6 子棋启发式搜索算法的优化与设计

张　颖
（重庆大学 自动化学院，重庆　400044）

摘　要：将 6 子棋计算机博弈分为数据表示、界面、搜索引擎和评估函数 4 大模块予以实现，引入并优化了启发式搜索算法．提出了 6 子棋棋形的一种新的表示方法，为应用遗传算法奠定基础．仿真试验和实际比赛结果证明：该方法有效、可行．

关键词：计算机博弈；启发式搜索；评估函数；遗传算法；锦标赛方法
中图分类号：TP 11　　**文献标识码**：A　　**文章编号**：1001-988 X (2008)04-0025-06

Optimization and design of connect6 heuristic searching algorithm

ZHANG Ying
(College of Automation, Chongqing University, Chongqing 400044, China)

Abstract: Connect6 game system is divided into 4 parts: UI, data structure, search engine and evaluation function, and then the heuristic search is applied to optimize the searching engine. A new expression of connec6 chess type is introduced, which makes the genetic algorithm becoming exercisable. The results of experiment show that methods above are effective in getting better parameters and improving the AI of connect6 game.

Key words: computer game system; heuristic search; evaluation function; genetic algorithm; tournament algorithm

在人工智能领域始终将棋类的机器博弈作为常用的研究平台之一[1]．以棋类游戏为研究和验证平台，各种搜索算法、智能方法在计算机博弈中都可以得到广泛的应用．

由于 6 子棋的特殊性，每轮一方走 2 颗棋子增加了博弈分支因素，其 game-tree 的复杂度可达 10^{140}（按 30 手计算），state-space 复杂度可达 10^{172}，与围棋相当[2]．6 子棋具有高复杂度的同时却有着简单的规则和潜在的公平性，这使它成为一个新的研究热点．从其复杂度可以看出，信息的有效表示和搜索算法的优化对平台“智能”程度影响很大．本文引入一种新的棋形表示方法和启发式搜索算法对 6 子棋评估函数和搜索引擎部分做出优化．

1　问题的描述

1.1　规则

基本规则：由黑白两方，双方各执黑子和白子，黑先白后，采用围棋的 19 路棋盘，除黑方第一手棋先行 1 颗子外，之后双方轮流按照每手连续行棋 2 子，以先连成 6 子者为赢家．

1.2　问题的分解

本文针对上述规则，将 6 子棋计算机博弈问题分解为 4 个子问题：数据表示、界面、搜索引擎和评估函数[3]．通过几个模块的实现和相互衔接构成一个完整的机器博弈问题，并引入智能算法，提高其搜索速度和效率，为赢得比赛奠定基础．

2　数据结构和定义

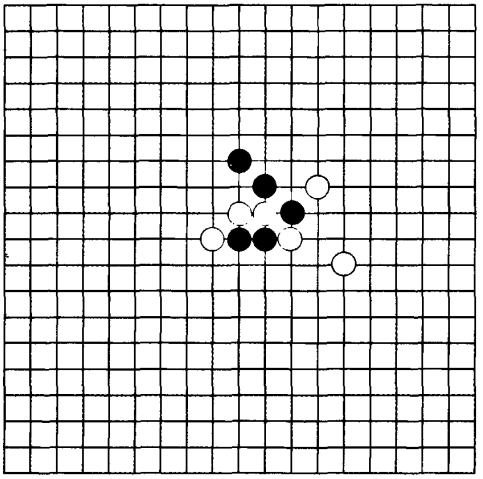
在计算机系统中，问题的背景信息和相关领域将决定系统的初始状态、算法选择和目标状态．提高 6 子棋机器博弈问题赢得比赛的概率，离不开高效的搜索算法．因此，本文借助 6 子棋的背景信息和机器博弈背景信息，以此作为提高搜索效率的启发信息．按照文献[4]，利用启发信息的搜索方法

收稿日期：2008-04-25；**修改稿收到日期**：2008-06-16
作者简介：张颖（1981—），男，甘肃兰州人，硕士研究生．主要研究方向为计算机博弈和智能系统．
E-mail: zy2rl@sina.com

叫做启发性搜索方法. 为进一步离散化、数量化6子棋机器博弈问题，以便计算机模拟实现，本文将6子棋机器博弈进程分别表示为3种离散数据结构：棋局状态，棋子状态和棋形. 下面分别描述其表示方法.

