点格棋中主要技术的分析

作者：张晨光　　指导老师：李淑琴

摘要

棋类博弈是人工智能的重要研究主题之一。点格棋早期在欧洲比较流行，是国际计算机奥林匹克锦标赛的比赛项目，在国际上也有不少知名院校开发了相关软件，如 UCLA（加州大学洛杉矶分校）数学系的 Tom 开发了棋力相关软件，且运算效率很高，同时还可以随时调整棋盘大小。近几年，由于国内计算机锦标赛开展，也逐渐被国内计算机博弈爱好者熟悉。从在我国开展计算机锦标赛以来，点格棋就成为赛事的主要比赛项目之一，在全国大学生计算机博弈大赛中点格棋也是参赛人数最多的项目之一。

本文将系统的介绍点格棋，包括点格棋的起源、基本内容的介绍、以及相关的搜索算法、评估函数等。通过本文章可以比较全面的了解计算机博弈中的点格棋，由浅入深的学习点格棋的具体内容。国内并没有太多的文章比较全面的介绍点格棋，我们大多数的了解局限于国外的文章。学习起来可能会有一些障碍，所以我希望可以通过本文更加全面的介绍点格棋的有关内容。让想接触点格棋和想参加大学生计算机博弈大赛的同学们可以更加方便的入门。本文仅代表自己粗略的意见，如果有错误的地方，欢迎指正。

关键词：人工智能　点格棋　UCT 算法　α－β 剪枝搜索算法 棋盘表示　评估函数

目录

1 绪论

1.1 研究背景及意义

博弈是人工智能的重要研究主题，人工智能的发展在很大程度上得益于博弈研究的发展。1997 年著名的深蓝计算机战胜国际象棋世界冠军卡斯帕罗夫成为轰动一时的新闻事件。点格棋是由法国数学家爱德华·卢卡斯在 1891 年提出的纸和笔的双人游戏。早期在欧洲比较流行，是国际计算机奥林匹克锦标赛的比赛项目，在国际上也有不少知名院校开发了相关软件，如 UCLA（加州大学洛杉矶分校）数学系的 Tom 开发了棋力相关软件，且运算效率很高，同时还可以随时调整棋盘大小。近几年，由于国内计算机锦标赛开展，也逐渐被国内计算机博弈爱好者熟悉。从在我国开展计算机锦标赛以来，点格棋就成为赛事的主要比赛项目之一，在全国大学生计算机博弈大赛中点格棋也是参赛人数最多的项目之一。

初学者在规则上并不会受到太大的阻碍然而在计算机博弈我们不仅仅局限于规则的了解。无论是什么样的博弈比赛，步法的生成和棋盘局势的评估是至关重要的。这就需要我们更加深入的、更加统一的了解认识点格棋统。统一的认识有利于我们之后深入研究和大家的交流进步，不至于在交流时各说各话，再浪费不必要的时间去统一大家程序中一些基本内容。对新手对代码的理解也是非常的有帮助的。

