doi:10.3969/j.issn.1005-3026.2016.08.002

证据计数法在落子类机器博弈中的应用

高 强[1,2], 徐心和[1]
(1. 东北大学 信息科学与工程学院, 辽宁 沈阳 110819; 2. 沈阳大学 辽宁省装备制造综合自动化重点实验室, 辽宁 沈阳 110044)

摘 要: 详细阐述了基于"与或树"的证据计数法原理,综述了证据计数法在一些落子类博弈系统中的应用,论述了证据计数法和 PN^2 算法的缺陷. 基于 PN^2 算法,提出了一种两级的 PN 算法,即 PN-DFPN,其中第一级采用标准的 PN 算法,第二级采用一种深度优先的 PN 算法代替 PN^2 算法中的第二级 PN 算法,弥补了 PN^2 算法存在的不足. 将 PN^2 和 PN-DFPN 算法应用于求解 7×7 和 9×9 棋盘的六子棋开局局面上,实验证明,PN-DFPN 在搜索效率和求解能力上都明显优于 PN^2.

关 键 词: 计算机博弈;证据计数法;两级 PN 算法;与或树;博弈问题理论解;六子棋
中图分类号: TP 301.5 **文献标志码**: A **文章编号**: 1005-3026(2016)08-1070-06

Application of Proof-Number Search to Computer Lazi Games

GAO Qiang[1,2], *XU Xin-he*[1]
(1. School of Information Science & Engineering, Northeastern University, Shenyang 110819, China; 2. Key Laboratory of Manufacturing Industrial Integrated Automation, Shenyang University, Shenyang 110044, China. Corresponding author: GAO Qiang, E-mail: tommy_06@163.com)

Abstract: The principles of proof-number based on the AND/OR tree are expounded, the application of proof-number search to computer Lazi games is summarized and the flaws of proof-number and PN^2 search are discussed. Moreover, a new two-level algorithm, i.e., PN-DFPN search, is introduced, which performs at the first-level a standard proof-number search and at the second-level a depth-first PN search that replaces the second-level of PN^2. This algorithm covers the shortage of PN^2 search. PN^2 and PN-DFPN are applied to solve some openings of Connect 6 on 7×7 and 9×9 boards. The experimental result shows that PN-DFPN is more efficient than PN^2.

Key words: computer game; proof-number search; PN^2; AND/OR tree; game-theoretical value; Connect 6

自从图灵提出"计算机博弈问题可以用来挑战计算机的智能水平"这一观点以来,实现计算机博弈问题的最终求解一直都是全世界机器博弈研究者们梦寐以求的事情. 为了获得博弈问题的理论解,必须尽可能地完全展开博弈树. 证据计数法(proof-number search[1])是一种基于与或树的搜索算法,它在搜索过程中要保存整个搜索树,而且它只处理结果为"是或否"的问题,所以被计算机博弈领域的学者们广泛应用于博弈问题的求解研究. 标准的证据计数(PN)法存在一些缺陷,因此出现了一些 PN 的改进算法,比如:DF-PN[2],PN^2[3],PDS-PN[4]等. 落子类机器博弈问题主要包括 k 子连珠棋(常见的主要有五子棋(Go-moku)、六子棋(Connect 6)等)和围棋(Go). 学者们在求解这些落子类博弈问题的过程中都用到了证据计数法[5-6]. 本文主要提出了一种两级 PN 算法,即 PN-DFPN 算法. 通过实验将此算法应用到求解小规模棋盘的六子棋开局局面中,结果证明此算法弥补了 PN^2 算法存在的缺陷,并且求解效率优于 PN^2 算法.

1 证据计数法

1.1 一种与或树模型

证据计数法是一种基于与或树的搜索算法.

收稿日期: 2015-05-01
基金项目: 国家自然科学基金资助项目(61370153).
作者简介: 高 强(1980-),男,辽宁沈阳人,东北大学博士研究生;徐心和(1940-),男,黑龙江哈尔滨人,东北大学教授,博士生导师.