2.1 棋局状态

6子棋机器博弈采用的是19×19棋盘，本文利用19×19矩阵 S 与之对应，以表示棋盘状态，定义矩阵 S 为棋局状态矩阵. 在矩阵 S 中，每个数据代表棋盘上一个交点(即落子点)的棋子状态，本文将棋子状态分解为4种，分别定义黑子为0，白子为1，空为2，棋盘外为−1，如图1所示.



$$S=\begin{pmatrix}
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&0&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&0&2&1&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&1&1&0&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&1&0&0&1&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&1&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2&2
\end{pmatrix}$$

图1 某棋局状态及其对应矩阵

Fig 1 State of chessboard and chess-matrix

2.2 棋子状态

在6子棋机器博弈进程中，还必须关注每步落棋时落子位周围一个范围内棋盘上棋子的分布状态，用来后续判断棋形. 通过研究，本文以长度均为13的horizonalCodeStatus、verticalCodeStatus、leftSlopeCodeStatus和rightSlopeCodeStatus 4个一维数组作为棋盘的棋子状态分布数组，利用它们分别表示每步落棋时当前棋盘落子位置4个方向的棋子分布状况. 为利于计算机实现，在每个数组中，作出如下定义.

定义1 数组以当前的落子位为状态数组的第7位，如图2所示.

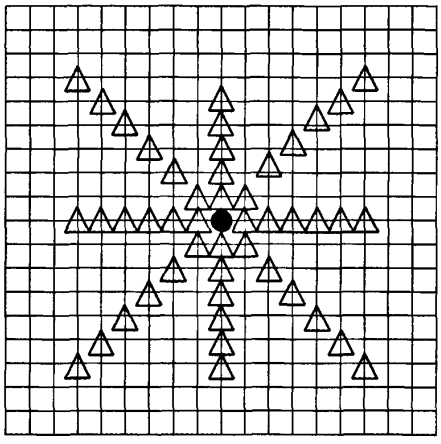


图2 棋子状态数组获取方法

Fig 2 Method of getting chess state array

定义2 数组任何一位的取值来源于棋子状态值所构成的集合{−1,0,1,2}，在机器博弈中，取其一，且仅取其一.

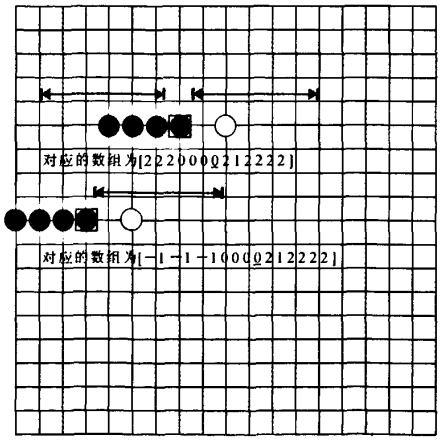


图3 棋子状态数组表示

Fig 3 Chess state array

定义3 数组必须分别表示一条直线中每个点在棋盘外、黑棋、白棋、无子的情况. 若某一方向若干位后超出边界，则记超出的点为−1. 例如在

图 3 中，图中方框位置为落子点，对应的数组为[2 2 2 0 0 0 0 2 1 0 0 0 0]和[-1 -1 -1 0 0 0 0 2 1 0 0 0 0].

