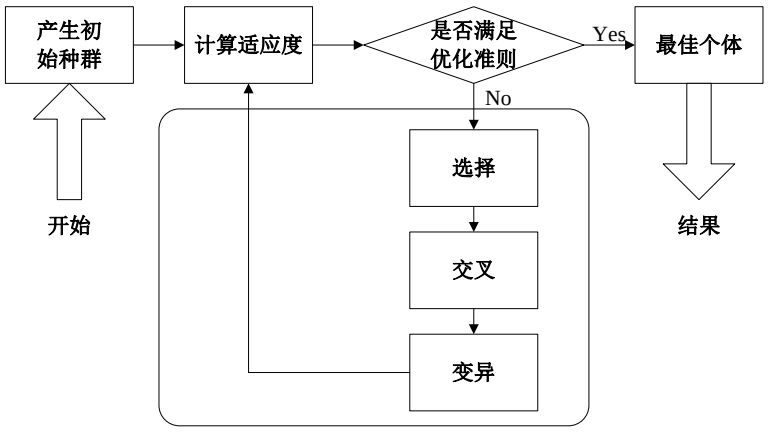


图 4.1 遗传算法的一般流程

4.2.3 遗传算法的特点

传统的优化方法主要有三种：枚举法、启发式算法和搜索算法。随着当前所求问题种类的不同以及问题规模的扩大，需要一种能以有限代价来解决搜索和优化的通用方法。遗传算法正是这样一种算法，它不同于传统的搜索和优化方法。主要区别在于[43]：

① 遗传算法不需要求导或其他辅助知识，而只需要影响搜索方向的目标函数和相应的适应度函数；

②遗传算法强调概率转换规则，而不是确定的转换规则；

③遗传算法可以更加直接地应用；

④遗传算法对给定问题，可以产生许多的潜在解，最终选择可以由使用者确定。

4.3 改进的遗传算法

自从 1975 年 J.H.Holland 系统地提出遗传算法的完整结构和理论以来，众多学者一直致力于推动遗传算法的发展，对编码方式、控制参数的确定、选择方式和交叉机理等进行了深入的探究，将多种策略引入遗传算法，提出了各种变形的遗传算法（Variants of Canonical Genetic Algorithms，简称 VOGA）以提高性能。其基本途径概括起来有下面几个方面：

① 改变遗传算法的组成成分或使用技术，如选用优化控制参数、适合问题特性的编码技术等；

② 采用混合遗传算法；

③ 采用动态自适应技术，在进化过程中调整算法控制参数和编码粒度；

④ 采用非标准的遗传操作算子；

⑤ 采用并行遗传算法。

到目前为止，主要的几种改进的遗传算法包括：分层遗传算法、CHC 算法、messy 遗传算法、混合遗传算法、基于小生境技术的遗传算法、并行遗传算法、以及自适应遗传算法。

1. 分层遗传算法

分层遗传算法的思想是：对于一个问题，首先随机地初始化 $N \times n$ 个样本（$N \geq 2$，$n \geq 2$），并将它们分成 N 个子种群，每个子种群包含 n 个样本，接着对每个子种群独立地进行遗传进化，记它们为 $GA_i(I = 1, 2, \ldots, N)$。这 N 个种群所运行的遗传算法最好在设置特性上有较大的差异，这样就可以为将来的高层遗传算法产生更多种类的优良模式。分层遗传算法的基本流程如图 4.2 所示。

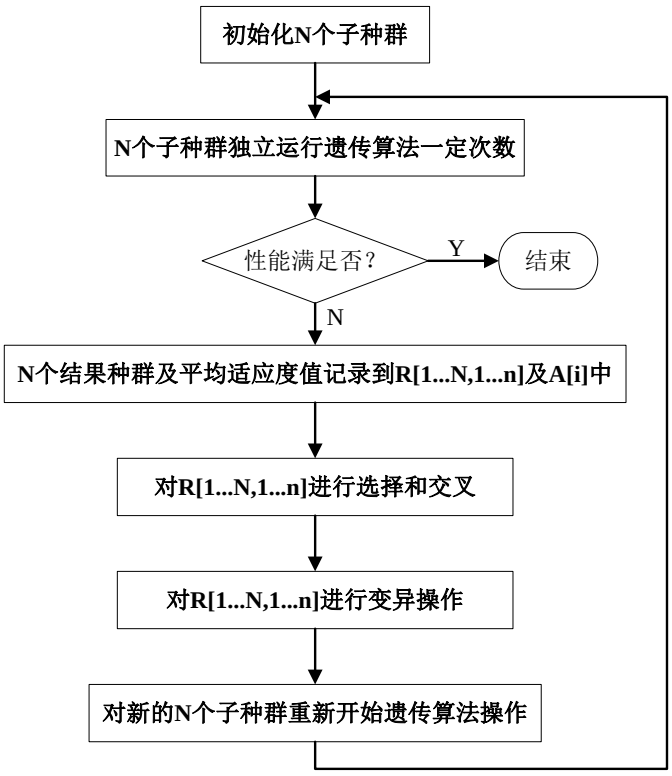


图 4.2　分层遗传算法流程图

2. CHC 算法

CHC 算法是 Eshelman 在 1991 年提出的一种改进的遗传算法的缩称，第一个 C 代表跨世代精英选择(Cross generational selection)策略，H 代表异物种重组(Heterogeneous recombination)，第二个 C 代表大变异(Cataclysmic mutation)。CHC 算法与基本遗传算法 SGA 不同点在于：SGA 的遗传操作比较单纯，简单地实现并行处理；而 CHC 算法牺牲这种单纯性，换取遗传操作的较好效果，并强调优良个体的保留。

3. messy 遗传算法

messy 遗传算法是 Goldberg 等人在 1989 年提出来的。这种改进算法是一种变长度染色体遗传算法。生物在进化过程中，染色体的长度是随着进化过程慢慢变化的；另外，遗传算法在实际应用中，有时也需要使用不同长度的编码串。因此这种改进算法在不影响模式定义距的情况下，使优良的模式得以增殖。

4. 混合遗传算法

梯度法、爬山法以及模拟退火算法等一些早期的算法虽然有各自的缺点，但它们的局部搜索能力还是很强，另外一些含有问题相关的启发知识的启发式算法的运行效率也很高。因此，如果能将这些算法的思想混合起来，去其糟粕取其精华，是可以构造更高效的遗传算法的，用这种思想构造的遗传算法就称为混合遗传算法。现在比较

常见的混合遗传算法有一下几种：模拟退火遗传算法[44-46]，免疫遗传算法[47-48]，小生境遗传算法[49-51]，模糊遗传算法[52-55]，混沌遗传算法[56-57]，以及量子遗传算法[58-60]等。

5. 基于小生境技术的遗传算法[61]

小生境(Niche)在生物学中是指特定环境下的一种生存环境。物以类聚，人以群分，其实任何生物都是这样，都会在一个特定的区域和特定群体生活在一起，例如鱼生活在水里就不会跑到陆地上来，蚯蚓喜欢在潮湿的泥土里生活，猫狗则喜欢在干燥的地方栖息。在用基本遗传算法求解多峰值函数的优化计算问题时，基本遗传算法往往陷入局部最优，这个缺陷难以克服。而在基本遗传算法的基础上借鉴小生境的概念能够较好的解决这一缺陷。

6. 并行遗传算法

并行遗传算法是一种多参数、多个体同时优化的方法。标准遗传算法是以个体为运行对象，遗传算法在运行过程主要的运算量在适应度函数的计算上。事实上，各个个体的适应度的计算是彼此独立的，具有一种天然的并行结构，因此，人们开发了多种并行遗传算法(Parallel Genetic Algorithm，简称 PGA)

7. 自适应遗传算法

在基本遗传算法当中，存在着 2 个非常重要的参数，它们是交叉概率 P_c 和变异概率 P_m，这 2 个参数对遗传算法行为、性能以及收敛性的影响是至关重要的。交叉概率 P_c 越大，2 个个体之间发生交叉的几率越大，那么生成新个体的速度也越快。但是，如果 P_c 过大，种群中适应度较高的个体则以极大的几率与其他个体发生交叉，从而使优秀个体遭到破坏；如果 P_c 过小，种群里个体间发生交叉的几率过小，难以给种群带来新个体，致使搜索过程缓慢。变异算子的作用是为种群产生新的基因，设置合适的变异概率 P_m 可以给种群带来新的基因，产生新鲜血液，也有机会生成更优秀的个体，和交叉率相同，过大过小的变异率也会导致算法的性能下降，甚至失效。过大的 P_m 使得遗传算法类似于甚至完全等同于纯粹的随机搜索算法，而 P_m 过小，则不易产生新的个体结构。交叉率 P_c 和变异率 P_m 的选取至关重要而又难以确定，需要进行反复的实验，但还是难以找到最佳值。Srinvivas 等人提出了一种自适应遗传算法(Adaptive GA,AGA)，在该算法中，交叉率和变异率能够根据种群进化的情况自动调整。当种群中各个个体的适应度非常接近或者趋于局部最优时，算法将自动调整 P_c 和 P_m 的值，使之增大，因为较大的 P_c 和 P_m 可以使种群更容易产生新的个体，跳出局部最优；当种群适应度差别比较大，算法将使 P_c 和 P_m 减小。另外，对于种群中适应度较高的个体，将被赋予较小的 P_c 和 P_m，这个该较优个体有更大的几率进入下一代；而种群中适应度较低的个体，将被赋予较高的 P_c 和 P_m 以使得该个体有更大的几率被淘汰，这样更符合达尔文“优胜劣汰，适者生存”的原理，保证了遗传算法中种群个体的多样性，同时又保证了其收敛性。

在自适应遗传算法中，P_c 和 P_m 按式（4.1）和式（4.2）进行自动调整：

$$P_c=\begin{cases}\dfrac{k_1(f_{\max}-f')}{f_{\max}-f_{avg}}. & f\geq f_{avg}\\ k_2, & f<f_{avg}\end{cases} \tag{4.1}$$

$$P_m=\begin{cases}\dfrac{k_3(f_{\max}-f)}{f_{\max}-f_{avg}}, & f\geq f_{avg}\\ k_4, & f<f_{avg}\end{cases} \tag{4.2}$$

式中，f_{max}——群体中最大的适应度值；

f_{avg}——每代群体的平均适应度值；

f'——要交叉的两个个体中较大的适应度值；

f ——要变异个体的适应度值。

这里将 k_1,k_2,k_3,k_4 取值在(0, 1)区间内即可实现自适应调整了。在这种思想中，当适应度越接近最大适应度，交叉率和变异率就越小；当等于最大适应度时，交叉率和变异率就变为零。对于算法处于进化后期时，这种方法可能比较合适，但是由于初期种群中较优个体几乎不发生变化，而这个时候的最优个体并不一定是全局最优个体，所以对于进化初期，这种计算公式并不理想，为此进行进一步的改进，按如下公式计算：

$$P_c=\begin{cases}P_{c1}-\dfrac{(P_{c1}-P_{c2})(f'-f_{avg})}{f_{\max}-f_{avg}}. & f'\geq f_{avg}\\ P_{c1}, & f'<f_{avg}\end{cases} \tag{4.3}$$

$$P_m=\begin{cases}P_{m1}-\dfrac{(P_{m1}-P_{m2})(f_{\max}-f)}{f_{\max}-f_{avg}}, & f\geq f_{avg}\\ P_{m1}, & f<f_{avg}\end{cases} \tag{4.4}$$