如图 1 所示，树中只包括“或”节点（图 1 中的方块）和“与”节点（图 1 中的圆），每个节点的估值只有三种可能：“真”，“假”，“未知”. 被估值为“未知”的节点能够被扩展，能够被扩展的节点都是树的中间节点. 叶子节点有三个类型，其中被估值为“假”或“真”的节点属于终端节点；而被估值为“未知”的节点属于前端节点；未被估值的节点（可能由于内存空间已满或已经超过规定的搜索时间使得某些节点未被估值）也属于前端节点. “或”节点一般代表我方，因为有选择着法的主动权，哪个着法好便选哪个. 因此一个被扩展的“或”节点的值确定如下：如果此节点至少有一个子节点的值为“真”，则该节点的值为“真”；否则该节点的值为“假”. 而“与”节点一般代表对方，因为只能被动地等待对方出招，必然要考虑最为不利的局面. 因此，一个被扩展的“与”节点的值确定如下：如果此节点至少有一个子节点的值为“假”，那么该节点的值为“假”；否则该节点的值为“真”.

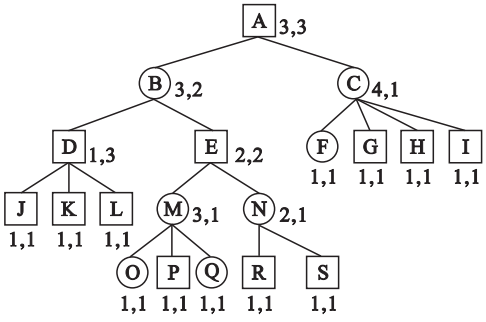

图 1　含有最可能证明节点的与或树

Fig. 1　AND/OR tree that includes MPN

1.2　证据计数法的定义及工作原理

证据计数法是一种最佳优先的搜索算法，同时它也是一种基于与或树的算法（见图 1）.

文献[1]给出了证据计数搜索算法的非形式化定义. 在上一节提到的与或树模型的基础上，树中每一个节点的值是一个二元组，即（proof number，disproof number）. 在证据计数法中，与或树被证明和被反驳的过程（通过不断反复地选择一个最可能证明的节点来实现证明和反驳的过程）并不冲突，并且是同步的[1]. 最可能证明的节点（most proving node，MPN）是证据计数法在搜索过程中选出的将要被展开的一个叶子节点. 选择 MPN 的原则为：对于“或”节点，需根据各个子节点的 proof number 项的数值进行选择；对于“与”节点，需根据各个子节点的 disproof number 项的数值来进行选择，文献[7－8]给出了相应的公式如下.

对于“或”节点，其二元组为

$$\left(\min_{i=1,\cdots,k} p_{\mathrm{n}}(n_i),\ \sum_{i=1}^{k} d_{\mathrm{n}}(n_i)\right), \tag{1}$$

对于“与”节点，其二元组为

$$\left(\sum_{i=1}^{k} p_{\mathrm{n}}(n_i),\ \min_{i=1,\cdots,k} d_{\mathrm{n}}(n_i)\right). \tag{2}$$

式中：$p_{\mathrm{n}}(n)$ 表示 proof number 项；$d_{\mathrm{n}}(n)$ 表示 disproof number 项；n 表示中间节点；$i = 1,\cdots,k$ 表示中间节点 n 的子节点下标. 本文以图 1 为例阐述选择 MPN 的过程.

1）树中的根节点 A 的计数值为（3，3），说明证明此树需要 3 个子节点，反驳此树同样也需要 3 个子节点，由于 A 为“或”节点，所以需根据其所有子节点的 proof number 项的值选择 MPN，B 的 proof number 项的值最小，所以选择节点 B 为 MPN；

2）节点 B 属于“与”节点，所以需根据其所有子节点的 disproof number 项的值选择 MPN，节点 E 的 disproof number 项的值最小，所以选择节点 E 为 MPN；

3）节点 E 的情况与根节点相似，因此选择其子节点 N 为 MPN；

4）节点 N 的情况与步骤 2）相似，但是它的两个子节点 disproof number 项的值相同，在这种情况下，选择最左侧的节点作为 MPN[1].