2.3 棋形

2.3.1 *棋形的定义* 通过棋局状态矩阵和不同的棋子状态分布数组，可以比较清楚地描述某一时刻机器博弈进程中棋局状态和棋子状态的变化，但是，如何有意识地控制这个变化，使之朝利于己方赢棋的方向发展，还必须解析这些状态含有什么信息. 通过研究发现，棋形可以充分描述出状态中含有的丰富信息，再借助高效的智能搜索算法和评估函数(评估函数帮助计算机搜索对当前棋型局面作出适当的判断，计算出下一步着法)就能够部分解决上述问题. 通过分析，本文采用如下 15 种主要棋形：连六(获胜)、长连(获胜)、活五、眠五、死五、活四、眠四、死四、活三、眠三、朦胧三、死三、活二、眠二、死二.

以下给出其中部分代表性特征的棋形状态的定义，其他定义都可依此类推得到.

连六：在棋盘的纵向、横向或斜向的任意一条线上，6 颗同色棋子不间隔地相连.

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连六或长连，即不让对方获胜).

活四：在同一直线上的 4 颗同色棋子，符合“对方必须用两手棋才能挡住”的条件.

眠四：在同一直线上的 4 颗同色棋子，符合“对方用一手棋就能挡住”的条件.

死四：在同一直线上的 4 颗同色棋子，它们已无法形成连六或长连.

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.

2.3.2 *棋形的特征码* 必须对上述棋形进行数字化状态描述，然后才能利用棋子状态分布数组获得所需棋形信息，本文通过引入棋形特征码表示方式来实现这一目的. 以图 5、图 6 中活四为例说明棋形特征码的具体实现方法. 在落子点左右各 6 位范围内来考虑棋子分布，3 个方框内都是活四，它们的区别仅仅是非主要位置白子位置不同. 在活四棋形中，一行(直线)中有 4 个同色的子紧密连续地连在一起，两端必须同时各有 2 个子位为空(图中方框所示)，除此 6 位，其他均为无关位(图中三角所示位和箭头所示左右所有方位的子位)，子任意，即空、白、黑及其组合均可. 为此活四的特征(以黑棋为例)用数字表达即为{2,2,0,0,0,0,2,2}，只需要一个 8 位的数组或者字符串就能描述“活四”的特点. 在判断活四时，只要搜索 13 位棋子状态数组并存在连续[2 2 0 0 0 0 2 2]数据串，则判断存在一种活四棋形.

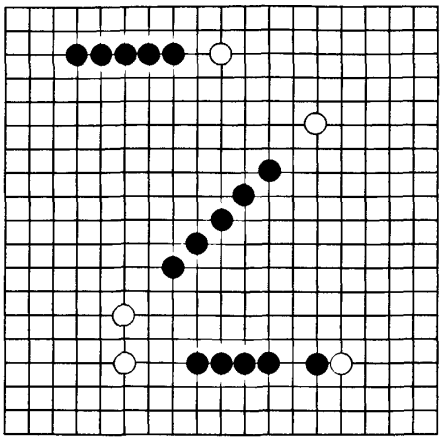


图 4　活五

Fig 4　Active five

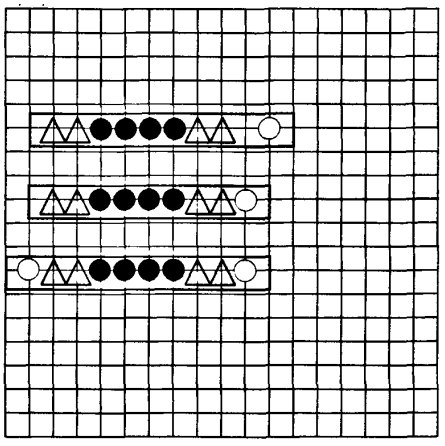


图 5　活四

Fig 5　Active four

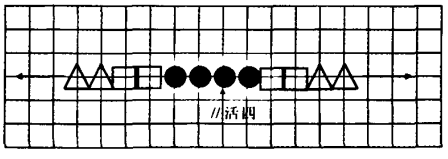


图 6　活四特征

Fig 6　Character of active four

以下为棋形特征表的一个片段：

表1 棋形特征码表

Tab 1 Code table of chess character

棋形	连六	活五(1)	活五(2)	…	眠五(1)
特征	000000	2000002	22000020	…	1000002
棋形	眠五(2)	…	活四	眠四(1)	…
特征	1000020	…	22000022	1000022	…

以下再以“眠五”其中的一个棋形为例(活四只有一个棋形，眠五对应很多棋形)给出该棋形的特征码：{1,0,0,0,0,2,0}，如图7示.