2 点格棋的基本介绍

2.1 点格棋棋盘的介绍

点格棋属于典型的添子类游戏，非常适合于计算机博弈，游戏棋盘为 m×n,比赛中的棋盘一般为 6×6，如下图所示。

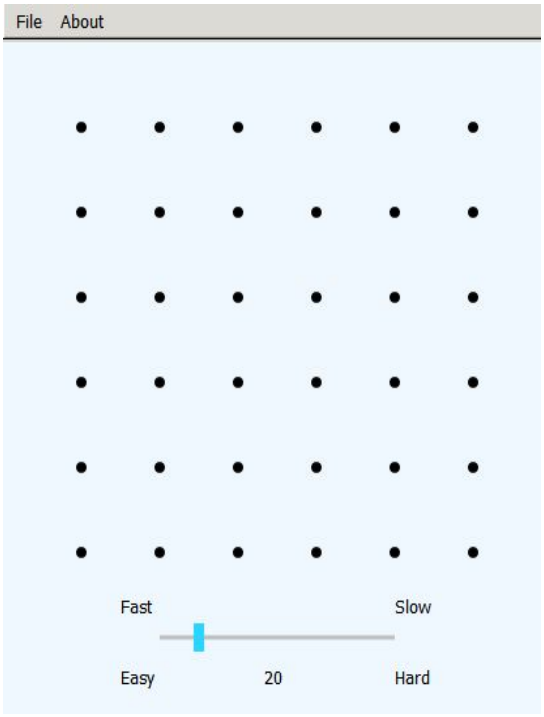

图 1-1-1

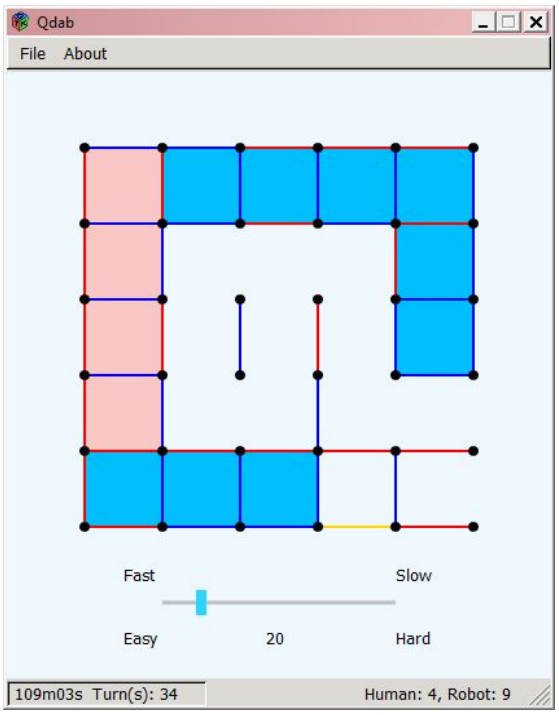

图 1-1-2

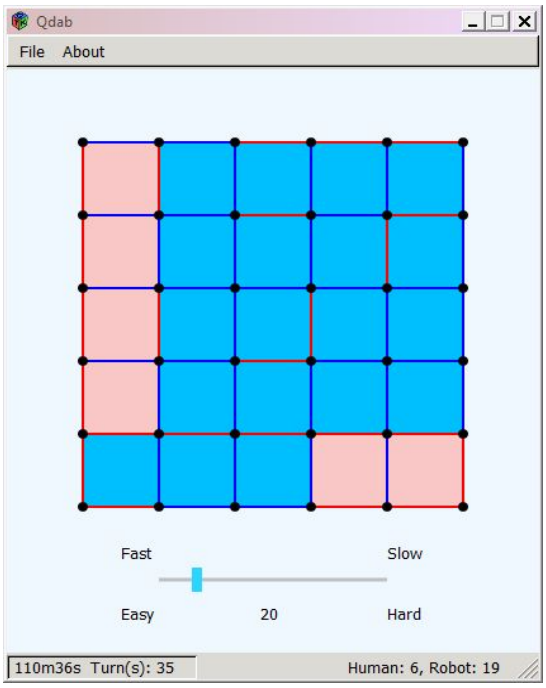

图 1-1-3

图 1-1-1 为未走之前的棋盘，图 1-1-2 为对弈过程中的棋盘。图 1-1-3 为对弈结束时的棋盘，明显蓝格子多，则蓝格子一方胜出。

2.2 点格棋的比赛规则

目前，国际计算机奥林匹克锦标赛和全国大学生计算机博弈大赛中采用 6×6 的棋盘的规则为：

（1）双方以此将临近的两点连线，不可越点不可重边，不可以连对角线。

（2）每一个格子被四个边围城，最后一条边的所有者便是这个格子的拥有者。

（3）连接一条边拥有一个格子后，可以再下一步，直到不再有格子占领。

（4）格子格子占满游戏结束，格子数最多的一方胜。

一般因为有先后手之分，先后手有不同的优势，为了公平起见每局比赛都需要进行两次，每一方都有一次先手和后手的机会。

2.3 点格棋中的基本概念

在点格棋中有一些基本概念经常会被用到在研究点格棋时基本的的概念必须熟悉，对以后文献的阅读和点格棋的研究是必不可少的，下面我们对一些基本的概念进行简单介绍：

（1）自由度——格子中未被占领的边数。

（2）C 型格——格子的自由度为 1 时该格子为 C 型格。

（3）相邻——坐标分别为（i,j）和（k,l）的两个格子，当且仅当|i-k|+|j-l|=1,并且公共边没有被占领时为相邻。

（4）一轮（turn）——比赛中一方走完，换另一方开始时为一轮。

（5）一步（move）——临近两点连成线的一步。

（6）补格——完成对格子的占领的一步。

（7）链——存在一组格子 BN=(b0,b1,..,bi,...,bn),有 bi(bi∈BN)自由度为 2（格子中未被占领的边），b0,bn 之间最多只有一个相邻属于 BS，其余的格子均有两个相邻，并且相邻的格子属于 BN 中的格子，那么 BN 称为链，明显 b0.bn 为链的端点，并且它们可以是同一个格子。