其中，$P_{c1}=0.9$, $P_{c2}=0.6$, $P_{m1}=0.1$, $P_{m2}=0.001$。f_{max} 为群体中最大的适应度值；f_{avg} 为每代群体的平均适应度值；f'为要交叉的两个个体中较大的适应度值；f 为要变异个体的适应度值。

本文将采用自适应遗传算法对评估函数进行优化。

4.4 遗传算法应用于评估函数

已经有人尝试着把遗传算法引入评估函数[62-63]，但其效果不是很理想。本文对把遗传算法应用于六子棋评估函数的参数调整和优化的策略做了相应的改进，取得了一定的效果。

4.4.1 编码

在使用遗传算法进行优化搜索之前，首先要将所求问题解空间的可行解数据表示成遗传空间的基因型串结构数据，这些串结构数据的不同组合就构成了不同的可行解。普遍使用的编码包括二进制编码、灰度编码、可分解/可拼接编码、实数编码、符号编码等。在本文中，采用遗传算法所要优化的对象是六子棋机器博弈系统评估函数中各个参数，这些参数代表的是一个路中含有不同个数的同色棋子所对应的分数。那么总共就有 7 个参数，分别是含有 0 到 6 颗棋子时有效路的分数。因为当一条路不含有棋子时，它对双方均不产生威胁，因此它的分数可是设置为最低，这里设为 0；而当一条“路”上含有 6 粒棋子时，则表示该色棋子已经获胜，因此给它设置一个确定的最高分，这里设为 10000。那么这两个参数已经确定下来，有待确定和优化的参数剩下有 5 个，即一条路上分别有 1、2、3、4、5 粒棋子的情况下该路的分值。

遗传算法处理的对象主要是个体，个体包括一组染色体串，其中每一个染色体对应于评估函数中的一个参数值，也就是上面所提到待定的“路”的 5 种状态所设置的分值。将这 5 个参数组合在一起，即为一个个体。例如：如果这 5 个参数分别为 100、200、400、800、1000，则其对应的个体如图 4.3 所示。

100	200	400	800	1000

图 4.3 染色体串

4.4.2 适应度函数的计算

遗传算法在遗传进化的过程中，主要依据适应度函数计算得到的个体适应度值的大小来进行遗传进化操作。虽然遗传算法在优化过程中不需要外部信息，但是适应度函数的准确度却是至关重要的，它必须准确的反应个体的优劣好坏。错误的适应度函数显然会造成遗传算法朝错误的方向进化，无法得到更优秀的个体或种群。

对于一般的优化问题，适应度函数往往是根据目标函数进行变形转换得来的。例如：要解决一元函数 $f(x)=x\sin(10\pi\cdot x)+2.0 \quad x\in[-1,2]$ 的最大值的优化问题，我们可以直接采用函数 $f(x)=x\sin(10\pi\cdot x)+2.0$ 来作为遗传算法的适应度函数。某个个体能够使适应度函数 $f(x)=x\sin(10\pi\cdot x)+2.0$ 的值越大，则表示该个体越优，显然这个适应度函数能够正确反应个体符合题设的程度。但是对于机器博弈来说，因为没有一个确定的目标函数，棋局的好坏程度也很难用一个线性或者非线性的函数来表示，因此本文采用了文献[62]中锦标赛算法的思想，并做了相应的改进。

4.4.2.1 锦标赛算法应用于六子棋

在文献[62]中，作者将锦标赛算法应用在了中国象棋的遗传算法优化问题当中，文献[63]中，作者又将锦标赛算法移植到了六子棋的遗传算法优化问题当中，锦标赛

算法的主要思想就是让各个个体之间进行互相对弈，通过统计每个个体胜利的次数和比率来反映个体的优劣程度。其具体过程如下：

首先，将包含有 m 个个体的种群打散分成 n 组，分别记为 $Group[i]$（i = 1，2，3，…，n），那么每个小组就包含有 m/n 个个体，所有个体的适应度初始值为 0。然后分别在各组内再进行小组内的循环赛训练，即让每个都要与小组内其他所有个体之间进行先后手的 2 场比赛，每一回合胜利的一方适应度加 1，失败的一方适应度减 1，最后得分最高的即为每个组的冠军，分别记为 $Champion[i]$（i = 1, 2, 3…, n），其为筛选出的最佳。最后对 n 个冠军相互之间做选择、交叉和变异等遗传操作产生 $m-n$ 个新个体，分别记为 $NewBody[i]$（$i = n+1, n+2, \ldots m$），以 $Champion[i]$ 和 $NewBody[i]$ 为个体形成一个个体数为 m 的新种群，即下一代种群。

例如：

① 现有包含 20 个个体的种群，个体分别记为 Body[1]、Body[2] …Body[20]。

② 把这 20 个个体随机分成 5 组，分别记为 Group[1]、Group[2] …Group[5]，每组含有 4 个个体。

假设分组结果为：

Group[1]：Body[1]、Body[2]、Body[3]、Body[4]

Group[2]：Body[5]、Body[6]、Body[7]、Body[8]

Group[3]：Body[9]、Body[10]、Body[11]、Body[12]

Group[4]：Body[13]、Body[14]、Body[15]、Body[16]

Group[5]：Body[17]、Body[18]、Body[19]、Body[20]

③ 让每个小组内的个体与本小组其他所有个体对弈，进行循环赛训练，第一小组训练的结果如表 4.1 所示。

表 4.1　第一小组比赛结果

Group[1]	Body[1]	Body[2]	Body[3]	Body[4]
Body[1]		负	胜	胜
Body[2]	胜		胜	胜
Body[3]	负	负		负
Body[4]	负	负	胜	

表中某个值所对应的行列坐标分别表示进行对弈的 2 个个体，行坐标表示采取先手的个体，列坐标表示采取后手的个体，比赛结果以第一列的个体为准。例如表 4.1 中行坐标是 Body[1]，列坐标是 Body[3]的结果是胜，它表示：个体 1 和个体 3 进行对弈，个体 1 采取先手，对弈的结果是个体 1 胜利。那个根据表 4.1 所给出的对弈结

果，假设所有个体的初始适应度都为 0，适应度函数用 F 表示，可统计计算出该小组中各个个体的适应度为：

F（Body[1]）= 2； F（Body[2]）= 6；

F（Body[3]）= -6； F（Body[4]）= -2；

那么，第一组的冠军就为适应度函数最高的个体 Body[2]，

即 Champion[1]= Body[2]。

那么根据同样的方法，在 Group[2]、Group[3]、Group[4]、Group[5]中分别找出它们的最优个体 Champion[2]、Champion[3]、Champion[4]、Champion[5]，再根据所计算出来的适应度，在各个 Champion[i](i=1,2,3,4,5)中进行选择、交叉、变异等遗传操作，求得 NewBody[i]（i = 1, 2,…,15），那么 Champion[i](i=1,2,3,4,5)和 NewBody[i]（i = 1, 2,…,15）共 20 个个体形成了新一代的种群，进入下一代的遗传进化过程。

该算法存在着一些问题：

首先，将种群分成小组，取每个小组的最优个体进行选择、交叉、变异等遗传操作，这样每个小组只有那一个最优个体可以进行遗传操作，将它的基因遗传到下一代，其他个体的优秀基因可能得不到遗传。假设原来种群有 20 个个体，将这 20 个个体分成 5 个小组，每组 4 个个体，每个小组内部的 4 个个体进行锦标赛选择一个最优个体出来进行遗传操作，这样 5 个小组只能产生 5 个最优个体进行遗传操作，也就是说 20 个个体里面只有 5 个个体的基因可能遗传到下一代中，而其他 15 个占了初始种群的 75%却都得不到遗传，这样可能造成部分优秀基因得不到遗传，同时可能造成收敛过快；

其次，直接采用在小组内对弈取得的成绩作为适应度来与其他小组的优胜者进行遗传操作是不准确的，如果小组 1 内的最优个体在小组 1 内取得了 8 分，小组 2 内的最优个体在小组 2 内取得了 6 分，并不能说明个体 1 一定比个体 2 优秀，这将造成适应度的评判错误。

4.4.2.2 对锦标赛算法的改进

为了避免过去锦标赛算法存在的潜在问题，本文采用的做法是：

①将包含 m 个体 individual[i] (i = 1 , 2 , … , m)的种群随机分成 n 个小组 group[j] (j=1,2,…, n)。初始时，假设个体适应度 fitness[i] = 1 (i = 1 , 2 , … , m)，先在单个小组 group[j] (j = 1 , 2 , … , n)内部进行循环赛，所有个体都与 group[j]内其它 n-1 个个体进行先手和后手的 2 次对局，其适应度值的计算规则是胜方加 1，败方适应度不变，平局为双方各加 0.5。待 n 个小组的内循环赛都结束后，将 n 个小组中每小组的最优个体提取出来，并分别记录为这 n 个小组内最优个体的适应度为 best_fitness[j](j = 1,2,…, n)。

②将各小组的最优个体组成一个精英个体种群，命名为精英小组，在精英小组进行循环赛，并分别记录精英小组中 n 个精英个体的适应度为 elite_fitness[j] (j = 1,2,…, n)。

③确定包含 m 个个体 individual[i] (i = 1,……,m)的父代种群的适应度值，由式(4.6)求得：

$$fitness[i] = \frac{fitness[i] * elite\_fitness[j]}{best\_fitness[j]} \quad （i = 1,2•••m \quad j = 1,2,•••n） \qquad (4.5)$$

式（4.5）中 *i* 为个体编号，*j* 为该个体 *i* 所在的小组编号。

④将每个小组当中的最优个体加入到当前父代的子代种群中间。

⑤在当前父代种群的所有个体中进行遗传操作，产生剩下新的子代个体。

例如：

表 4.2 给出了小组 1 的结果

表 4.2 小组 1 的比赛结果

Group[1]	Body[1]	Body[2]	Body[3]	Body[4]
Body[1]		负	胜	胜
Body[2]	胜		胜	胜
Body[3]	负	负		负
Body[4]	负	负	胜	

可计算出第一组中各个个体的适应度为：

F（Body[1]）= 5； F（Body[2]）= 7；

F（Body[3]）= 1； F（Body[4]）= 3；

那么，第一组的冠军就为适应度函数最高的个体 Body[2]，

Champion[1]= Body[2]

best_fitness[1] = 7

表 4.3 给出了小组 2 的结果

表 4.3 小组 2 的比赛结果

Group[2]	Body[5]	Body[6]	Body[7]	Body[8]
Body[5]		胜	胜	胜
Body[6]	负		胜	负
Body[7]	负	负		负
Body[8]	平	胜	胜	

