文章编号: 1673-9868(2007)11-0138-05

基于遗传算法的六子棋博弈评估函数参数优化①

李　果

重庆大学 自动化学院, 重庆 400044

摘要: 将博弈问题分解为搜索引擎、走法生成、评估函数和开局库4大模块, 再将自适应遗传算法引入到评估函数中, 并通过锦标赛算法对评估函数中的参数组合进行自动调整和优化, 设计并开发出基于上述方法的离线自学习系统.

关　键　词: 六子棋计算机博弈; 博弈树; 评估函数; 锦标赛算法; 遗传算法

中图分类号: **TP11**　　文献标识码: **A**

计算机博弈[1]是人工智能领域一个极其重要且极具挑战性的研究方向, 而棋类游戏则是计算机博弈的一个标准性问题. 各种搜索算法、模式识别及智能方法在计算机博弈中都可以得到广泛的应用.

目前, 五子棋、中国象棋等棋类游戏的计算机博弈算法研究已相对成熟, 六子棋作为一个刚刚兴起不久的棋类游戏, 其计算机博弈算法的研究还相对较少. 由于六子棋的特殊性, 每次一方走下两颗棋子, 其game-tree的复杂度可达10^{140}(按30手计算), 远大于五子棋, 其state-space复杂度可达10^{172}, 与围棋相当. 为此, 有必要对六子棋的计算机博弈算法展开更加深入的研究, 以促进六子棋运动以及计算机博弈的更进一步发展.

六子棋计算机博弈可以分解为4个主要部分: 搜索引擎、走法生成、评估函数和开局库.

评估函数是模式识别和智能算法应用最为广泛的领域. 不管多么复杂的评估函数, 都可以表示为一个多项式. 评估函数一般来说必须包括5个方面的要素, 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. 每一方面的值又由许多参数值构成的. 即使最简单的评估函数也有20多个参数, 将这些值线性地组合在一起得到最终的评估值. 由于六子棋无子与子的差别(对于某一方), 其评估函数只跟当前棋子的位置以及其周边棋子的位置状态有关, 是以上所述5要素的一个子集, 因而可以得到简化. 但评估函数的确定通常依赖于编程者的棋类知识, 这就带来很大的局限性, 通过手动调整其参数很难达到全局最优.

为了解决这个参数优化的问题, 在六子棋计算机博弈领域引入了遗传算法. 通过离线自学习训练得到其评估函数的最佳参数组合.

1　六子棋的常见棋型及其状态演变的计算机描述

要把遗传算法引入到六子棋的评估函数, 必须深入了解六子棋的一些常用基本棋型以便对六子棋的常

① 收稿日期: 2007-04-19
作者简介: 李　果(1981-), 男, 重庆人, 硕士研究生, 主要从事模式识别与智能系统的研究.

用基本棋型之间的内部联系做更深一步的研究. 下面分别给出六子棋的常用基本棋型以及基本棋型间的状态演变.

1.1 六子棋的棋型

六子棋常见棋型的定义、英文名称和符号表示如下:

【六连】: 在棋盘的纵向、横向或斜向的任意一条线上, 形成的 6 颗同色棋子不间隔地相连. 英文名称: Continuous six. 符号表示: C6.

【长连】: 在棋盘的纵向、横向或斜向的任意一条线上, 形成的 7 颗或 7 颗以上同色棋子不间隔地相连. 英文名称: Continuous long. 符号表示: Cl.

【活五】: 在同一直线上的 5 颗同色棋子, 符合"对方必须用两手棋才能挡住"的条件. "挡住"是指不让另一方形成六连或长连. 英文名称: Active five. 符号表示: A5.

【眠五】: 在同一直线上的 5 颗同色棋子, 符合"对方用一手棋就能挡住"的条件. 英文名称: Sleep five. 符号表示: S5.

【死五】: 在同一直线上的 5 颗同色棋子, 它们已无法形成【六连】或【长连】. 英文名称: Dead five. 符号表示: D5.

其它【活四】【眠四】【死四】等定义类似, 其符号表示分别为 A4、S4、D4、A3、S3、D3、A2、S2、D2, 除此之外, 还有个比较特殊的棋型:

【朦胧三】: 在同一直线上的 3 颗同色棋子, 符合"再下一手棋只能形成眠四, 但如果再下两手棋的话就能形成活五"的条件. 英文名称: Indistinct three. 符号表示: I3.

