文章编号:1007-1423(2020)22-0009-06 DOI:10.3969/j.issn.1007-1423.2020.22.002

实现不围棋博弈程序的一种策略及关键算法

陈雪健,张利群,曹杨

(辽宁石油化工大学计算机与通信工程学院,抚顺 113001)

摘要:

针对不围棋,设计一种高效、可行、易于理解和使用的着法可行性判定算法。分析不围棋的三要素,设计对应的物理结构,介绍一种实战采用的博弈策略,并对同色邻接棋子位置入栈算法、气的计算算法及着法可行性判定算法进行详细分析,最终实现不围棋计算机博弈程序中这些关键算法。实验结果表明算法满足棋规要求,安全可靠,且运算速度快。这种算法的设计与实现对其他棋类博弈程序的设计与实现具有一定的参考价值。

关键词:

不围棋;博弈;栈;策略;算法排序

基金项目:

国家级大学生创新创业训练项目(No.201910148026)

0 引言

近年来,中国大学生计算机博弈大赛暨中国计算机博弈锦标赛开展得越来越好,主要表现在两个方面,一个是参加大赛的高校代表队越来越多,另一个是竞赛棋种越来越多,截止到2019年,比赛棋种达到19种。

在中国大学生计算机博弈大赛暨中国计算机博弈锦标赛中,不围棋是在2012年引入的竞赛棋种,当年共有8个代表队参加不围棋的比赛[1-4]。2019年的中国大学生计算机博弈大赛暨中国计算机博弈锦标赛中共有12个代表队参加不围棋的比赛,可见大学生对不围棋的参与度不高,原因是多方面的,其一是对不围棋的研究文献极少,可借鉴的内容不多,其次与不围棋本身具有的特点有重要关系,那就是行子策略复杂、棋局的复杂度高以及行子可行性判定算法实现困难[5-10],这些特点限制了许多大学生组队参赛。本文介绍了不围棋博弈程序实现过程中的一种策略以及着法可行性的判定算法。

不围棋(No Go)也是国际计算机奥林匹克竞赛棋种,其棋盘、棋子和棋规定义如下。

(1)棋盘

不围棋棋盘同九路围棋棋盘,即9×9围棋的棋盘,如图1所示。

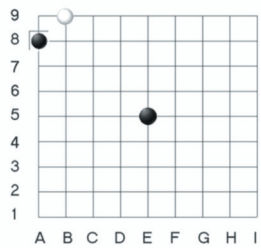

图1 不围棋棋盘

(2)棋子

黑、白两色围棋棋子。

(3)棋规

①黑子先手,双方轮流落子,落子后棋子不可移动;

②吃子:一个棋子在棋盘上,与它直线紧邻的空点是这个棋子的"气"。棋子直线紧邻的点上,如果有同色棋子存在,则它们便相互连接成一个不可分割的整体,它们的气也应一并计算。棋子直线紧邻的点上,如

果有异色棋子存在,这口气就不复存在。如所有的气均为对方所占据,便呈无气状态,无气状态的棋子不能在棋盘上存在,应当提出盘外。当一方落子,使得对方的棋子为无气状态,就是吃子。

③对弈的目标不是吃掉对方的棋子,不是占领地盘,恰恰相反,如果一方落子后吃掉了对方的棋子,则落子一方判负;

④对弈禁止自杀,落子自杀一方判负;

⑤对弈禁止空手(pass),空手一方判负;

⑥每方用时 15 分钟,超时判负;

⑦对弈结果只有胜负,没有和棋。

1 一种行棋的策略

从不围棋的棋规中可以看出,对弈的目标不是吃掉对方的棋子,不是占领地盘,取胜的关键是谁最后有位置可行棋,谁最后取得胜利。

虽然占领地盘大小不决定胜负,但是从实战经验来看,往往占用地盘多且眼多的一方在收官阶段行棋的位置就多一些,获胜的概率就大一些,因此在自己成眼的同时,破坏对方成眼,逼迫对方眼少,最终使对方无处可走,直至输棋。为阐述方便,所有图示当中,以黑子代表我方,白子代表对手方。在实现不围棋计算机博弈程序时,策略有很多,以下是自己采用的一种策略。