可计算出第二组中各个个体的适应度为：

F（Body[5]）= 6.5； F（Body[6]）= 3；

F（Body[7]）= 1； F（Body[8]）= 5.5；

那么，第二组的冠军就为适应度函数最高的个体 Body[5]，

Champion[2]= Body[5]

best_fitness[2] = 6.5

表 4.4 给出了小组 3 的结果。

表 4.4 小组 3 的比赛结果

Group[3]	Body[9]	Body[10]	Body[11]	Body[12]
Body[9]		胜	负	平
Body[10]	负		负	负
Body[11]	平	胜		胜
Body[12]	平	胜	负	

可计算出第三组中各个个体的适应度为：

F（Body[9]）= 4.5； F（Body[10]）= 1；

F（Body[11]）= 6.5； F（Body[12]）= 3.5；

那么，第三组的冠军就为适应度函数最高的个体 Body[11]，

Champion [3] = Body [11]

best_fitness [3] = 6.5

表 4.5 给出了小组 4 的结果。

表 4.5 小组 4 的比赛结果

Group[4]	Body[13]	Body[14]	Body[15]	Body[16]
Body[13]		负	胜	负
Body[14]	平		平	平
Body[15]	胜	平		平
Body[16]	平	胜	胜	

可计算出第四组中各个个体的适应度为：

F（Body[13]）= 3； F（Body[14]）= 4；

F（Body[15]）= 3.5； F（Body[16]）= 5.5；

那么，第四组的冠军就为适应度函数最高的个体 Body[16]，

Champion[4]= Body[16]

best_fitness[4] = 5.5

表 4.6 给出了小组 5 的结果。

表 4.6 小组 5 的比赛结果

Group[5]	Body[17]	Body[18]	Body[19]	Body[20]
Body[17]		负	胜	胜
Body[18]	平		胜	胜
Body[19]	平	负		平
Body[20]	平	平	胜	

可计算出第五组中各个个体的适应度为：

F（Body[17]）= 4.5； F（Body[18]）= 6；

F（Body[19]）= 2； F（Body[20]）= 3.5；

那么，第五组的冠军就为适应度函数最高的个体 Body[18]，

Champion[5] = Body[18]

best_fitness[5] = 6

分别得到五组的冠军 Champion[1] 、Champion[2]、 Champion[3]、Champion[4]、Champion[5]为最佳：

Champion[1]= Body[2]

Champion[2]= Body[5]

Champion[3]= Body[11]

Champion[4]= Body[16]

Champion[5]= Body[18]

这五个冠军组成小组进行循环赛，假设该精英小组经过循环赛后计算的适应度为：

F_ Champion (Body[2]) =3

F_ Champion (Body[5]) =5

F_ Champion (Body[11])=9

F_ Champion (Body[16])=6

F_ Champion (Body[18])=2

根据式（4.5）计算：

第一组的比例如下：

F_Champion(Body[2]) = 3

Fitness(Body[1]) = 5 * 3 / 7 = 15/7

Fitness(Body[2]) = 7 * 3 / 7 = 3

Fitness(Body[3]) = 1 * 3 / 7 = 3/7

Fitness(Body[4]) = 3 * 3 / 7 = 9/7

第二组的比例如下：

F_Champion(Body[5]) = 5

Fitness(Body[5]) = 6.5 * 5 / 6.5 = 5

Fitness(Body[6]) = 3 * 5 / 6.5 = 15 / 6.5

Fitness(Body[7]) = 1 * 5 / 6.5 = 5 / 6.5

Fitness(Body[8]) = 5.5 * 5 / 6.5 = 27.5 / 6.5

第三组的比例如下：

F_Champion(Body[11]) = 9

Fitness(Body[9]) = 4.5 * 9 / 6.5 = 40.5 / 6.5

Fitness(Body[10]) = 1 * 9 / 6.5 = 9 / 6.5

Fitness(Body[11]) = 6.5 * 9 / 6.5 = 6.5

Fitness(Body[12]) = 3.5 * 9 / 6.5 = 31.5 / 6.5

第四组的比例如下：

F_Champion(Body[16]) = 6

Fitness(Body[13]) = 3 * 6 / 5.5 = 18 / 5.5

Fitness(Body[14]) = 4 * 6 / 5.5 = 24 / 5.5

Fitness(Body[15]) = 3.5 * 6 / 5.5 = 21 / 5.5

Fitness(Body[16]) = 5.5 * 6 / 5.5 = 6

第五组的比例如下：

F_Champion(Body[18]) = 2

Fitness(Body[17]) = 4.5 * 2 / 6 = 9 / 6

Fitness(Body[18]) = 6 * 2 / 6 = 2

Fitness(Body[19]) = 2 * 2 / 6 = 4 / 6

Fitness(Body[20]) = 3.5 * 2 / 6 = 7 / 6

计算出所有个体的适应度以后，再在所有个体中进行遗传操作。

经过改进以后，改变了以前只是在各个小组的最优个体中间进行选择，复制，交叉，变异等操作的不足，使得遗传操作是在整个种群中进行，避免了以前仅在最优个体中进行遗传操作，造成大部分次优个体的优秀基因得不到机会遗传的问题，同时避免了群体收敛过快的问题，保证了种群的多样性。

4.4.3 选择算子

选择算子是遗传算法的三个基本算子之一，它是交叉操作的前提条件，只有选择出了合适的父代个体，才能在所选择的父代个体上进行交叉操作，产生新的个体。选择的方法多种多样，包括随机遍历抽样(stochastic universal sampling)、局部选择(local selection)、截断选择(truncation selection)、轮盘赌选择(roulette wheel selection)等等。本文选择采用轮盘赌选择算法。

采用轮盘赌选择算法之前，需要先计算个体的适应度，在上一节中已经计算出了种群中所有个体的适应度。轮盘赌选择算法的原理是使得适应度越大的个体被选择到的概率越大，从而使优秀个体的优秀基因被遗传到下一代的几率更大。由于该算法类似于博彩游戏中的轮盘赌，因此命名为轮盘赌选择。

例如：假设个体 A、B、C、D 的适应度分别为 5、2、2、1，那么它们的选中概率如表 4.7 所示。

表 4.7 个体 A、B、C、D 的适应度及相关信息

个体	适应度	选择概率	累计概率
A	5	0.5	0.5
B	2	0.2	0.7
C	2	0.2	0.9
D	1	0.1	1

首先根据个体的适应度计算个体被选中的概率，然后将轮盘分层 4 个扇区，如图 4.4 所示。

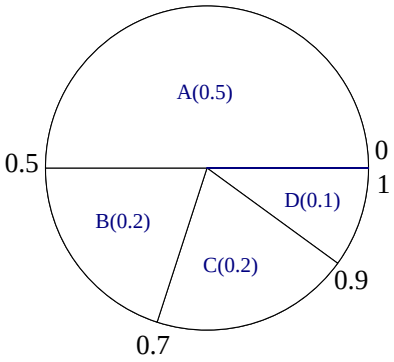


图 4.4 轮盘赌实例

因为种群大小是 4，因此需要产生 4 个[0,1]之间的随机数，就相当于转动了 4 次轮盘，获得了 4 个转盘停止时指针的位置，指针停止在某一扇区，该扇区代表的个体即被选中。根据表 5.7 及图 5.4 可以知道，当随机数在[0,0.5]内，则选择 A；在[0.5,0.7]内，则选择 B；在[0.7,0.9]内，则选择 C；在[0.9,1]内，则选择 D。假设 4 个随机数分别是 0.22、0.43、0.67、0.84，那么产生的新的种群是 A、A、B、C。可以发现，适应度低的个体 D 在轮盘赌选择算法中被淘汰掉，而适应度高的个体 A 则 2 次被选中，所以，轮盘赌选择算法体现了达尔文优胜劣汰的原则，使优秀个体被保留的概率更大，而较差的个体被淘汰的概率更大，从而达到了使种群进化的目的。

4.4.4 交叉算子

交叉，也叫做重组，是把两个父个体的基因进行重新组合，形成新的个体。在遗传算法中，交叉算子起到了核心的作用，因为它影响着遗传算法的全局搜索能力，它通过在两个父代个体的基因位中随机选取一个或多个基因位作为交叉的位置，然后交换父代个体部分结构组成新个体，从而使遗传算法的搜索能力得以提高。最常用的交叉算子是一点交叉，但一点交叉一次操作只能改变搜索空间的一个参数。两点和多点交叉表现出很好的搜索能力[64]。它们可以同时改变搜索空间的多个参数，因此搜索路径为空间任意方向。一致交叉[65]作为一种特殊的多点交叉，原理上可形成任意新的模式，有助于搜索到解空间中新的区域，但也极有可能破坏低阶、短距、高适应度的模式。多点交叉可等效为几个一致交叉的组合[66]。由于均匀交叉更加广义化，将每个参数都作为了潜在的交叉点，因此本文将选用均匀交叉。

均匀交叉的思想是：首先根据交叉率 P_c 随机地产生一个与个体等长的 0 - 1 掩码，掩码中的片断表明了哪个父个体向子个体提供变量值，然后通过这个掩码和选择的父个体一起确定子个体。如图 4.5 所示：

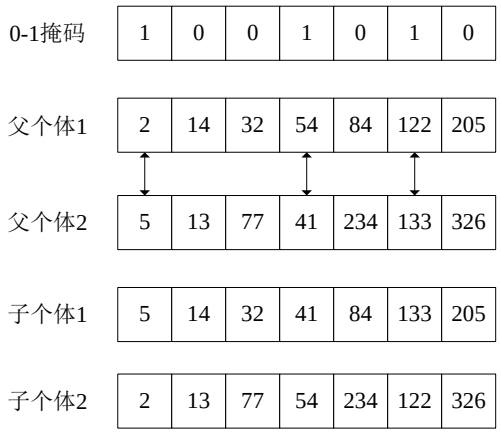


图 4.5　均匀交叉

图 4.5 中已经给出通过选择算子选择出的 2 个父代个体，父个体 1{ 2 , 14 , 32 , 54 , 84 , 122 , 205 }和父个体 2{ 5 , 13 , 77 , 41 , 234 , 133 , 326 }，另外根据交叉率 P_c 随机生成 0 - 1 掩码为 1001010，0-1 掩码表示 2 个父代个体的 1、4、6 位需要进行交叉，那么交叉后的子代为子个体 1{ 5 , 14 , 32 , 41 , 84 , 133 , 205 }和子个体 2{ 2 , 13 , 77 , 54 , 234 , 122 , 326 }。