1.2 六子棋各棋型间的状态演变

上文已对六子棋的常用基本棋型做出了简单的定义, 每种棋型都包含不同的对应状态. 下面给出六子棋中各棋型的状态演变情况:

$A5^{\rightarrow}(S5 \cup C6 \cup A4 \cup S4)$　$S5^{\rightarrow}(C6 \cup A5 \cup A4 \cup A3 \cup S5 \cup D5 \cup D4 \cup S3 \cup I3 \cup D3 \cup S2 \cup D2)$

$D5^{\rightarrow}(S3 \cup S4 \cup D3 \cup D2)$　$A4^{\rightarrow}(A5 \cup S4)$　$S4^{\rightarrow}(S5 \cup A5 \cup A3 \cup I3 \cup A2)$

$D4^{\rightarrow}(D5 \cup D3 \cup S3 \cup S2 \cup I3)$　$A3^{\rightarrow}(A4 \cup S4 \cup S3 \cup D3 \cup S2)$　$I3^{\rightarrow}(S4 \cup S3 \cup D3 \cup S2 \cup D2)$

$S3^{\rightarrow}(S4 \cup D4 \cup D3 \cup D2)$　$D3^{\rightarrow}(D4)$　$A2^{\rightarrow}(A3 \cup S2)$　$S2^{\rightarrow}(S3 \cup D2)$　$D2^{\rightarrow}(D3 \cup A2 \cup S2)$

其中, 符号"∪"表示"或"的关系. 上述的演变情况包括了本方走一步和对方走一步时的情况, 而且每一种演变情况包括不止一种的对应状态, 当然状态演变为自身的情况都存在, 例如, $A5^{\rightarrow}A5$ 这种情况是肯定存在的, 但没有给出.

这里, $A5^{\rightarrow}(S5 \cup C6 \cup A4 \cup S4)$ 表示黑白双方的某一方在现有 A5 棋型的情况下, 当己方或对方落下一子后, 会转变成 S5 或 C6 或 A4 或 S4 的棋型. 当然, 每一种棋型包含不同的对应状态. 以 $A5^{\rightarrow}S5$ 为例, 可能包含不同的棋型对应状态.

把棋型之间演变情况的不同对应状态作为一个参数, 那么所有可能的演变情况中的不同对应状态就组成了参数组. 给每种状态赋予不同的值, 组成不同的参数组, 形成原始种群, 这样就可以用遗传算法来做参数的调整和优化.

2　基于遗传算法的六子棋评估函数

已经有人尝试着把遗传算法引入评估函数[2], 试图学习并发现棋子之间内部的动态联系, 但其效果并不理想. 本文应用遗传算法做六子棋评估函数的参数调整和优化, 并且为了加强遗传算法的局部搜索能力, 在进化过程中采用了改进的遗传算法——自适应遗传算法.

2.1 六子棋评估函数的确定

上文已经提到, 六子棋的评估函数只是评估函数 5 要素的一个子集, 只跟当前棋子的位置以及其周边

棋子的位置状态有关, 因而可以得到简化. 首先从当前落子和当前棋盘状态出发, 以当前落子为中心, 分别从横向、纵向、左斜向、右斜向4 个方向定长地扫描步长为 13 的点, 得到 4 个方向的状态数组 horizonalCodeStatus, verticalCodeStatus, leftSlopeCodeStatus, rightSlopeCodeStatus, 每个数组的长度为 13. 数组中的每一位有 4 个值: - 1, 0, 1, 2, 分别表示棋盘外、黑棋、白棋、无子的情况. 其中, 当前的落子在状态数组的第 7 位. 例如一个黑棋 A3$\rightarrow$A 4 的状态数组为{ 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2}. 然后通过下式将状态数组映射为一个数值 value:

$$value += value \times 4 + aStepStatus[i] + 1 \qquad i = 0, 1, \cdots, 12$$

其中, aStepStatus 表示 4 个方向中其中一个方向的状态数组, 每一个 value 将对应一个 stepValue, stepValue 的值首先靠编程者的经验人为的给出. 程序预先把 value 作为"键", 把 stepValue 作为"值"存储在一个哈希表中. 查找哈希表, 通过 value 分别得到 4 个方向的 stepValue: stepHorizonalValue, stepVerticalValue, stepLeftSlopeValue, stepRightSlopeValue. 如果 value 在哈希表中不存在, 其对应的 stepValue 默认为 0, 表示此时的 value 对于棋局影响不大或几乎没有影响. 最终当前落子的评估函数:

stepValue= stepHorizonalValue+ stepVerticalValue+ stepLeftSlopeValue+ stepRightSlopeValue