链又分为长链和短链。

①短链（short chain）——由 1 个或者两个格子组成的链。

②长链（long chain）——由三个及以上的格子组成的链。

（8）环(circle)——BN=(b0,b1,..,bi,...,bn),有 bi(bi∈BN)自由度为 2，并且 bi 均有两个相邻，这两格子属于 BN，那么，BN 就成为环，环一般就是长链首尾相接形成的，所以至少有四个格子。

（9）双交（double cross）——下一步棋可以占领两个格子的情况叫做双交。

（10）残局——棋局中只有长链和环的情况叫做残局。

（11）接合点——棋局中自由度为 3 的格子叫做接合点。

2.3.1 点格棋中长链和环的处理

在点格棋比赛中长链处理时有几条重要的的规则。

（1）先进入长链的一方会被下一轮的走棋方占领长链中的格子，所以在点格棋对弈过程中最后一轮的走棋方应该迫使先一轮的走棋方进入长链。最后一轮长链的占领往往对比赛的胜负起到了关键性的作用。

（2）重要的一条定理：Dots+Doublecrosses=Turns

其中，Dots 是指初始棋盘上其盘点的个数，Doublecrosses 是指棋局结束时双交的个数。Turns 是指棋局结束时总轮数。km.

（3）长链定理：如果棋盘上有奇数个点，则先手方应该争取形成奇数条长链，后手方则应该争取形成偶数条长链。如果棋盘上有偶数个点，先手方应该争取形成偶数条长链，后手方则相反。所以为了达到目的我们需要在对弈的过程中，要想办法制造或者破坏长链，必要的时候必须让格，不能仅仅局限于眼前的利益。所以就需要有好的搜索算法和评估函数来估计棋局的发展方向。

3 点格棋棋盘的表示(全部以 6×6 棋盘为例)

大学生计算机博弈比赛中，点格棋是比赛是一个限时的比赛项目，比赛的总体时间是有限的。所以我们必须要做到充分的利用时间，尽可能的在最少的时间内完成最多的搜索。点格棋的对弈，每一步走棋都是不确定的，我们并不知道对方的走棋策略,所以我们要尽可能的模拟多的情况,所以就需要搜索的更深更广。这样我们才能了解更多种可能发生的情况,获得一个胜率比较大而且相对比较可信的走棋策略。所以我们需要优化我们的棋盘表示方法，减少一些不必要的时间开支，把时间充分的利用起来。

3.1 用坐标表示棋盘

点格棋的棋盘我们已经介绍过如图 1-1.传统的表示一般使用坐标来表示每个点的位置，进一步表示棋盘中格子和边的位置。

3.2 用数字代号来表示棋盘

但是用坐标来表示具体位置在棋盘界面的编写时可以使用，但是在对弈的过程中用坐标来表示每个点，在搜索的时候表示会异常的繁琐，所以我们可以编号表示每一个点，并记录其具体的坐标。这样在表示边时也会相对比较方便，搜索时，表示各个特征值会相对比较简单，对效率的提升也有很大的帮助。最后在决定要走的棋之后再转化为相应的坐标，在界面上表现出来。

3.3 转化棋盘的点来表示棋盘

用数字代号来表示棋盘时虽然可以减少一些繁琐的表示工作，但是在格子的表示，尤其是格子的自由度判断时，每一次都需要查找每个点之间的链接以及它们坐标的关系、位置，搜索的过程比较繁琐，这样在长链和环的判断和构造上会带来很大的麻烦，然而长链和环的判断和构造是比赛中非常重要的过程。所以我们避无可避只能进行搜索判断，这样一来就需要浪费很多的时间。然而这并不是我们想要的结果，所以我们尝试一种新的表示方法。不直接表示每一条边或者每一个点。我们表示每一个格子，并且把每个格子用一个点来表示，利用图论的思想，把每个格子的自由度用度来表示。初始每个点的度都是 4，如图 3-3，在对弈的过程中如果有相应的边产生则剪短点之间的链接，并且把每个点的度减 1，如图 3-3-2 这种方法比较方便的记录了对弈过程中的特征量，搜索时减少了查询的时间，而且也相应减少了代码量，使程序员的编码过程和思路更加清晰、更加