4.4.5　变异算子

所谓变异算子，是指将个体编码串中的基因值根据变异概率 P_m 来产生一个新的基因值来替换原来的值，从而形成一个新的个体，它在遗传进化过程中同样具有重要的作用。根据达尔文进化论的思想，交叉使得种群中的信息在不同个体之间得到交换，所产生的后代能够继承父代的优良基因，新一代种群对环境的平均适应度要优于父代。但是交叉算子只是对现有的基因进行排列上的重组，不能产生新的基因材料。随着搜索的进一步进行，种群中的个体也具有局部相似性，这种局部相似性将导致交叉失败，使搜索处于停滞状态，陷入局部最优解之中，即发生“早熟”现象。此时变异算子就尤为重要，变异算子以较小的概率改变个体的基因值，使种群中的个体呈现多样性，引导搜索转向，从而有机会爬山，收敛到全局最优。所以变异操作是引导搜索爬山，防止“早熟”最重要的手段。可以看出，变异算子设计的成功与否决定着搜索能否达到更好解或者最优解。一般的变异方法包括：实值变异、二进制变异等。

本文中采用这样的变异算子：

首先根据变异率 P_m 确定种群中对应于某个个体的 0-1 掩码，0-1 掩码的长度与个体编码串的长度相同。0-1 掩码中 1 对应的位置，就是该个体将要发生变异的位置。如果某一基因值 x 需要发生变异，那么首先产生一个[-0.2 , 0.2]之间的随机数 r，该基因值由 x 更改为 $x \cdot (1 + r)$ 。变异算子的例子如图 4.6 所示：

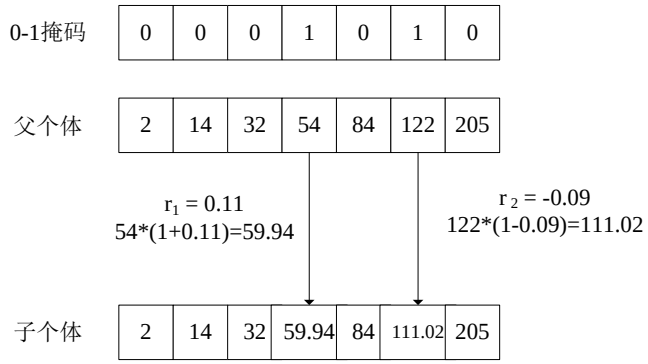


图 4.6 变异算子

假设父个体的编码串为{ 2 ， 14 ， 32 ， 54 ， 84 ， 122 ， 205 }，根据变异率 P_m 生成的 0-1 掩码为 0001010，由于 0-1 掩码的 4、6 位为 1，所以父个体 4、6 位的基因值将要发生变异。首先在[-0.2 ， 0.2]之间随机生成 r_1 = 0.11 ， r_2 = -0.09，所以变异后子个体的 4、6 为基因值分别变异为 $54 * (1 + r_1) = 59.94$, $122 * (1 + r_2) = 111.02$，变异后的子个体的编码串为{2 ， 14 ， 32 ， 59.94 ， 84 ， 111.02 ， 205}。

4.5 遗传算法优化评估函数的实验结果及分析

4.5.1 算法收敛性实验与分析

使用自适应遗传算法对评估函数的参数进行优化时，遗传算法的各个参数设置如表 4.8 所示。

表 4.8 遗传算法参数设置

种群大小	组数	编码串长度	初始种群
49	7	5	每个个体的5个基因分别在[0,4000]随机产生
交叉率 P_c	变异率 P_m		
动态调整	0.15		

实验中随机生成的初始化种群的片段如图 4.7 所示，可见每个个体内的 5 个基因是没有规律完全随机产生的。初始种群内某 2 个个体进行对弈的棋局如图 4.8 所示。

图 4.7　遗传算法初始化种群片段

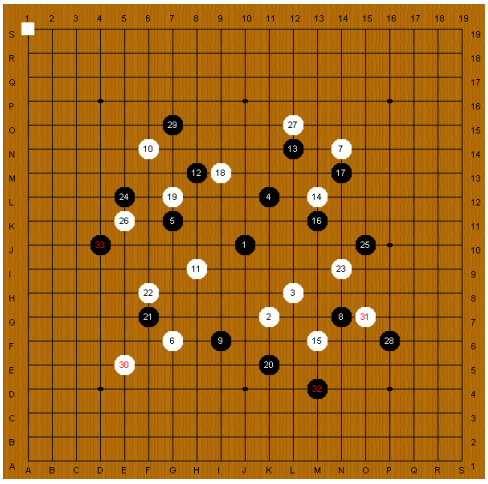


图 4.8　优化前的对弈情况

从图 4.8 可以看到，在优化前，对于随机生成的参数组，博弈程序落子的位置杂乱无章，毫无智能可言。若将改进前的锦标赛算法引入遗传算法进行优化，种群大小为 49，分为 7 个小组，搜索进行到 7 代就基本收敛，最优个体与初始种群有关，之后进化缓慢。使用该方法进化到 7 代以后的种群片段如图 4.9 所示。可见改进前的算法收敛过快。

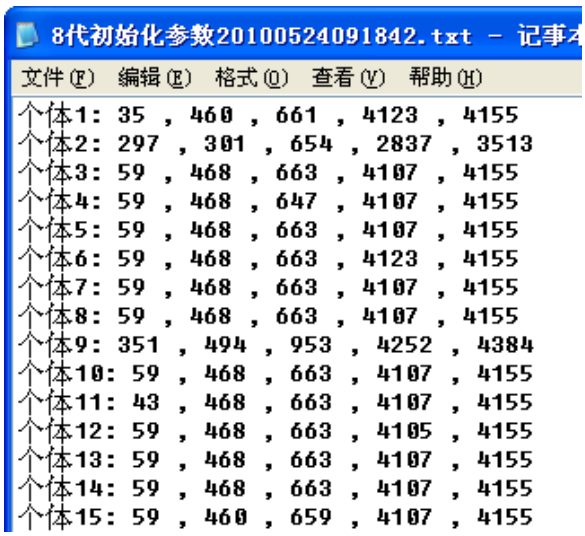


图 4.9　改进前遗传算法的种群片段

接下来采用改进后的遗传算法进行进化计算，观察每一代染色体的情况。图 4.10 展示了改进后的遗传算法进化 15 代后种群的染色体片段。从图中可以发现，即使进化到 15 代，种群中依然保持较多的基因种类，算法并没有收敛，因此，可以得出结论：改进后的遗传算法能够保持种群的多样性，避免过早收敛的情况。

图 4.10 改进后的遗传算法第 15 代种群片段

4.5.2 优化效果的实验与分析

在与上一节实验条件相同的情况下，用改进后的遗传算法对评估函数参数进行寻优，进化过程中每一代的最优个体所代表的参数组如图 4.11 所示。

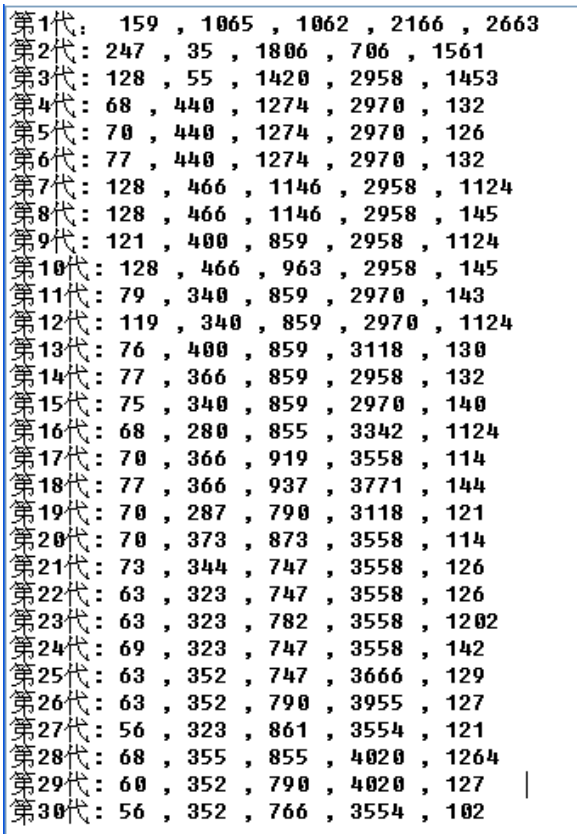


图 4.10 改进后的遗传算法前 30 代最有个体

从图 4.11 中可以看到，种群进化到 15 代的时候已经趋于稳定，此时博弈程序的棋力明显得到提升，与之前凭人工经验确定的一组参数进行对弈，双方获胜次数如表 4.9 所示。

表 4.9　GA15 代与人工确定的参数对弈结果

对弈次数 / 对弈双方	100	200	300	平均胜率
人工确定的参数	21 胜	58 胜	72 胜	151
GA 15 代	79 胜	142 胜	228 胜	449
GA 15 代胜率	79%	71%	76%	75%

可见进化 15 代的参数胜率达到 75%，在图 4.11 中，黑方是算法进化 15 代后确定的参数组，白方是人工凭经验确定的参数，此时优化后的参数组是 $\{68, 280, 855, 3342, 1124\}$。

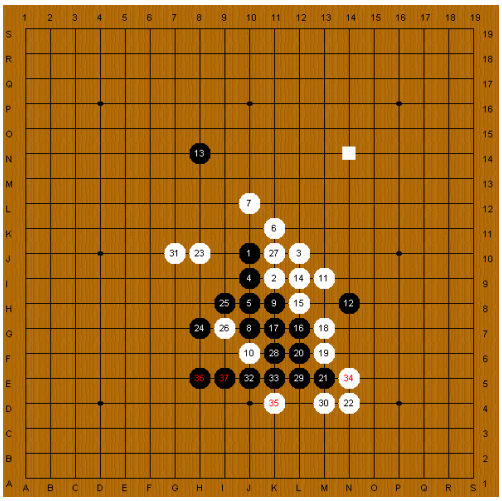


图 4.10　GA 优化 15 代后对弈棋局

若让算法再继续搜索到 30 代，可以看到 15 代到 30 代之间，参数的变化并不大，只是在局部进行调整，将 30 代的的参数组与人工确定的参数再次进行对弈，优化后的参数取得了 81%的胜率，此时的参数组是 $\{56, 352, 766, 3554, 102\}$，详细结果如表 4.10 所示。

表 4.10　GA30 代与人工确定的参数对弈结果

对弈次数 / 对弈双方	100	200	300	总和
人工确定的参数	23 胜	36 胜	57 胜	116 胜
GA 30 代	77 胜	164 胜	243 胜	484 胜

GA 30 代胜率	77%	82%	81%	81%

4.6 小结

通过上述实验验证和实战性对抗，证明该策略和算法保持了种群的多样性，避免了以前算法收敛过早的问题，改进效果明显、有效。尽管文章的研究对象是六子棋机器博弈系统，但其思想对其它机器博弈系统也有借鉴意义。