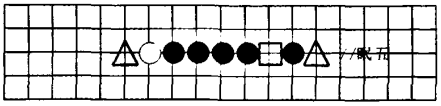


图7 眠五某一棋形的特征

Fig 7 Character of certain sleep five

必须注意的是：搜索必须按照从高级棋形往低级棋形的顺序搜. 即：先连六，然后活五，然后眠五，然后死五，然后活四…，直到最后都没有搜到就判断为没有匹配.

3 搜索方法的优化

对6子棋博弈树的搜索，本文采用基于α-β剪枝的策略搜索，深度为3层(实际6层，每一轮走两步).

3.1 α-β剪枝搜索

α-β剪枝搜索在极大极小算法基础上去掉了一些不影响最终结果的分支而返回与MIN-MAX相同走步的过程. 为了表述方便，我们不妨将走棋方定为MAX方，因为它选择着法时总是对其子节点的评估值取极大值，即选择对自己最为有利的着法；而将应对方定为MIN方，因为它走棋时需要对其子节点的评估值取极小值，即选择对走棋方最为不利的、最有钳制作用的着法[1].

α剪枝：图8中，从第一个左路分枝的叶节点倒推得到根MAX节点的评估值为5，根节点暂时取该值5，并记α为5. 此α值作为MAX方着法指标的下界. 再搜索此根MAX节点的其它子节点，例如B分支，B分支的MIN层节点必定$\leqslant$3，而MAX层节点取其子节点中最大的，而3$<$5，所以B分支在“3”节点后不用再继续扩展，可以直接剪掉此枝(整个B分支)，从而节省了大量的搜索时间，其他分支与此类似. 此类剪枝称为α剪枝. 当然α值并不是一直不变的，若某一分枝节点全部比该α值大，则α要重新界定.

β剪枝：同理，如图9所示，只是根换成MIN节点并取最小.

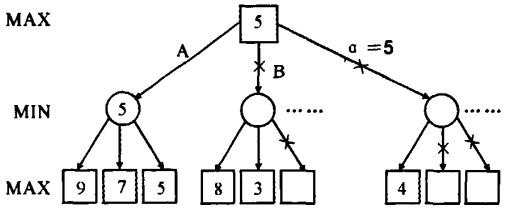


图8 α剪枝示意图

Fig 8 α cut-off

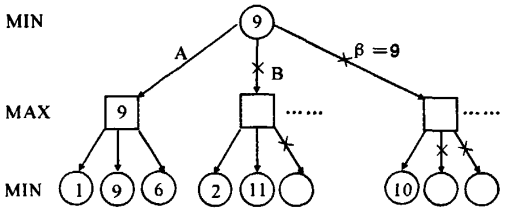


图9 β剪枝示意图

Fig 9 β cut-off

在实际多层搜索中，两者相互配合就实现了α-β剪枝搜索，这是一个典型的递归过程. 由于6子棋一次两步，通过实验和研究，本文在实现时，在普通递归过程中，还要多内嵌一个循环，再递归，实现两步递一层.

本文6子棋α-β剪枝两步搜索伪代码如下：

```
Int alphabeta(chessboard,n,alpha,beta)
  {……      //内部两层递归循环
    If(MIN NODE)        //如果是极小节点,beta剪枝
      {For(m1 = possiblemove(chessboard))
//第一步所有可能走法
        {make move m1;
        For(m2 = possiblemove(chessboard))
          //第二步所有可能走法
        {Make move m2;
        Stepvalue = alphabeta(chessboard,n-
        1,alpha,beta);  //递归搜索子节点
        unmake move m2;}  //还原
        unmake move m1;
        ……}
      Return beta;}
  Else{……}          //极大节点,α剪枝类似beta剪枝,略.
  }
```

3.2 6 子棋的搜索启发方法：策略和单步延伸

6 子棋博弈搜索树十分大，因此，6 子棋程序设计的一个关键就是设计各种好的启发式搜索方法使树的规模减小. 本文采用的启发式方法主要是结合单步延伸和威胁判断的思想.