2　证据计数法在落子类机器博弈中的应用

2.1　证据计数法在五子棋中的应用

文献[5]采用迫着搜索（threat-space search，TSS）与证据计数法（PN）相结合的搜索引擎来求解五子棋. 在文献[5]所做的实验中，发现只使用迫着搜索算法在某些分支上可能无法找到必胜的策略（而实际上必胜策略存在于该分支中）. 所以在使用迫着搜索的基础上，引入了 PN 算法，PN 算法根据已经搜索过的结点的历史信息调整搜索方向，使之朝着当前代价最小的方向扩展搜索范围，弥补了迫着搜索算法的不足，采用这种混合算法的搜索引擎于 1993 年实现了对五子棋的最终求解.

2.2　证据计数法在六子棋中的应用

文献[6]将 PN 搜索应用到寻找六子棋的理论解中，采用分布式系统的方式并行地评估或展开前端节点，以提高 PN 的搜索效率. 文献[8]中，

对前端节点的 p_n 和 d_n 函数值采用加权值的方式进行差别化,实现了对前端节点的有效排序.

2.3 证据计数法在围棋中的应用

文献[9]中,采用 λ 搜索(一种基于与或树的搜索算法,用来衡量走棋方实现一个目标需要多少步)与深度优先的证据计数法相结合的方式完成对围棋吃子问题的搜索. 由于 λ 算法能够有效减小搜索空间,证据计数法又是一个搜索效率高的算法,而这两个搜索算法都是基于与或树(这更有利于两种算法相结合),因此这一混合的搜索算法更加有效.

3 证据计数法的缺陷及 PN^2 算法

3.1 证据计数法的缺点

1) 前端节点无差别. 根据证据计数法的定义,前端节点的值相等时首先选择最左端的节点进行展开. 这样的规定,将出现一些不必要的展开,从而浪费时间和内存空间,特别是对深度优先的 PN 算法的搜索效率影响非常大. 文献[9]针对这一缺点进行了改进.

2) 占用巨大的内存空间. 根据证据计数法的定义,必须要将整个搜索树存放到内存中,这将占用巨大的内存空间. DF-PN 搜索算法[2]、PN^2 算法[3]、PDS-PN 混合搜索算法[4],job-level 分布式处理技术[6]等都在一定程度上解决了这一问题.

3) 有向非循环图. 对于吃子或可移动棋子的博弈问题,在搜索树中的某一个局面能够经由不同路径到达. 这样,这个局面就对应多个父节点,那么在回溯更新时,某些节点的 p_n 和 d_n 就不会被更新. 对于落子类博弈问题来说,围棋存在这样的情况. 文献[2]提出了解决这一问题的方法.

3.2 PN^2 搜索算法

文献[1]提出了 PN^2 算法以减少 PN 算法所占用的内存空间,文献[3]对算法做了更详细的阐述. PN^2 由两个 PN 搜索组成,其中第一层 PN(称为 PN_1)调用第二层的 PN(称为 PN_2)来对 PN_1 的最佳前端节点进行评估. 由于 PN_2 受到能够存放在内存中的最大节点数的限制,文献[3]提出了一种对 PN_2 所占用内存的限制方法,定义了一个函数:

$$f(x)=\frac{1}{1+e^{(a-x)/b}}. \tag{3}$$

式中:x 表示 PN_1 搜索树的尺寸;a,b 为两个正整数. 显然,此函数随着变量 x 的增大而增大. 假设 N 表示可以存放在物理内存中的节点数,那么 PN_2 的最大尺寸为 $y=\min(xf(x),N-x)$. 当节点数已经超过 y 或者 PN_2 搜索树的根节点已被证明(proven)或反驳(disproven),停止 PN_2 搜索. 当完成 PN_2 搜索,其根节点的子节点被保存到内存中,而这些子节点所展开的搜索树所占用的内存将被释放. 最佳前端节点(PN_2 搜索树的根节点)的子节点只有在被称为最佳前端节点之后,才被估值.

由以上对 PN^2 算法的阐述可知,此算法受内存空间的严格限制,相对于 PN 搜索,虽然可以减少搜索树所占用的内存空间,但这也成为该算法的一个明显的缺陷,也就是说,当所限定的内存空间已满时,将停止搜索.

4 一种改进的两级 PN 算法

标准 PN 搜索所需内存空间太大,PN^2 虽然对搜索所需内存空间做了严格限制,但它同样存在"当内存已满时,搜索将停止"的缺陷,这限制了 PN^2 算法的求解能力. 本文提出了一种新的两级 PN 搜索,即 PN-DFPN 搜索算法.