5 基于微粒群算法评估函数的构造

5.1 微粒群算法及其应用改造

5.1.1 微粒群算法与人工生命

微粒群算法(PSO)是一种演化计算技术，是 1995 年由美国社会心理学家 James Kennedy [67]和电气工程师 Russell Eberhart [68]共同提出的。他们萌生微粒群算法的思想是受到他们早期对许多鸟类群体的行为进行建模与仿真研究结果的启发。他们的模型及仿真算法主要利用了生物学家 Frank Heppner 的模型。Frank Heppner 的鸟类模型在反映群体行为方面与其他类模型有许多相同之处，所不同之处在于:鸟类被吸引飞向栖息地。在该仿真模型中，一开始每只鸟均无特定的目标进行飞行，直到一只鸟飞到了栖息地，随后其他的鸟也将飞到栖息地，从而就形成了鸟群。

事实上,鸟类的飞行规则很简单,即每一只鸟都试图停在鸟群中而又不互相碰撞。若种群中的某些鸟发现栖息地并飞离鸟群飞向栖息地时,周围的其他鸟也将飞向栖息地。接着将有越来越多的鸟落在此栖息地。

Eberhar 和 Kennedy 对 Heppner 的模型进行了修正，以一个没有质量没有体积的微粒代表群体中的一只鸟，结合信息交流的社会性（群体中的个体通过与周围同类的个体进行比较，并模仿其优秀的行为）与智能性，使群体中每一微粒能够飞向潜在的解空间并在最好解处降落。该算法的关键在于如何保证微粒降落在最好解处而不降落在其他诸如局部最优解处，这就是信息交流的社会性及智能性所在。

Eberhart 和 Kennedy 经过研究，在 1995 年的 IEEE 的国际神经网络学术会议上正式发表了题为"Particle Swarm Optimization"的文章，提出了微粒群优化算法(Particle Swarm Optimization，PSO)，这标志着微粒群算法的正式诞生。

微粒群算法最早是被应用于非线性连续函数的优化和神经元网络的训练[69-70]，后来也被用于解决约束优化问题，Ebethart 和 Kennedy 后来又提出了 PSO 的离散二进制版本[71]，用于解决组合优化问题。Eberhart 还用 PSO 分析了人类的帕金森综合症等颤抖类疾病。Yoshida 等人通过 PSO 对各种离散的连续变量进行优化，达到了控制核电机组输出稳定电压的目的。事实上，已经有人将微粒群算法应用在了人工智能领域，文献[72]提出了一种基于粒子群算法的足球机器人的动作选择算法。该算法给出了一个足球机器人的动作集合，根据赛场的实际情况为足球机器人分配角色与任务，并利用粒子群算法为足球机器人选择合适的动作。可以说，该足球机器人也是一个人工生命体，文中采用微粒群算法也是为了解决该人工生命体的行为选择问题。但是到

目前为止，还没有学者将微粒群算法应用到机器博弈的研究领域，因此，本文尝试将微粒群算法应用到博弈机器人行为选择的参数优化当中，取得了不错的效果。

5.1.2 标准微粒群算法

PSO 算法与遗传算法有相似之处，都是基于“群体”与“进化”的思想，也需要计算群体中个体的适应度，根据适应度来进行后续操作。在微粒群算法 PSO 的思想中，将群体中的每个个体看作是 D 维搜索空间中的一个没有体积没有质量的微粒（点），在搜索空间中以一定的速度和方向飞行，飞行速度以及方向根据它本身的飞行经验和同伴的飞行经验来动态调整。所有的微粒都有自己的适应度（fitness value），这个适应度的大小取决于该微粒在 D 维搜索空间中所处的位置。那么在本文所针对的问题中，由于需要优化的参数有 5 个，即“路”中分别存在 1～5 颗棋子时该“路”给定的分值 ScoreOfPath[1]～ScoreOfPath[5]，因此这里的搜索空间是 5 维的，只有当 5 个参数达到最优化，那么代表这 5 个最优参数的微粒所处的位置的适应度就最大。

标准 PSO 首先随机的初始化一定数量的微粒，这些微粒在搜索空间中的位置以及微粒的速度和移动方向都是随机产生的，然后通过自身的经验和群体中其他微粒的经验共同影响来调整微粒的速度和方向，在逐渐调整中找到最优解。微粒通过跟踪两个“极值”来调整自己的速度和方向，一个是微粒在自身移动过程中所经过的最优位置，称为个体极值 P_{id}，它相当于微粒自己的移动经验；另一个极值是整个群体中所有微粒目前搜索到的最优位置，该极值是全局极值 P_{gd}，可以看作是微粒群体的社会经验。另外，也可以不用整个种群而只用其中一部分为微粒的邻居，那么在所有邻居中的极值就是局部极值。用整个种群的就是全局版 PSO，用一部分为邻居的就是局部版 PSO，全局版收敛快，但有时会陷入局部最优，局部版收敛慢，但不容易陷入局部最优。在实际应用中，可以先使用全局版 PSO 搜索到解的大致位置，再用局部版 PSO 进行小范围搜索。

在叙述标准 PSO 算法的过程之前，先定义几个变量的含义：

m：算法中种群的规模，即种群中微粒的个数为 m；

D：搜索空间的维数，也可以说是待搜索参数的个数。在本文中，待优化的参数是 5 个，因此本文中 $D = 5$。

$X[i]$：$X[i]$是一个 D 维向量 $X[i] = \{ x_{i1}, x_{i2}, x_{i3}, \dots, x_{iD} \}$，它代表种群中微粒 i 在 D 维搜索空间中的位置。在本文中，$X[i]$是一个 5 维向量，即 $X[i] = \{ x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5} \}$。

$V[i]$：$V[i]$也是一个 D 维向量 $V[i] = \{ v_{i1}, v_{i2}, v_{i3}, \dots, v_{iD} \}$，它代表微粒群中微粒 i 在 D 维搜索空间中各个维度方向上的飞行速度。在本文中，也是一个 5 维向量，即 $V[i] = \{ v_{i1}, v_{i2}, v_{i3}, v_{i4}, v_{i5} \}$。

fitness[i]： *fitness[i]*是微粒群中微粒 *i* 的适应度。

pbest[i]： *pbest[i]*是微粒 i 自身搜索到的最优位置。

pbest_fitness[i]： *pbest_fitness[i]*是微粒 *i* 目前所经过的适应度最大的位置，即位置 *pbest[i]*处的适应度。

gbest： *gbest* 是整个微粒群当中到目前为止搜索到的最优位置。

gbest_fitness： *gbest_fitness* 是整个微粒群当中到目前为止搜索到的最优位置的适应度。

标准 PSO 算法的过程如下：

①初始化微粒群，在合理的范围内随机生成微粒群中 *m* 个微粒的初始位置 *X[i]* 和初始速度 *V[i]*（*i = 1 , 2 , ... , m* ），计算每个初始微粒的适应度 *fitness[i]*，同时比较并设置 *pbest_fitness[i]*、*gbest* 和 *pbest[i]*，*i = 1 , 2 , ... , m*；

②根据式(5-1)和式(5-2)调整各个微粒的速度 *V[i]*和位置 *X[i]*；

$$v_{id} = w \cdot v_{id} + c_1 \cdot rand() \cdot (p_{id} - x_{id}) + c_2 \cdot rand() \cdot (P_{gd} - x_{id}) \quad (5\text{-}1)$$

$$x_{id} = x_{id} + v \quad (5\text{-}2)$$

其中 v_{id} 表示微粒 *i* 在第 *d* 维上的速度，c_1 和 c_2 称为学习因子，它们是调节 P_{id} 和 P_{gd} 的重要参数，分别调节向全局最好微粒和个体最好微粒方向飞行的最大步长。如果 c_1 和 c_2 过小，那么微粒受到自己飞行经验和微粒群中其他微粒的飞行经验也越小，可能导致远离目标区域；如果 c_1 和 c_2 过大，那么可能导致微粒突然向当前搜索的最优位置以很大的步长移动，或者飞出合理区域，或者错过了飞行过程中可能遇到的更合理的位置[73]，因此设置 c_1 和 c_2 为[0, 1]内均匀分布的随机数。式中的 *w* 称为惯性权重，它的作用是使微粒保持原有的运动惯性，使其有扩展搜索空间的趋势，有能力搜索新的区域[74]。若 w 较大，那么当前的速度对调整后的速度影响较大，从而能够扩大搜索范围，提高了算法的全局搜索能力；若 w 较小，那么当前的速度对调整后的速度影响较小，算法主要在当前位置的附近搜索，算法的局部搜索能力得到加强。

可以发现，式（5-1）由三部分组成，第 1 部分体现了微粒对当前自身运动状态的信任，依据自身的速度进行惯性运动；第 2 部分是“认知(cognition)”部分，体现了微粒本身获得的经验对自己下一步行为的影响；第 3 部分是“社会(social)”部分，表示微粒间的信息共享和相互合作，即群体信息对微粒下一步行为的影响。

③计算移动后微粒的适应度 *fitness[i]*，比较每个微粒当前的适应度与该微粒所经过的最好位置的适应度 *pbest_fitness[i]*，如果 *fitness[i]* > *pbest_fitness*；则更新 *pbest_fitness[i]*和 *pbest[i]*；

④比较每个微粒当前的适应度 *fitness[i]*与该微粒群所经过的最好位置的适应度 *gbest_fitness*，如果 *fitness[i]* > *gbest_fitness*；则更新 *gbest_fitness*；

⑤如果达到结束条件(足够好的位置或最大迭代次数)，则结束；否则转步骤②。

基本微粒群算法的流程如图 5.1 所示。

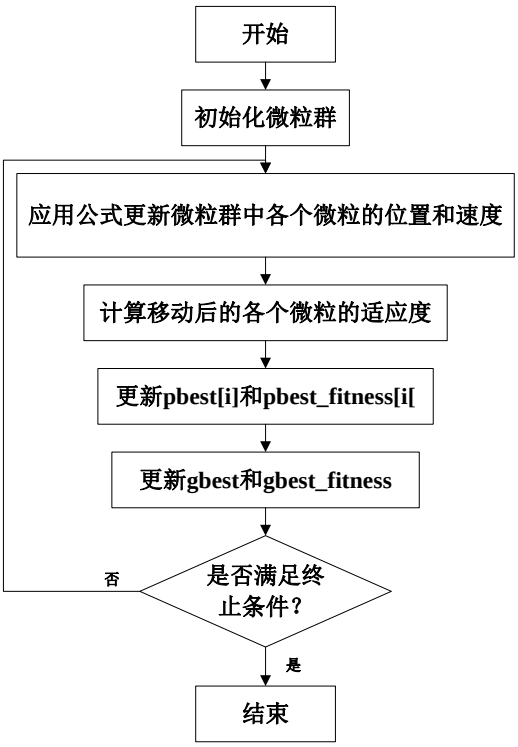


图 5.1 微粒群算法流程图

5.1.3 微粒群算法的改进

标准微粒群算法在实际应用过程中，对于某些多峰函数优化的问题，由于进化过程中种群多样性损失过快，PSO 算法容易陷入局部极值，引起算法过一早收敛。针对这一问题，很多学者都提出了相关的改进方法。到目前为止，对 PSO 算法的改进非常多，例如压缩因子 PSO 算法、混合 PSO 算法、保证种群多样性的 PSO 算法、免疫 PSO 算法、协同 PSO 算法、社会分工 PSO 算法、离散 PSO 算法等等。