威胁的定义[2]：假设一个玩家(称为白，W)不能连六. 如果 W 需要下 t 颗子来避免 B 连成六子，则称 B(黑)对 W(白方)有 t 个威胁.

对于 6 子棋，对敌方造成 3 个或 3 个以上的威胁就赢. 因此，赢的策略就是在阻挡所有对方威胁的同时，产生 3 个或 3 个以上的威胁. 通常一个三威胁由一个双威胁和一个单威胁组成，而判断双威胁可以转化为用棋形特征码判断有无活四或活五，相应地，单威胁也可以找到很多对应的棋形.

威胁判断与单步延伸：把威胁判断加入 α-β 递归函数的递归部分前面，从而在博弈树递归扩展之前，先做威胁判断：包括敌我双方，先判断己方目前有无形成双威胁，若有，直接连六获胜(对于常规扩展时连六状态在后面叶子节点才出现的博弈树，此举可以减少搜索很多分支)，若没有，则继续判断对方有无三威胁，若有则认输，没必要继续搜索；若没有三威胁，则继续判断对手有没有双威胁，若有则只需要以该双威胁为根节点重新扩展能堵住该双威胁的招法，不用继续扩展之前一层后续所有招法，因为面对双威胁己方肯定要防守，否则就要输，所以其他分支搜索无意义，这就可以减少搜索很多无用分支. 最后是判断对方的单威胁，若有，先阻止，然后再启动扩展，分析第二步的走法. 以上处理方法是对应情况的固定对策，所以不用搜索，直接在扩展前就单独提出“固化”处理方法. 若是上面的威胁暂时在该层都没有搜到，就要递归来进行比较复杂的潜在威胁招法判断，比如活三、死三、活二，这些棋形每次下 2 颗子就有机会形成真正的迫着. 对于这些状况本文没有单独提出模块，而是在实现过程中通过评估函数结合适当的评估值来判断. 策略流程如图 10 所示.

此方法的另一个优点是随着后续的测试和总结，各种新发现的威胁分支可以随时加入，比如朦胧三的某种状态、特殊的活二状态等等，甚至可以加入一些欲擒故纵的策略. 不需要对原流程主循环做出改变，只要在递归开始前加入相应的分支判断即可，即为后期优化提供模块化扩展.

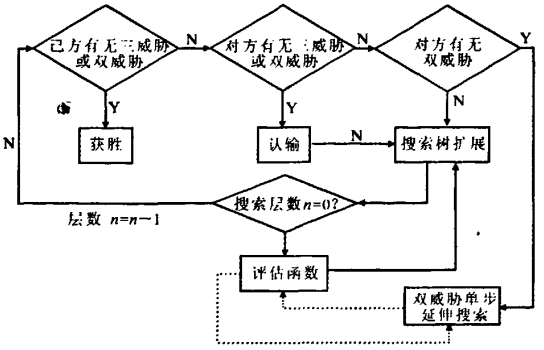


图 10　加入威胁分支判断的策略流程图

Fig 10　The flow chart of researching based on threats and singular extension

3.3 对棋盘搜索的优化：脱离战场策略

在中国象棋中，兵种的走法有规则，动址(要走的子的位置)确定落址也就确定了(比如，马走日，起点确定，落点也就确定). 对于 6 子棋，棋盘内的所有空位都是合理的落子点. 而棋盘是 19×19的，每一步的可行招法为 19×19 步（只是大约估算，不考虑已经被占位的位置)，而两步的话，就是大约 19^4 步. 己方深度为 3 层的话，就是大约 19^{12}步，这个搜索量对于计算机博弈穷尽搜索是很大的，实际证明很费时间.