4.1 PN-DFPN 算法的定义

4.1.1 第一级:PN 搜索

PN-DFPN 算法的第一级采用标准的 PN 搜索算法. 首先展开一层子节点,回溯更新父节点. 根据 PN 算法的选择原则,选择一个 MPN,根据 3.2 节给出的公式 $y=\min(xf(x),N-x)$,计算出第二级搜索允许占用的空间,将此 MPN 交给第二级搜索进行展开评估. 当第二级搜索结束并返回该 MPN 的评估时,第一级搜索将回溯更新父节点,再重新选择 MPN.

4.1.2 第二级:DFPN 搜索算法

DFPN 是一种深度优先的 PN 搜索[2],采用 $p_t(n)$ 和 $d_t(n)$ 来限制对节点 n 的展开程度,实际上是用来检测 MPN 是否在节点 n 的子树中. 对节点 n 展开的结束条件为

$p_n(n)\geqslant p_t(n)$ 或者 $d_n(n)\geqslant d_t(n)$.

若满足结束条件,说明 MPN 不在节点 n 的子树中,因此 DFPN 搜索将回溯到节点 n 的父节点;若不满足结束条件,则将继续展开节点 n,以寻找 MPN. 当最佳前端节点被选定以后,p_t 和 d_t 将被更新;假设 DFPN 正在检测"或"节点 p,并且其子节点 c_1 是 p 的所有子节点中含有最小 p_n 值的节点,并设 p_{n_2} 仅大于 p_n 值,那么 DFPN 以下列 p_t 和 d_t 来检验节点 c_1:

$$\left.\begin{aligned} p_t(c_1) &= \min(p_t(p), p_{n_2}+1), \\ d_t(c_1) &= d_t(p) - (d_n(p) - d_n(c_1)). \end{aligned}\right\} \quad (4)$$

这里 $d_n(p) - d_n(c_1)$ 就是除了节点 c_1 以外，节点 p 所有子节点的 d_n 之和. 当 $p_n(c_1) \geqslant p_t(c_1)$ 时，说明“或”节点 p 的子节点 c_1 的 p_n 值在子节点中不是最小的，因此在 c_1 的子树中不存在 MPN. $d_t(c_1)$ 设定是为了保证节点 p 的所有子节点的 d_n 之和不超过 $d_t(p)$. 也就是说，$d_t(c_1)$ 用来记录节点 p 的子节点数量.

相似地，假设 DFPN 正在检测“与”节点 p，设 d_{n_2} 仅大于 d_n（其中 d_n 是 p 所有子节点中的最小值），那么 DFPN 以下列 p_t 和 d_t 来检验节点 c_1：

$$\left.\begin{aligned} p_t(c_1) &= p_t(p) - (p_n(p) - p_n(c_1)), \\ d_t(c_1) &= \min(d_t(p), d_{n_2}+1). \end{aligned}\right\} \quad (5)$$

以图 2 为例，详细阐述第二级 DFPN 搜索的原理. 首先，DFPN 算法设置节点 A 的 $p_t(A) = d_t(A) = \infty$. 对于“或”节点 A 来说，由于其子节点 $p_n(B) < p_n(C)$，根据 PN 算法的规则，所以选择节点 B. 接下来计算节点 B 的 p_t 和 d_t：$p_t(B) = \min(p_t(A), p_n(C)+1) = p_n(C)+1 = 4$，$d_t(B) = d_t(A) - (d_n(A) - d_n(B)) = \infty - 1$. 对于“与”节点 B，由于其子节点 $d_n(D) < d_n(E)$，因此选择节点 D 作为 MPN. 接下来计算节点 D 的 p_t 和 d_t. 根据式(5)，可得 $d_t(D) = \min(d_t(B), d_n(E)+1) = 3$，$p_t(D) = p_t(B) - (p_n(B) - p_n(D)) = p_t(B) - p_n(E) = 4 - 1 = 3$. 只要 $p_n(D) < 3$ 并且 $d_n(D) < 3$，DFPN 将继续展开节点 D 的子树而不会回溯更新其父节点，因为在节点 D 的子树中包含 MPN.