在式（5-1）中，惯性权重 w 的选取至关重要。惯性权重较大有利于提高算法的全局搜索能力，较小则会增强算法的局部搜索能力，只有全局和局部搜索能力达到平衡，算法的效果才最为理性，因此惯性权重的取值直接影响着算法的搜索性能。针对惯性权重的选取问题，研究人员先后提出了线性递减权值(LDIW)策略、模糊惯性权值(FIW)策略和随机惯性权值(RIW)策略等。由于计算机博弈程序的特殊性，本文将采用基于聚焦距离变化率动态改变惯性权重的改进微粒群算法[74]。

首先给出平均聚集距离和最大聚集距离的定义[75]：

平均聚集距离 *MeanDist*：

$$MeanDist = \frac{\sum_{i=1}^{m}\sqrt{\sum_{t=1}^{D}(p_{td}-x_{id})^2}}{m} \tag{5-3}$$

最大聚集距离 *MaxDist*：

$$MaxDist = \max_{i=1,2,\ldots,m} \sqrt{\sum_{t=1}^{D} (p_{td} - x_{id})^2} \tag{5-4}$$

其中 m 为微粒群中的微粒数，D 为搜索空间的维数，p_{td} 为粒子群目前搜索到的最优位置，x_{id} 为每个粒子目前搜索到的最优位置。

那么聚焦距离变化率的定义如下：

微粒目前的聚焦距离的变化率定义为式(5-5)。

$$k = \frac{MaxDist - Mea}{MaxDist} \tag{5-5}$$

每迭代一次就计算此次得到的平局聚焦距离和最大聚焦距离，这样就可以得到此次的聚焦距离的变化率，据此能够判断此次的粒子是应该提高其全局搜索能力还是需要提高其局部搜索能力，然后对它的惯性权重进行调整。

在使用式（5-1）进行计算之前，先计算出目前的聚焦距离的变化率 k，然后使用 k 通过式（5-6）来确定惯性权值 w。

$$w = \begin{cases} (a_1 + |r|/2.0)|\ln k|, & |k| > 1 \\ a_1 a_2 + |r|/2.0 & 0.05 \le |k| \le 1 \\ (a_1 + |r|/2.0)\dfrac{1}{|\ln k|}, & |k| < 0.05 \end{cases} \tag{5-6}$$

其中 $\alpha_1 = 0.3$，$\alpha_2 = 0.2$，r 为一个[0,1]间均匀分布的随机数。该选择策略即随机地选取 w 值，使 w 随聚焦距离的变化率自适应地调整，使之较好地适应复杂的实际环境，从而可以更灵活地调节全局搜索与局部搜索能力。当聚焦距离变化率较大时表明粒子的最大聚焦距离和平均聚焦距离相差较大，此时粒子的全局搜索较差，故应使粒子尽快地进入全局搜索，相反我们即应该提高粒子的局部搜索能力。另外，惯性权值的随机性在一定程度上与遗传算法的变异算子较为相似，这将有助于保持种群的多样性。那么，改进后的微粒群算法流程图如图 5.2 所示。

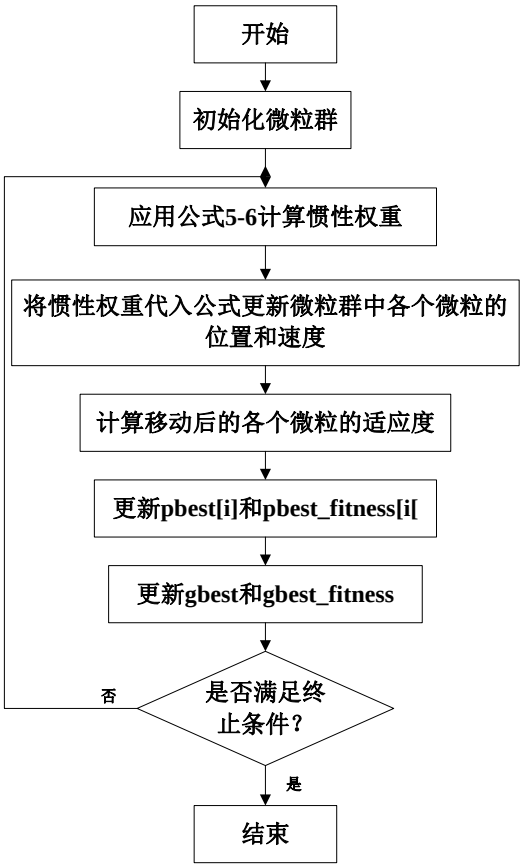


图 5.2 改进微粒群算法流程图

5.2 评估函数及其应用改进

在使用微粒群算法对评估函数的 5 个参数进行寻优的过程中，与遗传算法类似的是，需要计算个体的适应度。那么与上一章所使用的遗传算法相同，在使用微粒群算法进行寻优的过程中也需要采用互相对弈的方式，通过比赛的得分来评判微粒所处位置的好坏。因此，为了使 2 组参数能够进行互相对弈，首先需要对博弈程序进行改进。显然原来的博弈程序只有一套评估函数，程序在与对手对弈的过程中，只通过该套评估函数以及某一组确定的参数对局面进行估值。所以，我们所需要做的改进是，将原来的博弈程序扩展出另外一套评估函数，即使该博弈程序中拥有 2 套评估函数模块，这样在进行自己与自己对弈训练的过程中，程序通过 2 套评估函数，读取不同的 2 组参数，从而实现该 2 组参数的比赛，进行自学习训练。

5.3 微粒群算法的实施

微粒群算法与遗传算法相类似，也需要计算各个微粒的适应度。但是，与遗传算

法不同的是，遗传算法需要计算出每个个体确切的适应度大小，根据确切的适应度大小，计算每个个体被选中遗传到下一代的几率，而微粒群算法则不同，它不要求计算出每个微粒适应度具体的值，而只需要知道：

①每个微粒当前的适应度与该微粒历史最大适应度谁大谁小，并不需要得到具体的数值，如果当前的适应度大，那么则要更新该微粒的历史最优位置；

②当前所有微粒中适应度最大的微粒所处的位置，与微粒群中所有微粒的历史最优位置孰优孰劣，如果当前微粒群中适应度最大的微粒所处的位置优于当前记录的历史最优位置，那么则更新历史最优位置。

根据微粒群算法以及计算机博弈程序的特殊性，本文将微粒群算法应用于六子棋博弈程序评估函数参数寻优的具体实施流程如下：

①将原来的六子棋博弈程序进行扩展，使其具有两套评估函数模块，并能够使程序自身能够使这两套评估函数读取不同的参数组进行自身的对弈。

②为了和遗传算法进行对比，因此与上一章中遗传算法的种群大小保持一致，设定的微粒群中微粒的个数 $m = 49$；待优化的参数有 5 个，因此搜索空间的维度 $D = 5$。随机生成 49 个微粒，初始化 49 个微粒的初始位置 $X[i]=\{x_1,\ x_2,\ x_3,\ x_4,\ x_5\}$以及速度 $V[i]=\{v_1,\ v_2,\ v_3,\ v_4,\ v_5\}(i = 1, 2, \ldots, 49)$。

③初始化微粒群中各个微粒的历史最优位置 *pbest[i]*和所有微粒的历史最优位置 *gbest*。由于是初始化阶段，因此 *pbest[i]*为各个微粒初始化的位置。*gbest* 初始化的方法是：首先从微粒群中选取 2 个微粒进行对弈，失败的一方淘汰，胜出的一方再从其余没有进行过对弈的微粒中选取一个进行对弈，直到最后胜利的一个微粒出现，该微粒的位置即设置为 *gbest*。

④根据式（5-2）更新各个微粒的位置。

⑤更新 *pbest[i]*和 *gbest*。用各个微粒的当前位置代表的参数与各个微粒的历史最优位置 *pbest[i]*进行对弈，如果当前位置获胜，则更新 *pbest[i]*，并且让这个获胜的位置代表的参数组再与 *gbest* 代表的参数组进行对弈，如果又获胜，则更新 *gbest*。

⑥根据式（5-3）（5-4）（5-5）（5-6）计算惯性权重 *w*。

⑦根据式（5-1）更新速度 *V[i]*。

⑧重复④—⑦，直到满足结束条件。

5.4 实施结果的评估

5.4.1 优化效果的实验与分析

根据表 5.1 给定的参数初始化微粒群，初始微粒群中各个微粒的位置与上一章中使用遗传算法时的初始种群保持一致。实验开始阶段，博弈程序也是没有任何智能，杂乱无章的下棋。在所有微粒经过 15 代的运动以后，得到的微粒位置片段如图 5.3 所示，得出的历史最优位置为 $X = \{-124, 275, 1229, 6518, 2763\}$。转换为评估函数的参数组后，博弈程序具有明显的智能。在与之前人工确定的参数组进行对弈实验，微粒群算法寻得的参数组能够达到 70%的胜率，详细结果如表 5.2 所示，其中一次对弈的棋局如图 5.4 所示。

表 5.1 微粒群算法参数设置

种群大小	维数	初始位置	惯性权重 w	速度 V	C_1	C_2	α_1	α_2	r
49	5	[0,4000]随机	根据公式产生	[-200,200]的随机值	2.05	2.05	0.3	0.2	[0,1]内的随机值

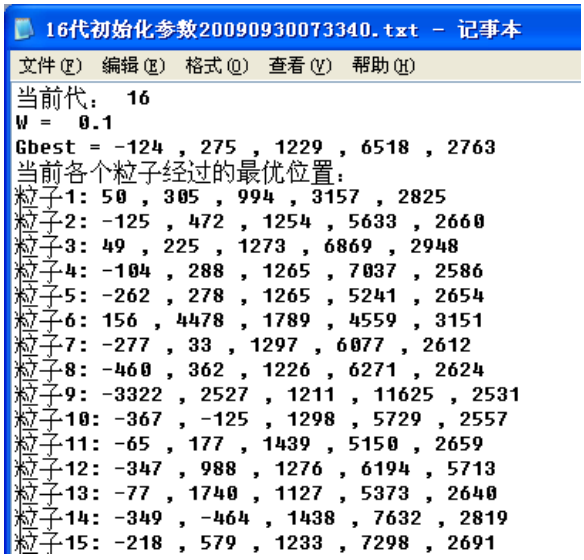


图 5.3 微粒群算法优化片段

表 5.2 PSO 与人工确定的参数对弈结果

对弈次数 / 对弈双方	100	200	300	平均胜率
人工确定的参数	34 胜	56 胜	89 胜	179
PSO	66 胜	144 胜	211 胜	421
PSO 胜率	66%	72%	70%	70%