简练。

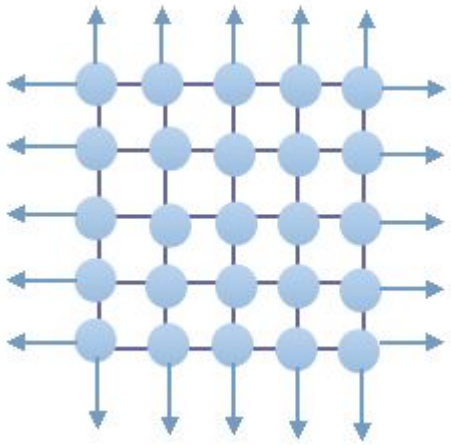

图 3-3

4 搜索算法

前面已经介绍过，目前点格棋一般采用的比赛方法是包干计时，每一方的时间都是有限的一般以 15min 作为双方时限，加时赛一般为 7.5min 每方。所以时间的限制的在点格棋设计时必须要考虑的问题，所以在点格棋搜索时时限是重要的依据，虽然搜索次数的增加会增大比赛的胜率但是必须要考虑时间问题，所以我们尽量要使用效率很高，准确率更高的算法。下面介绍几种常用的算法。

4.1 α - β 剪枝搜索算法

α - β 剪枝搜索算法是极大极小算法的发展，由于极大极小算法向下搜索的过程中分支过于庞大，而计算机并不能承受这么大的工程量，一般到差不多第五层程序就会崩溃，随意只能根据前几层节点的评估值来判断步法，然而这样的遍历的层数过少，得到的结果并不精确，于是由此演化而来 α - β 剪枝搜索算法。

在博弈树边计算评估各节点的倒推值的同时，根据评估出的倒推值的范围，及时停止扩展那些已无必要再扩展的子节点，即相当于剪去了博弈树上的一些分枝，从而可以使搜索的深度增大，提高了搜索效率的同时提高。具体的剪枝方法如下：

①对于一个 MIN 节点，若能估计出其倒推值的上确界 β，并且这个 β 值不大于 MIN 的父节点的估计倒推值的下确界 α，即 α ≥ β，则就不必再扩展该 MIN 节点的其余子节点了(因为这些节点的估值对 MIN 父节点的倒推值已经没有影响)。把这一过程称为 α 剪枝。

② 对于一个 MAX 节点，若能估计出其倒推值的下确界 α，并且这个 α 值不小于 MAX 的父节点的估计倒推值的上确界 β，即 α ≥ β，则就不必再扩展该 MAX 节点的其余子节点了(因为这些节点的估值对 MAX 父节点的倒推值已经没有影响)。把这一过程称为 β 剪枝。

α - β 剪枝算法是一种优化以后算法比极大极小算法优化了很多，但是 α - β 剪枝算法剪枝效率与博弈树分支的排列有很大的关系，理想排列顺序下的剪枝效率和最差情况下相比差别极大，所以他的准确性并不是很好。当然，以后也有人将其改进，在这里不做详细的解释。

4.2 UCT 算法

UCT 又名 UCB for Tree Search，是 UCB(Upper Confidence Bound)在 Tree

Search 上的应用。而 UCB 本来是为了解决吃角子老虎机问题(Bandit Problem)而产生的。问题简述如下：有若干台吃角子老虎机，每台机器可以投钱并拉动操纵杆，此时会得到收益(reward)，投钱、拉杆、得到收益的过程，称之为一个 Play。每台老虎机都有不同的收益率，如果玩家想要在若干次的 Play 里获得最大总收益，那么该怎么做？

一般来说，玩家会开始动手玩，并且依照目前积累的经验来决定下一次的 Play 要选择哪一台机器，这称之为开发(exploitation)。但是，如果玩家只是根据自己之前的经验在一台机器，而不去尝试其他的机器，很有可能会忽略收益率更高的机器，所以适度地尝试其他机器是必须的，这称之为探险(exploration)。UCB 试图解决的 ExE(exploitation vs. exploration)问题就是如何在开发与探险之间保持平衡。

UCB 根据目前得到的信息即经验，配合上一个调整值，力求在开发与探险间保持平衡。大致来说，每一次，UCB 会根据每一台机器目前的平均收益值(即每台机器到目前为止的所有表现)，加上一个额外的调整参数，得出本次 Play 此台机器的 UCB 值，然后根据此值，挑选出拥有最大 UCB 值的机器，作为本次 Play 所要选择的机器。其中，所谓额外参数，会随每一台机器被选择的次数增加而相对减少，其目的在于让选择机器时，不但但看重一个机器过去的表现，而可以适度地探索其他机器，这样才可能发现收益更大的机器。UCB 公式表示如下(也称为 UCB1)[14]：

$$\overline{X}_j + \sqrt{\frac{2\log n}{T_j(n)}} \qquad (1)$$

是第 j 台机器到目前为止的平均收益，是第 j 台机器被测试的次数，n 是所有机器目前被测试的总次数。下一个被选择的机器将是公式(1)的值最大的机器。前项为此台机器的过去表现，后项则是调整参数。