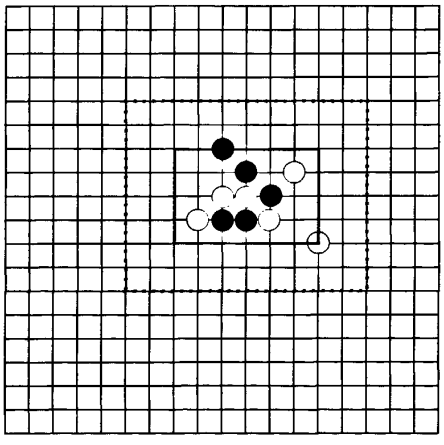


图 11　搜索范围限制

Fig 11　The area of searching

本文通过分析研究棋类的脱离战场策略，发现 6 子棋所有有用子都分布比较集中，于是提出一种方法：通过人为限定可行招法搜索范围，在几乎不损失准确度的情况下大幅度地减少搜索量，提升搜索时间. 方法如图 11 所示，在图中棋子集中区域，以上、下、左、右 4 个方向最边界棋子的坐标作为

边界勾画出一个矩形区域(如图11中黑色实线框就是当前棋盘内棋子的分布矩形区域). 然后以这个矩形为基础，每个边按坐标都扩大2个棋子位，“画出”一个新的虚线所示的矩形区域. 以这个虚线区域代替整个棋盘作为可行落子点的搜索范围，避免了全盘搜索. 实际效果证明，该方法对准确度影响很小，但是搜索时间却有显著提高. 表2为限制范围前后的时间对比，可以看出，在相同搜索算法下，限制了搜索范围后搜三层的时间也比全盘搜索搜一层的时间提高很多.

表2 限制前后搜索时间对比

Tab 2 The comparation of time used

	全盘搜索	限制搜索
搜索深度	1	3
走两步所用时间	平均20秒	平均2秒

4 评估函数的优化

评估函数一般来说必须包括5个方面的要素：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. 但是6子棋没有兵种之分，所以其评估函数只跟当前棋子的落子位置以及其周边棋子的位置状态(棋形)有关[5].

4.1 棋形评估值的设定

要得到最终的局面值，还需要各个棋形的评估值. 对于评估值的确定，本文采用了人工事先根据经验预先给出，然后把各棋形评估值保存为一个表. 该表即对应于后续遗传算法中的一个个体，后期进行优化得到最终值. 表3为部分棋形和其评估值的一个片段.

表3 部分棋形及其评估值片段

Tab 3 Segment of evaluating table

	连六或者长连	活五	眠五	死五
评估值	100 000	50 000	40 000	0

4.2 遗传算法优化棋形评估值

本课题组前期提出了一种用遗传算法[6]结合锦标赛方法来优化棋形评估值的方法(详细方法和操作步骤见文献[5]). 简述如下：由于遗传算法处理的对象主要是个体[7,8]，而个体包括一组染色体串，为此我们将前文描述的保存棋形评估值的表对应为一个个体，个体中每一个染色体分别对应于该表中每一个棋形的评估值，将染色体串解码到评估函数中即可求得评估值. 而遗传算法需要的适应度函数采用了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.

这个方法理论上可行，可在实际操作中存在很严重的问题，就是时间消耗和状态遗漏极不稳定，导致这个方法前期只是停留在理论阶段，没法实践操作. 我们知道，遗传算法加锦标赛算法时间消耗大，速度慢. 而本课题组前期在棋形表示的方法上的不足使得棋形多达几千种，而且还持续增加，数据库不稳定. 另外尽管理论上遗传算法可以优化很多参数，但几千个参数(染色体)人为的设定值已经很费时间，要用遗传算法和锦标赛方法再去优化，从时间上来讲几乎无法预算，也几乎无法完成，而且由于数据库的不稳定遗传效果也无法保证.