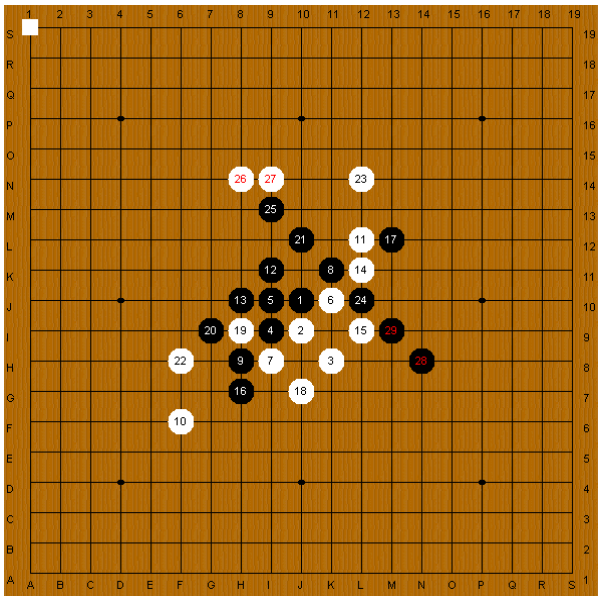


图 5.4 微粒群算法优化 15 代后的棋局

5.4.2 PSO 与 GA 对比实验的结果与分析

经过实验和分析可以看到，遗传算法和微粒群算法都可以使六子棋博弈程序从完全没有智能，胡乱落子通过自学习变成具有较强的棋力，能够对自己的落子位置做出比较合适的选择。因此，两种算法都可以用来对六子棋博弈系统中评估函数的参数进行优化，当然也可以扩展应用到其它棋类博弈程序当中。不管是用遗传算法优化过的参数，还是用微粒群算法优化过的参数，都可以轻松的打败过去我们手工凭自己的经验反复测试确定的参数组，胜率分别达到了 81%和 70%。

究竟哪种算法得到的参数组更优？本文进行了下面的实验，让用遗传算法经过 30 代进化的参数与用微粒群算法经过 15 代搜索得到的参数组进行对弈。实验结果如表 5.3 所示：

表 5.3 PSO 与 GA 对弈结果

对弈次数 / 对弈双方	100	200	300	平均胜率
GA	32 胜	61 胜	87 胜	180
PSO	68 胜	139 胜	213 胜	420
PSO 胜率	68%	70%	71%	70%

从表中可以看到，微粒群算法得到的参数组获得了 70%的胜率，其中一场对弈的棋局如图 5.5 所示，其中白方为微粒群算法得到的参数组。这样可以发现，微粒群算法得到的参数组虽然在与人工确定的参数进行对弈的时候，胜率只达到了 70%，没有

采用遗传算法得到的参数组 81%的胜率高，但是却能够在与采用遗传算法得到的参数组对弈的过程中取得较高的胜率。因此，就目前的状况来说，微粒群算法优化的效率更高，搜索全局最优的能力更强。但是该结论只是在本文所设计的具体的实验平台下得出的，对于其他学者和研究人员设计的不同平台，采取不同博弈策略的情况下，微粒群算法是否更优，还需要做进一步的研究和证明。

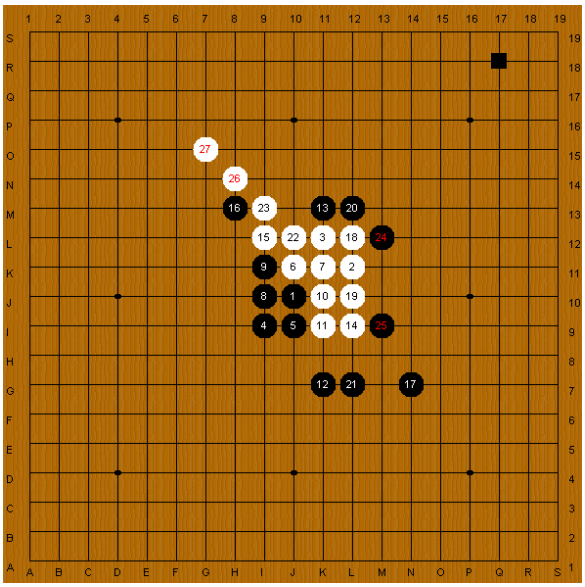


图 5.5 两种算法得到的参数组对弈的棋局

5.5 小结

本章使用基于聚焦距离动态改变惯性权重的改进微粒群算法对评估函数参数进行调整和优化，经过离线学习以后，取得了效果，证明微粒群算法可以用来解决博弈系统评估函数的参数优化的问题。

6 结论与展望

6.1 研究工作小结

本文以现有的计算机博弈理论的关键技术为基础，实现了六子棋博弈机器人的“大脑”部分，文章所依托的科研项目的最终目标是构造一个能够在物理棋盘上利用“手臂”和“眼睛”与人类下棋的博弈机器人。本文所做的主要工作有：

1、针对以往采用棋形的方法对局面进行估值，棋形复杂，判断难度大的问题，规范了“路”的策略。采用的路的策略以后，对局面的统计和评判得到很大的简化。

2、对于“路”的思想，设计了新的评估函数。

3、由于本研究的实现和验证实验等工作，都是在导师所带领的团队所开发的六子棋计算机博弈平台上进行的，因此，所做的工作还包括对该博弈平台进行改进，使该系统具有两套评估函数，以方便后面通过智能算法对评估函数进行寻优。

4、采用了改进的遗传算法对评估函数的参数进行优化，解决了以往采用人工凭经验手动调整评估函数参数，不仅繁琐而且难以达到全局最优的问题。

5、考虑到微粒群算法具有概念简单和易于实现的优点，在短短几年中得到很大发展，并在许多领域取得了成功的应用，因此，在本文中采用基于聚焦距离动态调整惯性权重的微粒群算法对评估函数的参数组进行搜索，取得了不错的效果。

本文的创新点有以下几个方面：

1、受到其他学者的启发，对“路”的思想进行了规范，使棋局的局面表示得到简化。

2、对以前学者使用遗传算法进行评估函数参数寻优的方法进行改进，避免了算法收敛过早的问题，同时达到了优化的效果。

3、创新的将微粒群算法应用到六子棋计算机博弈系统的评估函数参数优化问题上，并取得了效果。

6.2 存在的问题与不足

到目前为止，本系统已基本实现，而且具有一定的棋力。但是本系统还存在以下问题和不足：

1、微粒群算法搜索到了优秀的参数组，但是观察参数发现存在负数的问题，显然这是不合逻辑的。因此，在微粒群算法的应用上还需要进一步的调整，做一定的限制。

2、在使用遗传算法和微粒群算法进行搜索的过程中，都是在同一套博弈系统中进行自学习从而优化参数的，针对的是本系统中的“思考”逻辑，而事实上，在与其他人制作的博弈系统或是人类棋手进行对弈的过程中，这些对手的逻辑或者策略发生改变，这时候的参数组也许对于他们并不是优秀的。因此，在训练参数的时候，应该使用更多其他的博弈系统或是人类棋手来训练参数。

3、本文主要内容是通过训练评估函数的参数组来达到提升系统智能。事实上，还有其他许多地方都需要进一步的优化，例如提高搜索效率，加入开局库等方法。

4、博弈机器人的其它诸如手臂、视觉等部分还有待于设计和实现。

致谢

本论文的完成凝聚了很多人辛勤的劳动和汗水，在此，谨向给予我无私帮助的老师和同学们致以诚挚的谢意!向辛勤培育我成长的母校和领导们致以由衷的感谢和崇高的敬意!

感谢我的导师张小川教授。在课题研究过程中，我得到了张老师的全力支持和精心辅导。张老师不仅为我提供了良好的学习环境，完备的试验设施，在硕士论文的选题、研究过程中更是得到了张老师的无限鼓励、鞭策和帮助。在论文的撰写过程中，张老师不厌其烦，逐字逐句帮助我审阅和修改，更让我感受到了张老师的认真严谨以及对我的无私帮助。在三年的研究生学习阶段，导师渊博的知识背景、开阔的研究思路、严谨的治学态度、敏捷的思维、勤奋进取的精神都是我的学习榜样。值此论文完成之际，谨向张老师表示衷心的感谢和无比的敬意!

感谢我的女友张菲，在生活上给予无限的关爱。在我忙于实验和撰写论文的时候，她给我端茶递水，添置生活用品，照顾我。在我偷懒懈怠不务正业的时候，她督促我努力学习。在她的帮助和监督下，我不仅通过了英语六级考试，获得了软考中级证书，还在全国研究生数学建模竞赛以及校内活动中多次获奖。这所有的荣誉都掺杂着她的汗水。

感谢在计算机应用技术专业研究生阶段一起度过美好时光的同学们。刘羽、陆嘉希、万家强等同学不仅在学习讨论中给予我很多启发和帮助，并且给我的课余生活添加了许多绚丽的色彩。在学习上，李丙涛同学用他丰富的专业知识给了我许多帮助。还有李为、李翠珠、梁俊华、桂袁义、崔英志等同学也给了我许多帮助。

感谢实验室的陈恋昶、刘婵桢、郝艳伟、唐艳、魏建新等师弟师妹们，在日常的学习与生活中给我带来了开心与快乐!