而 UCB1-TUNED 是相对于 UCB1 实验较佳的配置策略。UCB1-TUNED 的公式如下：

$$V_j(s)=(\frac{1}{\varepsilon}\sum_{y=1}^{\varepsilon}\overline{X_{j,y}}^2)-\overline{X_{j,\varepsilon}}^2+\sqrt{\frac{2\log n}{\varepsilon}} \quad (2)$$

$$\overline{X_j}+\sqrt{\frac{\log n}{T_j(n)}\min\{0.25, V_j(T_j(n))\}} \quad (3)$$

让公式(3)的值最大的机器将是下一个被告选择来测试的机器。

根据不同的情况调整参数可能不同，所以我们可以在参数调整值的前边再加一个参数 C，即

$$\overline{X_j}+C\sqrt{\frac{\log n}{T_j(n)}\min\{0.25, V_j(T_j(n))\}} \quad (4)$$

这样在不同情况下，参数调整时能进一步的提高准确性。

UCT(UCB for Tree Search)其实就是把 UCB1 或 UCB1-TUNED(统称为 UCB)等公式运用于 Tree Search 上的一个方法。以概念而言，UCT 把每一个节点都当作是一个吃角子老虎机问题，而此节点的每一个分支，都是一台吃角子老虎的机器。选择分支，就会获得相应的收益。

Tree Search 开始时，UCT 会建立一棵 Tree，然后：

(1)从根节点开始

(2)利用 UCB 公式计算每个子节点的 UCB 值，选择 UCB 值最高的子节点

(3)若此子节点并非叶节点(从未拜访过的节点)，则由此节点开始，重复(2)

(4)直到遇到叶节点，则计算叶节点的收益值，并依此更新根节点到此一节点路径上所有节点的收益值,。

(5)由(1)开始重复，直到时间结束，或达到某一预设次数（主要是为时间考虑）。

(6)最后由根节点的所有子节点中，选择平均收益值最高者，作为最佳节点

此一节点，就是 UCT 的结果。

UCT 算法相对于 α - β 剪枝算法来说，大大的提升了评估值的准确性。得到的结果更为可靠。

5 估值函数设计

点格棋的评估算法主要是依据长链定理，目标是对弈中构造有利于本方的长链的奇偶数，整个棋局大致可分为三个部分开局、中局和残局，开局主要实现棋盘划分、板块规划，中局主要是以长链定理为依据构造有利于自己的局势，残局主要判断怎样走才能占领更多的格子，是贪婪走法占格还是让格。