本文提出一种新的棋形描述方法，避免了几千个参数的优化，且数据表示稳定，只需要优化12个参数：从棋形评估值表中分别按棋形挑出一个棋形特征码代表该棋形即可，由于只有12种棋形(所有死棋形算一种)，所以个体只需要12个染色体. 这样本课题组前期的方法就具有实际操作性，实际操作效果也比较明显.

5 结论

在6子棋计算机博弈系统实现中，本文结合了6子棋各种启发信息：在搜索引擎模块，通过分析棋子分布特点，人为地缩小搜索范围，减少了搜索树规模，使得搜索时间有了显著提升. 在搜索算法中，把特定威胁下被动地必须执行的步骤从搜索扩展中单独提取出来，有效地减少了搜索量，提高了效率. 而且对于类似双威胁分支采用单支延伸搜索策略，可以实现对最有用的招法实行进一步的更深的搜索，而排除无意义招法的时间消耗. 当程序拥有优化后的搜索算法后其搜索效率有了很大提升. 在评估函数中，由于棋形特征码的引入，使得判断的准确度和速度都有显著提升，并使得前期提出的遗传算法结合锦标赛方法优化参数组合的方法具有了实际操作性，棋力明显得到了提高.

参考文献：

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

[2] WU Chen，HUANG Dei-yen，CHANG Hsiu-Chen. CONNECT6[J]. *ICGA Journal*，2005(12)：234-241.

(下转第52页)

表1 双向斜行列校验码与方阵校验码的性能比较

Table 1 The performance comparison between the double catercorner line redundancy code and the linear redundancy code

	码长	信息位	码距	码率	误码率
方阵校验码	$n\times n+m+n$	$m\times n$	$\geqslant 3$	$\frac{m\times n}{m\times n+(m+n)}$	$>\frac{m\times n!}{4!\ (m-4)!}p^2$
双向斜行列校验码	$m\times n+2(m+n)$	$m\times n$	$\geqslant 5$	$\frac{m\times n}{m\times n+2(m+n)}$	$\leqslant\frac{m\times n!}{4!\ (m-4)!}p^2$

表1中码元矩阵为$m\times n$，二元信息传错的概率为p，且远远小于1. 与方阵校验码相比，双向斜行列校验码的监督码元有所增加，码率降低，整个矩阵的码长增加，降低了误码率，这也正说明香农编码定理[2]325的正确性.

2 结论

从上面的实例中可以看出：双向斜行列校验码利用了交错的思想，对斜行列码组进行奇偶校验，避开了用奇偶校验横行、竖列的缺点，成功解决了方阵校验码存在的问题. 同时，这种编码方式不仅可以发现奇数个错误，也克服了方阵校验码在同一行、列中不能发现偶数个差错的缺点，即兼具奇偶校验码和方阵校验码的特点，但是长度有限. 因此，双向斜行列校验码是一种检错能力较强、纠错能力和方阵校验码相当的检错码.

参考文献：

[1] 潘新民. 计算机通信技术[M]. 北京：电子工业出版社，2002.

[2] 曹志刚，钱亚生. 现代通信原理[M]. 北京：清华大学出版社，1992.

[3] 夏承遗，王 娟，唐树刚，等. 应用于计算机通信中的差错检测与控制技术[J]. 天津理工大学学报，2006，22(2)：53-55.

[4] 林 楷. 一种新型二维纠错码及其软判决译码方法的研究[D]. 成都：西南交通大学，2003：3.

[5] 李建设. 数据校验的实现方法[J]. 株洲工学院学报，2003，17(5)：42-44.

[6] 胡民山，罗来成. 数字通信中差错控制的性能估算[J]. 江西通信科技，1999，12(4)：21-22.

(责任编辑 孙晓玲)

(上接第30页)

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

[4] 蔡自兴，徐 光. 人工智能及其应用[M]. 北京：清华大学出版社，2003.

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

[6] HOLLAND J H. *Adaptation in Nature and Artificial System*[M]. Ann Arbor：The University of Michigan Press，1975.

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

[8] 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.

(责任编辑 马宇鸿)