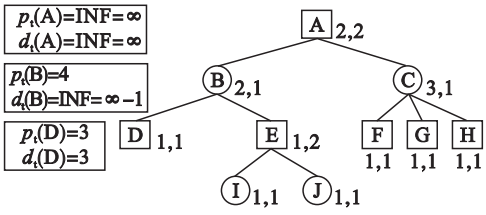

图 2　DFPN 实例
Fig. 2　Example for DFPN search

由以上对 DFPN 算法的阐述可知，如果节点的 p_n 或 d_n 满足展开的结束条件，那么此节点被展开的子树将被删掉，而 PN 在搜索过程中，需要保存整棵树，所以采用 DFPN 搜索代替第二级的 PN 搜索可以节省存储空间. DFPN 算法是深度优先的，因此它对节点排序是敏感的，本文在 DFPN 算法的基础上，针对每一层展开的子节点根据其估值的大小进行排序，这样使得该算法可以更快地找到 MPN，进而使得这个算法占用的内存空间要小于 PN 搜索，相对于 PN^2 算法，提高了对问题的求解能力.

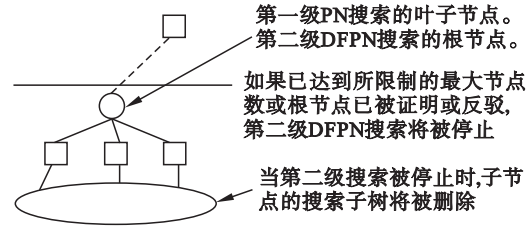

图 3　PN-DFPN 搜索
Fig. 3　PN-DFPN search

图 3 显示了 PN-DFPN 算法的整体结构. 在 PN-DFPN 算法中，当第一级 PN 搜索调用第二级 DFPN 搜索时，DFPN 搜索以第一级选出的 MPN 为根节点展开搜索，若分配给第二级搜索的空间已满或展开的前端节点被评估为“真”或“假”，则结束 DFPN 搜索并删除搜索过程中展开的子树，然后返回根节点（即第一级选出的 MPN），即将 proof number 和 disproof number 值返给第一级 PN 搜索.

4.2　应用于求解六子棋的 PN-DFPN 算法设计

本节将 PN-DFPN 搜索算法应用于求解六子棋局面，这里只对第一级 PN 搜索的“或”类型节点的子节点进行 DFPN 搜索（因为在六子棋中，“或”类型节点的分支因子远大于“与”类型节点[8]，也更容易出现多个 p_n 值相等的前端节点），利用深度优先搜索算法的特性，尝试能否更快地搜索到终端节点，即 proven 或 disproven 节点. 具体算法如下.

1）以六子棋开局局面作为根节点进行第一级 PN 搜索，如果搜索到终端节点，即 proven 或 disproven 节点，则结束 PN-DFPN 搜索；

2）如果未搜索到终端节点，而最佳前端节点为“或”类型节点的子节点，利用估值对这些节点（最佳前端节点可能是多个）进行排序，只对估值最高的节点调用第二级 DFPN 搜索算法；

3）DFPN 搜索以这个前端节点作为根节点，进行深度优先的搜索：

- 若存在 proven 或 disproven 节点，说明第一级搜索的根节点已被证明或反驳，回溯更新第一级搜索树的根节点并结束 PN-DFPN 搜索；
- 若对 DFPN 搜索算法限制的空间已满，并且未证明或反驳第一级 PN 搜索的根节点，则结束 DFPN 搜索，执行步骤 1）.

为了提高搜索效率，在 DFPN 搜索中，对每一

层的若干个子节点,根据估值大小降序排列.

5 实 验

分别以棋盘规模为 7×7 和 9×9 的六子棋为例,采用存在 7 个棋子的 380 个开局局面作为测试数据,分别使用 PN^2,PN-DFPN 搜索算法对这些局面进行求解,测试比较这两个算法所占用的空间(这里以搜索过的总节点数为标准)大小、完成求解所需时间(以 ms 为单位)以及求解能力.