1.1 先占角、再占边,最后占中间

在不围棋的行子过程中,构成眼最少用两子,就是角,其次是边,最后是中间,如图 2 所示。

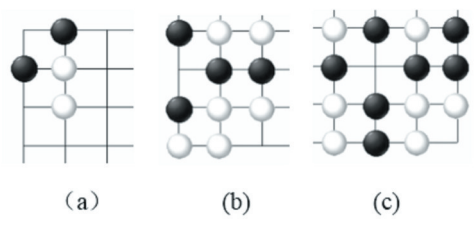

图2 三种类型的眼

事实上,由于不围棋棋规要求,即使构成上述眼的情况,确实限定了对方在眼处行棋,但是自己将来也不一定能在自己构成的眼处行棋,还要看周边的情况,也就是气的情况,这也是行棋过程中要注意的地方。

如图 3 所示,黑方眼的位置,黑方自己也不可以行棋,因为行棋就是自杀棋。

最好将两个以上的眼连在一起,也就是围棋的做活,迫使对方在自己的眼处无法行棋,而自己在收官阶段,由于有两个眼,至少还有两口气在,就能在自己的眼处行棋,如图 4 所示。

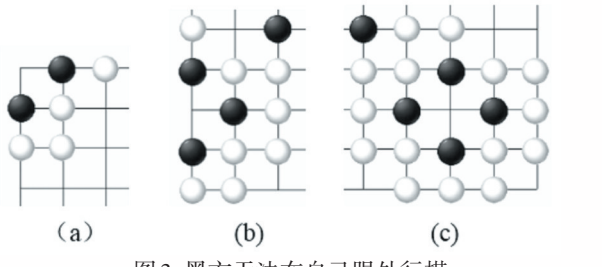

图3 黑方无法在自己眼处行棋

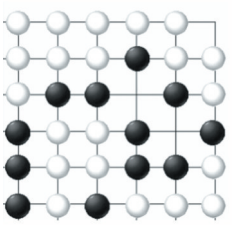

图4 在黑方眼处可以行一个黑子

1.2 攻击对手的策略

在对弈过程中,要极力破坏对手眼的形成,最好在行棋过程中既能自己形成眼,又能破坏对方眼的形成,这是最佳策略。在破坏对方成眼时,重点考虑如下几种情况,形成提前预判。由于各种情况的种类较多,这里仅举几个代表性的示例。

(1)行一子,欲形成四个眼,如图 5 所示。

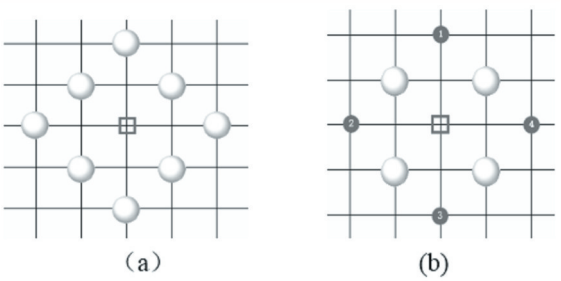

图5 行一子欲形成四个眼

(2)行一子,欲形成三个眼。

(3)行一子,欲形成两个眼,如图 6 所示。

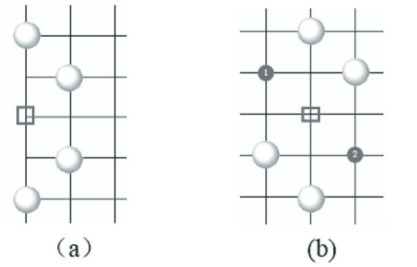

图6 行一子欲形成两个眼

(4)行一子,欲形成一个眼。

在行棋过程中,还要尽可能不让对方棋子做活,这样即使对方形成了单眼也无法在单眼处行棋。

2 不围棋的存储结构

针对不围棋棋盘行和列的特点,其棋盘的物理存储结构可以采用二维数组来表示,用 C 语言描述如下:

```
int b[9][9];
```

不围棋就有黑白两种棋子,可以将黑子和白子分别用 1 和 2 来表示。

假设目前棋局如图 7 所示,在计算机存储时,用二维数组表示如下:

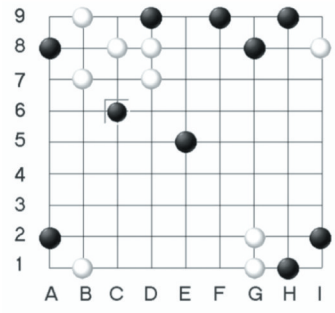

图7 棋局状态图

```
b[0][0]~b[0][8]: 0,2,0,1,0,1,0,1,0
b[1][0]~b[1][8]: 1,0,2,2,0,0,1,0,2
b[2][0]~b[2][8]: 0,2,0,2,0,0,0,0,0
b[3][0]~b[3][8]: 0,0,1,0,0,0,0,0,0
b[4][0]~b[4][8]: 0,0,0,0,1,0,0,0,0
b[5][0]~b[5][8]: 0,0,0,0,0,0,0,0,0
b[6][0]~b[6][8]: 0,0,0,0,0,0,0,0,0
b[7][0]~b[7][8]: 1,0,0,0,0,0,2,0,1
b[8][0]~b[8][8]: 0,2,0,0,0,0,2,1,0
```