现在人工智能是博弈获得评估值的比较好的手段，在这里我们简单的介绍 B-P 神经网络在点格棋博弈中的使用，如图 5-1

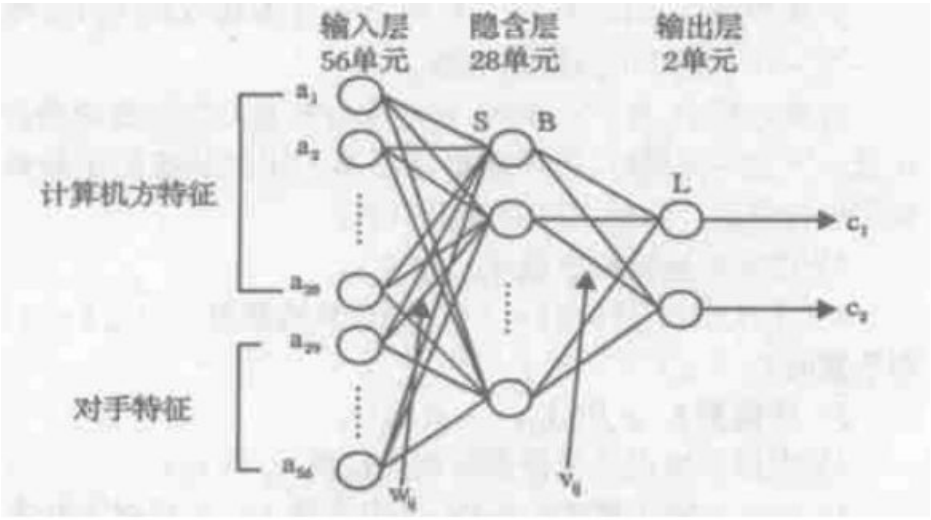

图 5-1

我们把棋盘的特征值作为输入层输入，评估值作为输出层，不断地进行训练学习得调整权值，得到比较精确的评估函数。

6 工作总结及未来展望

6.1 工作总结

本文完成的主要工作：

①介绍了计算机点格棋的相关背景和基本知识主要是点格棋的一些基本概念和处理链和环的策略。

②介绍了棋盘的表示方法，主要是转化后的方法。

③介绍了搜索算法，主要是 UCT 算法在点格棋中的使用。

④简单介绍了 B-P 精神网络在点格棋评估时的使用。

6.2 未来展望

①可以利用更加有力的硬件资源，提升棋力，充分利用多线程的优势，使搜索更加全面，准确性更高。

②对点格棋的研究可以更加的深入，与了解点格棋的人进行交流学习，期望可以学习到更多有关知识。

③在计算机博弈方面有所突破，利用想通之处，可以了解学习更多的棋种和更好的算法。

致　　谢

首先我要感谢我的指导老师李淑琴老师，在我完成论文过程中感到迷茫和无助的时候给我以鼓励和建议。李老师那严谨求实的治学态度、一丝不苟的工作作风和务实精神给我留下了极为深刻的印象，使我获得了一份宝贵的精神财富，我将受益终身。

感谢支持和帮助我的学长和队友，是他们在我迷茫的时候为我解惑，耐心的为我讲解，和我一起努力，他们是我学习的榜样。

感谢学校给我机会让我参加比赛，了解计算机博弈。更感谢为计算机博弈贡献的前辈们。

最后，向所有帮助过我的人致敬。

参 考 文 献

[1] 许峰雄, 黄军英, 蔡荣海等. “深蓝”揭秘: 追寻人工智能圣杯之旅. 上海: 上海科技教育出版社. 2005. 239-245.

[2] Albert L. Zobrist. Feature Extraction and Representation for Pattern Recognition and the Game of Go. Ph.D. Dissertation, University of Wisconsin, 1970.

[3] Zhiqing Liu, Qing Dou. Automatic Pattern Acquisition from Game Records in GO. in the Journal of China University of Posts and Telecommunications. 2007 14(1): 100-105.

[4] M. Boon. A pattern marcher for Goliath. Computer Go. 1990, (13): 13-22.

[5] B. Bouzy, T. Cazenave. Computer Go: an AI oriented survey. Artificial Intelligence. 2001, 132(1): 39–103.

[6] 危春波, 王海瑞, 文乔农. 博弈树搜索算法的分析与实现 .昆明:昆明理工大学信息工程与自动化学. 650051.

[7][美]Nils J Nilsson．人工智能[M]．北京: 机械工业出版社,2000．

[8]Knuth, D.E. and Moore, R.W. (1975). An Analysisof Alpha-Beta Pruning[J]. Artificial Intelligence,Vol. 6, No. 4:293-326.

[9]Berliner, H.J. (1979). The B*-tree search: Abest-first proof procedure[J]. Artificial Intelligence,Vol. 12, No. 1:23-40.

[10] Cohen P R, Feigenbaum E A.The handbook of artifical intelligence[M].New Jersey: Addision Wesley, 1982: 45- 80.

[11] Clancy W J.Heuristic classification[J].Artificial Intelligence,1985, 27:
289- 350.

[12] 王永庆.人工智能原理与方法[M].西安: 西安交通大学出版社, 2003:290-292.

[13] Luger G F.Artifical intelligence structures and strategies forcomplex problem solving [M].5th ed.Beijing: China Machine Press,2006: 110- 118.

[14] 王文杰, 叶世伟.人工智能原理与应用[M].北京: 人民邮电出版社,2004: 53-61.

[15] Elmo, Timoteus .Dots and Boxes. Loc Publishing,2011-07-30.

[16] BAUDIS Petr，GAILLY Jean-loup． Pachi: Software for the BoardGame of Go / Weiqi / Baduk［EB /OL]. [2012 － 07 － 04]. http:/ /pachi.or.cz /．

[17] AUER P, CESA-BIANCHI N, FISCHER P. Finite-time Analysis of the Multi armed Bandit Problem[C]//Proc.of Machine learning, Springer-Verlag, 2002, 47(2 /3) : 235 -256．