首先,在两个算法均实现求解 7×7 棋盘上所有 380 个局面的情况下,测试并比较访问的总节点数和消耗的总时间. 为了更加准确地体现各个算法搜索效率的优劣,统一设定 PN^2 和 PN-DFPN 的函数 $f(x)$ 中 a 和 b 两个参数值. 对于 PN^2,由于此次用来实验的局面相对复杂度比较低,根据文献[3],参数 a 的值应该设定得比较大,这样可以提高搜索效率,因此将 (a,b) 设定为 $(36\times2^{10},4.8\times2^{10})$. 由表 1 的数据可知,PN-DFPN 的搜索效率要优于 PN^2,这说明对以上测试局面求解,在单位时间内 PN^2 构建的搜索树更加庞大.

表 1 7×7 棋盘上算法的比较

Table 1 Comparison of the algorithms on 7×7 board

算法	总节点数	时间/ms
PN^2	11 378	21 923
PN-DFPN	9 793	19 590

对于计算机博弈问题,棋盘规模越大,越难以求解,因此,本文选择更难解的棋盘规模为 9×9 的六子棋,以此棋盘上存在 7 个棋子的 380 个局面作为测试数据,进一步比较两个算法的求解效率(见表 2),可见 PN-DFPN 的优势更加明显.

表 2 9×9 棋盘上算法的比较

Table 2 Comparison of the algorithms on 9×9 board

算法	总节点数	时间/ms
PN^2	22 790	112 842
PN-DFPN	11 325	53 554

将搜索的节点数限制在 5 000,测试这两个算法的搜索效率和求解能力,仍然采用7×7棋盘上存在 7 个棋子的 380 个局面作为测试数据. 由表 3 的数据可知,虽然 PN-DFPN 完成的时间要略多于 PN^2,但 PN-DFPN 完成求解的局面数要多于 PN^2,再一次说明 PN^2 在求解局面时,所需存储空间更大.

表 3 算法求解局面数和消耗时间的比较

Table 3 Comparison of the algorithms on the number of games and the time consumed

算法	完成求解的局面数	时间/ms
PN^2	178	9 683
PN-DFPN	199	10 972

为了进一步比较 PN-DFPN 算法和 PN^2 算法的求解能力,本文分别将两种算法所搜索的总节点数限制在 4 000,3 000,2 000,1 000,图 4 显示了在这些限制下,两种算法完成求解的局面数.

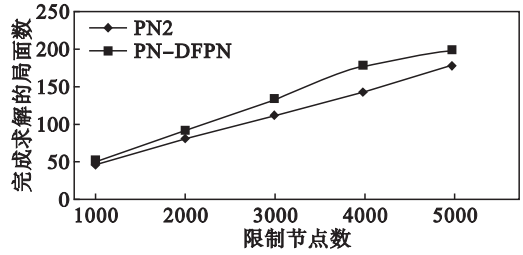

图 4 算法求解能力的比较

Fig. 4 Comparison of the algorithms on resolution

通过以上实验可以得出,两个算法在完成求解同样六子棋开局局面情况下,PN-DFPN 算法在消耗的空间和时间上都少于 PN^2 算法,这说明在求解过程中,PN-DFPN 算法构建的搜索树的尺寸更小;当规定存储在内存中的节点数分别限制在 1 000,2 000,3 000,4 000,5 000 个节点的情况下,PN-DFPN 算法完成求解的局面数更多,优势更明显.

6 结 语

本文详细介绍了证据计数法的原理,并综述了证据计数法在求解一些落子类博弈问题上的应用,对证据计数法存在的一些缺陷进行了论述,提出了一种两级 PN 算法,即 PN-DFPN 算法. 将此算法应用到求解 7×7 和 9×9 棋盘的六子棋开局局面中,结果表明此算法弥补了 PN^2 算法存在的缺陷,求解效率明显优于 PN^2 算法.

参考文献:

[1] Allis V. Searching for solutions in games and artificial intelligence[D]. Maastricht:University of Limburg,1994.

[2] Nagai A. Df-pn algorithm for searching AND/OR trees and its applications[D]. Tokyo:University of Tokyo,2002.

[3] Breuker D M. Memory versus search in games [D]. Maastricht:Maastricht University,1998.

[4] Winands M,Uiterwijk J,van den Herik H J. An effective two-level proof-number search algorithm [J]. *Theoretical Computer Science*,2004,313(3):511 - 525.

(下转第 1103 页)