也就是,棋盘上行棋位置有黑子的值是 1,有白子的值是 2,无子的位置为 0。

后面算法需要将上述二维数组的位置存储到一个栈中,这就需要将二维坐标(i,j)转化为一维坐标 k,转换方法如下:

```
k=i*9+j
```

反过来,将存储起来的一维坐标 k 转换为二维坐标(i,j)的方法如下:

```
i=k/9
j=k%9
```

3 着法可行性算法

在不围棋的“机-机”博弈过程中,对弈双方交替行棋,按照棋规,在行子过程中不能吃对方棋子,也不能自杀,因此,我方在某个位置准备行棋,就要判断该位置是否可以行棋,即要判断在这个位置行棋之后,是否我方连成一片的棋子有气,同时还要判断与我方行棋处邻接的对方棋子是否有气。

3.1 同色邻接棋子位置入栈递归函数

为了能够判断在 b[i][j]处我方落黑子之后,是否吃掉了对方棋子,就要看这个黑子邻接的白子是否有气,没有气就是吃子了,如图 8 所示,假设 b[i][j]为 b[5][3]。

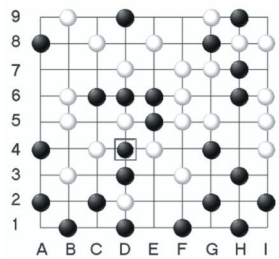

图8 黑子行棋位置

如何判断邻接的白子是否有气?这就需要把与黑子邻接的上下左右四个方向的白子块,存储起来,再计算其气数,只要有一个白子块的气数为零,则说明吃子发生了,这样我方在这个 b[i][j]处就不能行棋。判断黑棋是否是自杀,道理与这个相同,不再赘述。用于存储白子块、黑子块的线性表如下,采用的是栈的存储方式[11-14]。

```
int wstack[81], wtop=-1;
int bstack[81], btop=-1;
```

采用深度优先算法将所有邻接的白子所在位置全部存储在一维数组 wstack 中,且存储过程中,每一位置

都不重复存储。函数 int Isunique(int x)用于判定位置 x 是否在栈中出现,若出现函数值为 0,否则函数值为 1,函数 Isunique (int x)实现简单,在此略去。

邻接白棋子的位置入栈递归函数如下:

```
void Adwpush(int i,int j)
{
 if(j+1<9)
  if ((i,j+1)==2 && Isunique(i*9+j+1))
  {
   wtop++;
   wstack[wtop]=i*9+j+1;
   Adwpush(i,j+1);
  }
 if(j-1>=0)
  if ((i,j-1)==2 && Isunique(i*9+j-1))
  {
   wtop++;
   wstack[wtop]=i*9+j-1;
   Adwpush(i,j-1);
  }
 if(i+1<9)
  if ((i+1,j)==2 && Isunique((i+1)*9+j))
  {
   wtop++;
   wstack[wtop]=(i+1)*9+j;
   Adwpush(i+1,j);
  }
 if(i-1>=0)
  if ((i-1,j)==2 && Isunique((i-1)*9+j))
  {
   wtop++;
   wstack[wtop]=(i-1)*9+j;
   Adwpush(i-1,j);
  }
}
```

与(i,j)处邻接的黑子入栈函数,与函数 Adwpush()的类型及参数一致,可参照函数 Adwpush()实现,不妨叫做 Adbpush()。

3.2 气的计算算法

把邻接的白子块存储在了栈 wstack[]中,该白子块是否有气,只需检查白子块中每个棋子的上下左右是否有空位置,只要有一个白色棋子邻接一个空位置,这个白子块就有气。