2.2 适应度函数的计算 ——锦标赛算法

遗传算法[3] 在进化搜索中基本不需要外部信息, 仅以适应度函数(fitness function)为依据, 利用种群中每个个体的适应度函数来进行搜索. 一般而言, 适应度函数是由目标函数变换而成的. 但是六子棋的评估比较复杂, 很难像一般优化问题那样找到真实准确的目标函数, 因此本文设计了一个专门用于棋类优化问题的适应度函数计算方法——锦标赛算法.

首先, 对已开发的六子棋计算机博弈软件进行拓展, 拓展后的程序包含两个评估函数, 但是搜索引擎、走法生成都是共用的. 两个评估函数通过读取不同的参数组来进行计算机与计算机的对弈, 并决出胜负. 这样, 就可以通过竞赛的方式来确定下棋双方的适应度, 也就是参数组的优劣. 在相同适应度的初始状态下, 赢的一方对适应度进行加运算(奖赏), 输的一方对适应度进行减运算(惩罚), 和棋则不操作. 这样就解决了适应度函数不好确定的问题, 实现了通过适应度函数来决定个体的优劣程度, 体现了自然进化中的优胜劣汰原则.

遗传算法处理的对象主要是个体[4-5]. 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, 也就是上文提到的某一个状态数组的 stepValue. 将染色体串解码到评估函数中即可求得评估值.

2.3 遗传操作过程

2.3.1 锦标赛选择

锦标赛模式[6] 的缺点在于速度过慢. 为了加快速度对标准锦标赛做了改进. 将包含 m 个个体的种群随机分成 n 组, 分别记为 Group$[i]$ $(i= 1, 2, \cdots, n)$, 每组包含 m/n 个个体. 分别在各组内进行锦标赛训练, 得到每个组的冠军, 分别记为 Champion$[i]$ $(i= 1, 2, 3, \cdots, n)$, 其为筛选出的最佳. 从而对 n 个冠军相互之间做交叉和变异操作产生 $m- n$ 个新个体, 分别记为 NewBody$[i]$ $(i= n+ 1, n+ 2, \cdots, m)$, 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. 经过优化后, 以包含 20 个个体的种群为例, 在保证优胜劣汰原则的基础上, 可以节省约 80%的时间.

2.3.2 均匀交叉

交叉方法很多, 有单点交叉、多点交叉、顺序交叉、循环交叉等等. 为了使交叉在便于操作的同时更加广义化, 选用均匀交叉, 参数之间的间隔点作为潜在的交叉点. 均匀交叉根据交叉率 P_c 随机地产生与参数个数等长的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. 通过这个掩码和选择的父个体一起确定子个体.

2.3.3 变　　异

变异是指以等于变异率 P_m 的概率改变一个或几个基因, 对于二进制串来说, 就是根据变异率来实现基因的 0– 1 翻转. 变异是一种局部随机搜索, 与选择/ 交叉算子结合在一起, 保证了遗传算法的有效性, 使遗传算法具有局部的随机搜索能力, 同时使得遗传算法保持种群的多样性, 以防止出现非成熟的收敛.

2.4 改进的遗传算法

在整个遗传算法实现的过程中, 交叉率 P_c 和变异率 P_m 的选择是影响遗传算法行为和性能的关键所在, 直接影响算法的效率以及收敛性. P_c 越大, 新个体产生的速度就越快, 然而 P_c 过大时遗传模式或信息被破坏的可能性也越大, 使得具有高适应度的个体结构很快就会被破坏; 但是如果 P_c 过小, 会使搜索过程缓慢, 以至停滞不前, 得不到适应度较高的个体. 对于变异率 P_m, 如果 P_m 取值过小, 就不容易产生新的个体结构; 如果 P_m 取值过大, 那么遗传算法就变成了纯粹的随机搜索算法, 失去了其本来的意义. 目前, 还没有通用的一次性确定 P_c 和 P_m 的方法. 针对不同的优化问题, 需要反复通过实验和调试来确定 P_c 和 P_m, 这是一件非常繁琐的工作. 为此, 自适应遗传算法[7], P_c 和 P_m 能够随着适应度自动改变.