最后，感谢答辩委员会的各位老师评审我的论文和出席我的毕业答辩会。

陈光年

2010 年 4 月于重庆

参考文献

[1] 陆汝铃. 人工智能[M]. 北京:科学出版社,1995.
[2] [美]Nils. J Nilsson. 人工智能[M]. 北京:机械工业出版社, 2000.
[3] 蔡自兴, 徐光佑. 人工智能及其应用[M]. 北京:清华大学出版社, 2003.
[4] 罗伯特•吉本斯. 博弈论基础[M]. 北京:中国社会科学出版社, 1999.
[5] Frederic Friedel, michael 译注. 电脑国际象棋简史[EB/OL]. http://www.chessit.net/file_topic/computerchess/c_briefhistory.htm.
[6] 邵桂芳. 基于人工生命方法的自主智能体行为选择研究[D]. 重庆：重庆大学
[7] Seiji Yamada. Adaptive Action Selection without Explicit Communication for Multi-robot Box-pushing[A]. IEEE/RSJ International Conference on Intelligent Robots and Systems[C]. 1999.
[8] 谢汝林,李祖枢.2003. 基于优先度的人工生命体行为选择研究[J],广西师范大学学报,21(1):1-6.
[9] 李祖枢,谢汝林,张小川,邵桂芳.2005.人工生命体行为选择及其进化研究[J],模式识别与人工智能,18(3).
[10] 李祖枢.2005.基于图式理论的人工生命体行为选择体系结构研究(特邀报告)[J],第二届人工生命及应用专题学术会议论文集.
[11] Guifang Shao,Zushu Li.2008.The behavior selection of artificial life based on evaluation-tradoff structure[J], proceedings of the 7th world congress on intelligent control and automation,820-824.
[12] David N. L. Levy, eds. Computer Games[J]. New York: Springer New York Inc, 1998, 25(3): 335-365.
[13] Holland J H. Adaptation in Natural and Artificial Systems. Ann Arbor[J], MI: The University of Michigan Press, 1975.
[14] Mchelewicz Z. Genetic Algorithms + Data Structure = Evolutionary Programming[J]. Springer - Verlag, 1996.
[15] Fogel L J, et al. Artificial Intelligence through Simulation Evolution[J]. Chichester: John Wiley, 1966.
[16] Fogel D B. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence[J]. IEEE Press, 1995.
[17] Koza J R. Genetic Programming: On the Programming of Computers by Means of Netural Selection[J]. Cambridge, MA: MIT Press,1992.
[18] Dorigo M, Stutzle T. Ant Colony Optimization[J]. The MIT Press. Cambridge, Masschusetts London, England, 2004.
[19] Kennedy J, Eberhart K. Particle Swarm Optimization[C]. In: Proc IEEE Int Conference on Neural Networks, 1995,1942-1948.
[20] David j. Kruglinski, Scot Wingo & George Shepherd. Programming Microsoft Visual C++.5th Edition[M], Microsoft Press, 1999.
[21] Marsland T A.Computer chess and search[D].Edmonton:University of Alberta,1991.
[22] 王小春. PC 游戏编程[M]. 重庆:重庆大学出版社, 2002.1–27.
[23] David j. Kruglinski, Scot Wingo & George Shepherd. Programming Microsoft Visual C++.5th Edition[M], Microsoft Press, 1999.
[24] J.Schaeffer, Distributed Game-tree Searching, Journal of Parallel and Distributed Computing, Vol.6,

1989.
[25] A.Plaat, J.Schaeffer, W.Pijls, A.de Bruin, Best-first Fixed-depth Minimax Algorithms[J]. Aritificial Intelligence. 1996.
[26] Rivest, R.L.Intelligence, Game Tree Searching by MinMax Approximation[J]. Artificial 1998, Vol.34, No.1
[27] Shannon. Claude E Programming a computer for playing chess [J]. Philosophical Magazine Vol. 41 1950 256-275.
[28] TODHUN TER I. A History of the Mathematical Theory of Probability from the Time of Passcal to that of Lap lace[M] New York: Chelsea Publishing, 1965: 105-108.
[29] BOREL E. The theory of play and integral equations with skew symmetric kernels[J]. Econometrica, 1953, 21: 91-117.
[30] Rivest, R-L Game Tree Searching by MinMax Approximation [J]. Artificial Intelligence, 1998, Vol.34, No.1
[31] Donald E, Knuth and Ronald W, Moore. An Analysis of Alpha-Beta Pruning[J]. Artificial Intelligence, vol.6, 1975:293-326.
[32] 陆汝铃. 人工智能(上册)[M]. 第一版. 北京: 科学出版社, 1989.8
[33] 林尧瑞，马少平.人工智能导论[M].1989 北京 清华大学出版社.
[34] Francis Dominic Laram.博弈编程指南[EB/OL].http://www.gamedev.net.
[35] Herik,H.J.van den,and Allis,L.V.(eds.)(1992).Heuristic Programming in Artificial Intelligence 3:the third computer Olympiad[M]. Ellis Horwood Ltd.,Chichester,England.
[36] Yen SJ, Chen J C, Yang T N. Computer Chinese Chess[J]. ICGA Journal, 2004, (3): 3-18.
[37] 张小川，舒良等，关于六子棋计算机博弈策略的探讨[J]. 中国人工智能进展(2007)，2007.12
[38] 肖齐英, 王正志. 博弈树搜索与静态估值函数闭[J], 计算机应用研究, 1997.
[39] 计算机博弈[EB/OL]. http://www.elephantbase.net/computer.htm
[40] 王凌. 智能优化算法及其应用[M], 清华大学出版社, 2001.
[41] 王小平,曹立明. 遗传算法--理论应用与软件实现[M]. 西安: 西安交通大学出版社, 2002.
[42] Holland J H. Adaptation in nature and artificial system[M] . Ann Arbor: The University of Michigan Press, 1975.
[43] 沈大旺, 张慧. 遗传算法综述[J]. 黑龙江科技信息. 2009 (28):100.
[44] 武兆慧，张桂娟，刘希玉．基于模拟退火遗传算法的关联规则挖掘[J]. 计算机应用，2005,25(5):1 009-1 011.
[45] Jeong II-Kwon, Lee Ju-Jang. Adaptive Simulated Annealing Genetic Algorithm for Control Applications[J]. International Journal of Systems Science,1996,27(2):241-253.
[46] BergyPaul K, Ragsdale CliffT, Hoskote Mangesh. ASimulated Annealing Genetic Algorithm for the Electrical Districting Problem[J]. Annals of Operations Research, 2003(121):33-35.
[47] 吕军，冯博琴，李波．免疫遗传算法及其应用研究[J]. 微电子学与计算机，2005, 22(6): 221-224.
[48] Coello Carlos A Coello, Cortes Nareli Cruz. Hybridizing A Genetic Algorithm with An Artificial Immune System for Global Optimization[J]. Engineering Optimization, 2004, 36(5):607-634.
[49] 黄聪明, 陈湘秀. 小生境遗传算法的改进[J].北京理工大学学报, 2004, 24(8): 675-678.
[50] 班书昊, 杨慧珠. 小生境遗传算法在孔隙介质反演中的应用研究[J]. 石油勘探与开发, 2005, 32(2):78-81.

[51] Wei Lingyun, Zhao Mei. A Niche Hybrid Genetic Algorithmfor Global Optimization of Continuous Multimodal Functions[J]. Applied Mathematics and Computation(NewYork), 2005, 160(3): 649-661.

[52] 王兴成, 郑紫微. 模糊遗传算法及其应用研究[J]. 计算机术语自动化, 2000,19(2):5-9.

[53] Yun Youngsu, Gen Mitsuo. Performance Analysis of Adaptive Genetic Algorithms with Fuzzy Logic and Heuristics[J]. Fuzzy Optimization andDecision Making, 2003, 2(2): 161-175.

[54] Pal Sankar K, Bhandari Dinabandhu. Genetic Algorithms with Fuzzy Fitness Function for Object Extraction UsingCellular Networks[J]. FuzzySets and Systems, 1994(2): 129-139.

[55] Song Y H, Wang G S, Johns A T, et al. Improved Genetic Algorithms with Fuzzy Logic Controlled Crossover and Mutation[J]. IEE Conference Publication, 1996, 427(1):140-144.

[56] 周晓, 胡以华, 陈修桥, 等. 混沌遗传算法及其在函数优化中的应用[J]. 计算机与数字工程, 2005,33(7):68-70.

[57] QingzhangLu, Guoli Shen, Ruqin Yu. A Chaotic Approach to Maintain the Population Diversity of Genetic Algorithm in Network Training[J]. Computational Biology and Chemistry, 2003,27(3):363-371.

[58] 周露芳, 古乐野. 基于量子遗传算法的二位最大熵图像分割[J]. 计算机应用,2005,25(8):1 805-1 807.

[59] Grigorenko I, Garcia ME. Calculation of the Partition Function Using Quantum Genetic Algorithms[J]. Physical A, 2002,313(3):463-470.

[60] A Aahin Mehmet, Tomak Mehmet. The Self-consistent Calculation of A Spherical Quantum Dot: A Quantum Genetic Algorithm Study[J]. Physica E: Low-dimensional Systems,2005,28(3):247-256.

[61] 徐金梧, 刘纪文. 基于小生境技术的遗传算法[J]. 模式识别与人工智能. 1999(1).

[62] 王骄,王涛,罗艳红,徐心和. 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. 2005, 26 (10).

[63] 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).

[64] Eshelman L J, Caruana R, Schaffer D. Biases in the crossover landscape[R]. in Proc. 3rd Int. Conf. Genetic Algorithms,1989: 10-19.

[65] Syswerda G. Uniform Crossover in Genetic Algorithms[R].in Proc. 3rd Int. Conf. Genetic Algorithms, 1989: 2-9.

[66] Srinivas M, Patnaik L M. On Modeling Genetic Algorithms for Functions of Unitation[J]. IEEE Trans Syst, Man, and Cybernitics, 1996, 26(6): 809-821.

[67] Kennedy J, Eberhart R. Particle swarm optimization[A].Proc IEEE Int Conf on Neural Networks[C].Perth,1995.1942-1948.

[68] Eberhart R, Kennedy J. A new optimizer using particle swarm theory[A].Proc6th Int Symposium on Micro Machine and Human Science[C].Nagoya,1995.39-43.

[69] Eberhart R.C, Kennedy J. A new optimizer using Particle swarm theory[A]. Proceedings of the sixth international symposium on Micro Machine and Human Science[C]. Nagoya Japan, 1995:39-43.

[70] Kennedy J. The Particle swarm: social adaptation of knowledge[A]. Proceedings of IEEE International Conference on Evolutionary Computation[C], Indianapolis, IN, 1997, 303-308.

[71] Eberhart R, Kennedy J. Discrete binary version of the particle swarm algorithm[A], proc IEEE Int Conf on Systems, Man and Cybernetics[C], Orlando, 1997, 4104-4108.

[72] 刘钊,陈建勋. 基于粒子群算法的足球机器人动作选择研究[J]. 武汉科技大学学报(自然科学

版). 2006, 29(1).
[73] Shi Y. and Eberhart R.C.(1998b). A modified particle swarm optimizer[A]. Proceedings of the IEEE International Conference on Evolutionary Computation[C], 69-73. Piscataway, NJ: IEEE Press.
[74] 任子晖, 王坚. 一种动态改变惯性权重的自适应粒子群算法[J]. 计算机科学. 2009, 36(2).
[75] 李宁, 孙德宝, 等. 带变异的粒子群优化算法[J]. 计算机工程与应用, 2004, 17.

个人简历、在学期间发表的学术论文及取得的研究成果

姓名	陈光年	出生日期	1984-06-06	籍贯	湖北省武汉市
获得学士学位时间、学校		2007 年 7 月毕业于湖北警官学院			
现所学学科、专业		计算机应用技术		入学时间	2007-09
学习（大学以上）及工作经历					
年 月— 年 月		就学的学校、专业/工作单位、职务			
2003,9——2007,7		湖北警官学院信息安全专业			
2007,9——2010,7		重庆理工大学计算机应用技术			

在学期间发表的学术论文及取得的研究成果（包括鉴定项目、获奖、专利）

序号	论文或成果、专利名称	全体作者（按顺序排列）	发表刊物或鉴定单位或获奖名称、等级或专利类别	时间
1	六子棋博弈的评估函数	张小川，陈光年	重庆理工大学学报（自然科学版），2010,2 期	2010-02

在学期间尚未发表但已被录用的学术论文

序号	论文名称	全体作者（按顺序排列）	拟发表刊物	预计发表时间