计算白子块气的算法如图 9 所示,称之为 Haswqi( )算法,返回值是 1,就代表有气,算法返回值是 0,就代表无气。

对于邻接 b[i][j]的黑子,只有一个黑子块,其气的计算算法与此类似,不再赘述,称之为 Hasbqi( )算法。

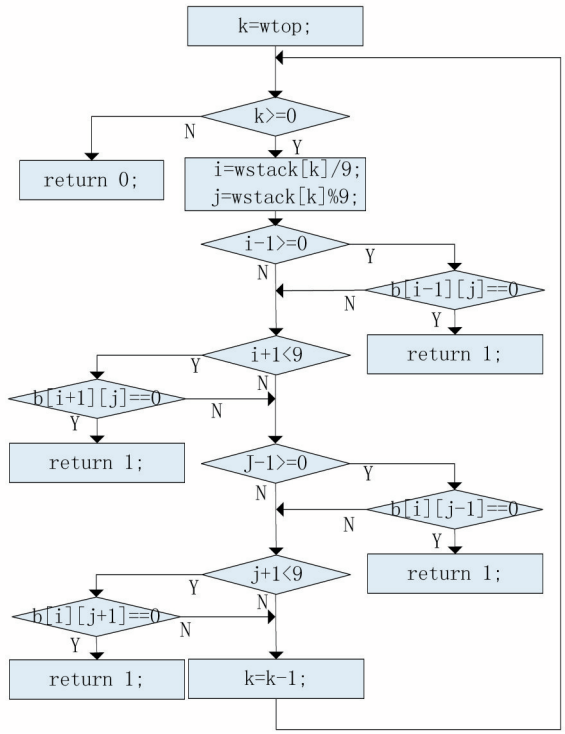

图9 Haswqi( )算法

3.3 着法可行性判定算法

有了上述算法,就可以给出我方在 b[i][j]处着法可行性判定算法,如图 10 所示。若在 b[i][j]处可行棋,算法返回值为 1,不可行棋,算法返回值为 0。

当我方要在 b[i][j]处行子,根据 i、j 的值,最多需要判断四个白子块的气,这四个白子块是分别从 b[i-1][j]、b[i+1][j]、b[i][j-1]及 b[i][j+1]处进行深度优先遍历得到的。其中 Initwstack()是白子块栈初始化函数,Initbstack()是黑子块栈初始化函数。

4 结语

由于不围棋的复杂性,无法生成及存储完整的博弈树,因此采用什么样的方法来准确评估当前局面的

好坏一直是不围棋博弈程序设计者研究的一个问题，而着法可行性判定算法又是局面评估的基础。

在中国大学生计算机博弈大赛暨中国计算机博弈锦标赛中，使用上述策略和算法实现的不围棋博弈程序，得到了检验。实践证明，博弈程序运行正确，速度快，安全可靠，不仅顺利完成全部比赛，而且达到了预期效果。

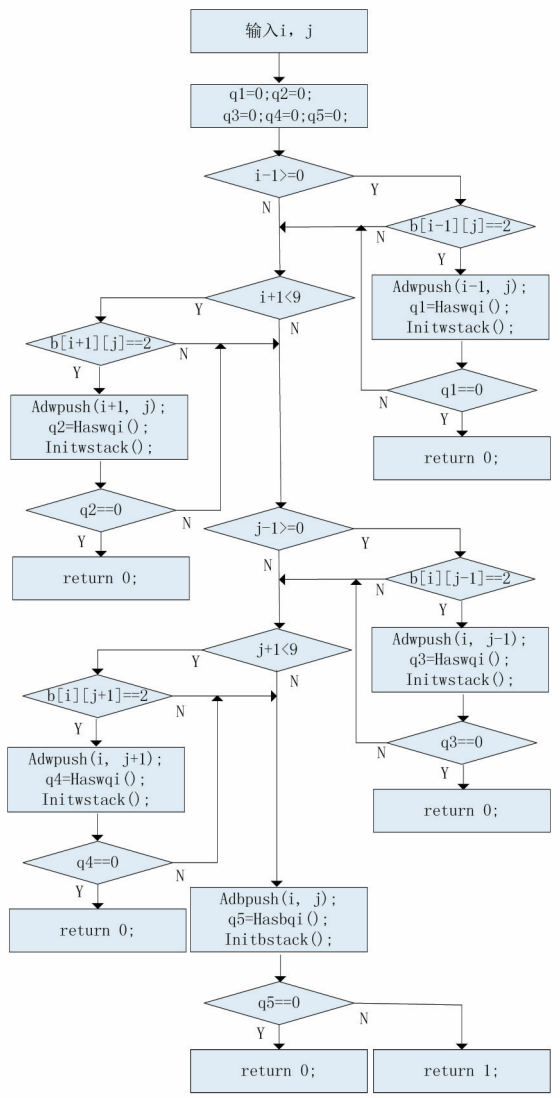

图10 着法可行性判定算法

参考文献：

[1]GAO Qiang, XU Xin-he. The NSCGT-CCGC Computer Games Tournament[J]. International Computer Games Association Journal, 2013, 36(4):252-254.