P_c 和 P_m 的计算公式如下:

$$p_c = \begin{cases} p_{c1} - \dfrac{(p_{c1} - p_{c2})(f' - f_{avg})}{f_{max} - f_{avg}} & f' \geqslant f_{avg} \\ p_{c1} & f' < f_{avg} \end{cases}$$

$$p_m = \begin{cases} p_{m1} - \dfrac{(p_{m1} - p_{m2})(f_{max} - f)}{f_{max} - f_{avg}} & f' \geqslant f_{max} \\ p_{m1} & f' < f_{max} \end{cases}$$

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.

经过改进的自适应遗传算法中的 P_c 和 P_m 能够提供相对某个解的最佳 P_c 和 P_m, 在保持群体多样性的同时, 保证了遗传算法的效率和收敛性.

3　实验设计及实验结果

为了测试进化效果, 找出较好的参数组, 本文设计了一个测试实验, 分别取出 50 代、100 代、150 代、200 代、250 代、300 代的冠军以及两个随机选手组成一个小组, 进行单循环比赛. 每两个对手之间都要进行互换先后手的两场比赛, 胜者积分加 1, 负者积分减 1, 平局双方积分不变, 最后积分情况如图 1.

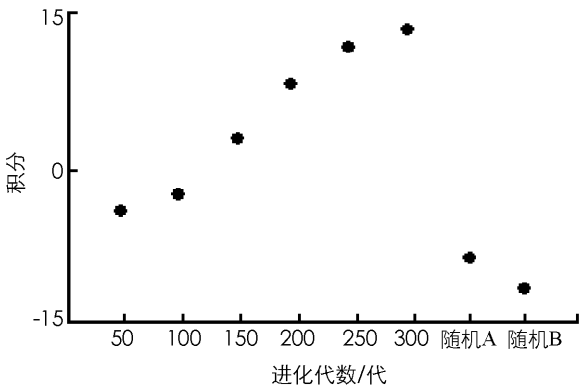

图 1　比赛结果

从图中可以看出, 得分随着代数的增加而单调增加, 而参加比赛的随机选手基本上没有获胜的机会. 这里给出了进化到 300 代时的训练结果. 用 300 代的参数组和完全用经验设定的参数组的程序进行了比赛, 结果用进化到 300 代的参数组的胜率明显占优.

5　结　　论

在六子棋的评估函数中, 引入了自适应遗传算法优化参数组合, 加强了传统的遗传算法的搜索能力, 避免了传统的爬山法容易陷入局部最优的缺点, 也不像模拟退火算法那样收敛过慢. 它同时维护一组参数, 既可以继承已有参数中的优良部分, 又可以避免因局部震荡而失败, 并且具有较高的效率和收敛速度, 可以最大限度的发挥评估函数的作用, 也为开发高质量的六子棋计算机博弈程序打下了坚实的基础.

参考文献:

[1] 王小春. PC 游戏编程 [M]. 重庆: 重庆大学出版社, 2002: 1- 27.

[2] Lorenz D, Markovitch S. Derivative Evaluation Function Learning Using Genetic Operators [A]. Proceedings of theAAA I Fall Symposium on Games: Planing and Learning [C]. New Carolina, 1993: 106- 114.

[3] Holland J H. A Daptation in Nature and Artificial System [M]. Ann Arbor: The University of Michigan Press, 1975.

[4] 米凯利维茨. 演化程序遗传算法和数据编码的结合 [M]. 北京: 科学出版社, 2000: 20- 23.

[5] 李敏强, 遗传算法的基本理论与应用 [M]. 北京: 科学出版社, 2002: 1- 15.

[6] 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: 195- 210.

[7] Angeline J, Pollack B. Competitive Environments Evolve Better Solution for Complex Tasks [A]. Proceedings of the Fifth International Conference on Genetic Algorithms [C]. SanMateo: Morgan Kaufman Publishers, 1993: 264- 270.

Adjustment and Optimization of the Connect6 Evaluation Function Parameters Based on Genetic Algorithm

LI Guo

College of Automation, Chongqing University, Chongqing 400044, China

Abstract: Genetic Algorithm(GA) is used in the computer game of Connect6. The problems of the game are divided into four parts: search engine, move generator, evaluation function and starting steps database. Then the modified GA is applied in the evaluation function, so as to automatically adjust and optimize its parameters through the Tournament Algorithm. At last, a self-learning system is designed and developed based on the method above.

Key words: computer game of connect6; game tree; evaluation function; tournament algorithm; genetic algorithm

责任编辑 张 枸