[2]TONG Guo-feng, XU Xin-he. Progress of Computer Games in China[J]. International Computer Games Association Journal, 2011, 34(3):168-170.

[3]王亚杰，邱虹坤，吴燕燕，等. 计算机博弈的研究与发展[J]. 智能系统学报，2016，11(6):788-798.

[4]王骄，徐心和. 计算机博弈:人工智能的前沿领域——全国大学生计算机博弈大赛[J]. 计算机教育，2012(7):14-18.

[5]郭琴琴，李淑琴，包华. 亚马逊棋机器博弈系统中评估函数的研究[J]. 计算机工程与应用，2012，48(34):50-54.

[6]张小川，陈光年，张世强，孙可均，李祖枢. 六子棋博弈的评估函数[J]. 重庆理工大学学报:自然科学，2010(2):64-68.

[7]张明亮，李凡长. 一种新的博弈树搜索方法[J]. 山东大学学报:工学版，2009，39(6):1-7.

[8]吕艳辉，宫瑞敏. 计算机博弈中估值算法与博弈训练的研究[J]. 计算机工程，2012，38(11):163-166.

[9]张小川，唐艳，梁宁宁. 采用时间差分算法的九路围棋机器博弈系统[J]. 智能系统学报，2012，7(3):278-282.

[10]高强，徐心和. 时间复杂性和空间复杂性研究[J]. 智能系统学报，2014，9(5):529-535.

[11]唐策善，黄刘生. 数据结构[M]. 合肥:中国科学技术大学出版社，1992.

[12]严蔚敏，吴伟民. 数据结构[M]. 北京:清华大学出版社，2002.

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

[14]张利群. 苏拉卡尔塔棋网络博弈平台的吃子算法[J]. 计算机工程与应用，2016，25(7):62-66.

作者简介：

陈雪健(1999-)，男，辽宁辽阳人，本科生，研究方向为计算机应用

通信作者：张利群(1965-)，男，辽宁抚顺人，硕士，教授，研究方向为计算机软件与理论、机器博弈，E-mail: zzllqun@163.com

曹杨(1982-)，女，辽宁抚顺人，博士，讲师，研究方向为机器博弈、计算机应用

收稿时间：2020-04-23　　修稿时间：2020-06-12

（下转第22页）

Multiple Exits Evacuation Algorithm Application and Improvement

LI Xin-xin

(School of Computer Science, South China Normal University, Guangzhou 510631)

Abstract:
For large buildings with complex structures, a well-designed emergency evacuation plan is essential to reduce casualties in the emergency. In order to make the evacuation path planning algorithm more adaptive for the specific building structure, a multi-exit single route evacuation method is proposed based on the multiple exits evacuation algorithm. The method is further refined and improved on the multiple exits evacuation algorithm. First, for specific building structures, we build graph models and add relevant properties to better simulate the real situation. Then, in order to give specific evacuation routes, the graph model is processed so that there is at most one path between each area and each escape exit. Finally, the multiple exits evacuation algorithm is introduced to calculate the escape route that each area should take. The method can be applied to evacuation route planning for different complex buildings.

Keywords:
Complex Architecture; Evacuation Planning; MEEA

（上接第 13 页）

A Strategy and Key Algorithms to Implement the Game Program of No Go

CHEN Xue-jian, ZHANG Li-qun, CAO Yang

(School of Computer and Communication Engineering, Liaoning Shihua University, Fushun 113001)

Abstract:
For No Go, an efficient, feasible, easily to understand and used algorithm of the move feasibility judgment was designed. The three key elements of No Go were analyzed, and the corresponding physical structure was designed. A game strategy used in actual combat was introduced. In addition, the algorithm with pushing the same color adjacent chess piece locations into stack, the algorithm of liberties calculation and the algorithm of the move feasibility judgment were analyzed in detail, and these key algorithms in the computer game program of No Go were finally realized. The experimental results show that these algorithms satisfy the requirements of the chess rules, are safe, reliable, and fast. The design and implementation of these algorithms have certain reference value for that of other chess game programs.

Keywords:
No Go; Game; Stack; Strategy; Algorithm