数独问题的求解、评价与生成算法的研究

王 琼[1,2], 邹 晟[3]

(1.南京师范大学 计算机科学与技术学院, 江苏 南京 210097;
2.江苏省信息安全与保密工程研究中心, 江苏 南京 210097;
3.南京师范大学 中北学院, 江苏 南京 210097)

[摘要] 将数独问题分解为求解初盘、难度评价、生成有解初盘、生成有唯一解初盘等子问题. 为求解初盘, 提出了基于最小候选数的搜索算法, 并基于算法中的判定树, 给出了难度指标的计算方法. 生成有唯一解初盘的算法分为两步: 首先生成有解初盘集合, 再利用判定树进行筛选.

[关键词] 数独, 候选数, 搜索算法, 判定树
[中图分类号] TP311.12 [文献标识码] A [文章编号] 1672-1292(2010)01-0076-04

Study on Solution, Evaluation and Generation Algorithm of Sudoku Problem

Wang Qiong[1,2], Zou Sheng[3]

(1. School of Computer Science and Technology, Nanjing Normal University, Nanjing 210097, China;
2. Jiangsu Research Center on Information Security and Confidential Engineering, Nanjing 210097, China;
3. Zhongbei School, Nanjing Normal University, Nanjing 210097, China)

Abstract: In the paper, Sudoku problem is divided into solving original layout, calculating difficulty indicator, generating original layout of solvable, generating original layout which has a unique solution. To solve original layout, search algorithm based on the minimum candidate number is proposed. Calculation method of game difficulty is established with decision tree. Generating original layout which has a unique solution is divided into two steps: first, solvable original layouts are generated, and then, desired layouts are selected by decision tree.

Key words: Sudoku, candidate number, search algorithm, decision tree

1 问题的分析

"数独"游戏由数学家欧拉发明, 目前在国内外非常流行. 游戏在 9×9的单元网格中进行, 单元网格不仅被分为 9行、9列, 也被分为 3×3个九宫格. 单元网格中已存在若干数字, 其余为空格. 游戏规则要求玩家在每个空格中填入 1~9之间的数字, 使每个数字在每行、每列、每个九宫格仅出现一次.

为便于描述, 本文首先确定以下名词:

初盘: 游戏开始时的单元网格状态, 其中包含若干数字和若干空格.

终盘: 游戏成功结束时的单元网格状态, 其中只包含数字, 不包含空格.

布局: 在从初盘求解终盘的过程中单元网格的状态.

死局: 布局中存在空格, 但在任何空格中填入任何数字, 都与游戏规则冲突.

国内许多论文对数独游戏的教学意义做了深入讨论, 但研究其求解算法的论文不多. 一般着重于讨论利用回溯法求解终盘的方法[1,2]. 也有文章结合 2008年美国 MCM赛题, 讨论不同难度级别的数独初盘的生成问题[3], 但其难度指标定义过于主观, 讨论也不够细致. 还有一些论文从遗传算法的角度来讨论初盘求解的问题[4,5].

本文认为数独问题可分为以下 4个子问题:

① 求解初盘: 由初盘求解终盘. 若存在多个解, 应列出所有终盘.

收稿日期: 2009-12-18
通讯联系人: 王 琼, 副教授, 研究方向: 算法与程序设计. E-mail: wangqiong@njnu.edu.cn

② 难度评价: 在求解初盘的过程中, 游戏难度应由计算量和选择难度决定.

③ 生成有解初盘: 生成的初盘必须有终盘解.

④生成有唯一解的初盘: 若一个初盘对应着多个终盘解, 可能会降低游戏的可玩性、趣味性, 所以应寻求生成有唯一终盘解的初盘.

2　求解初盘及难度评价

2.1　玩家玩法的分析

数独游戏的玩法是求解初盘的方法, 一般分为直观法和候选数法.

直观法是将盘面中已有的数字与所在的行、列、宫的约束相结合, 寻找能唯一确定数字的空格. 具体技巧分为: 单元唯一法、单元排除法、区块排除法、唯一余数法、组合排除法、矩形排除法. 虽然这类方法玩家最常使用, 但有时在布局中难以识别何处可以使用这种方法.

候选数法的基本运算是计算空格的候选数. 一个空格的候选数是指可能填入该单元的若干数字, 它是1~ 9除去同行、同列、同宫中已存在数字的集合. 候选数法以候选数计算为基础, 结合布局的特征, 即相关行、列、宫中候选数的规律, 得到某空格中应填的数字. 具体技巧分为: 显式唯一法、隐式唯一法、区块删减法、显式数对法、显式三数集法、显式四数集法、隐式数对法、隐式三数集法、隐式四数集法、矩形对角线法、*XY*形态匹配法、*XYZ* 形态匹配法、三链数删减法、*WXYZ* 形态匹配法.

玩家的技巧有数十种, 各自适应不同特征的布局. 一般求解初盘的过程, 必定需要多种技巧的组合, 且无一定之规. 为保证求解初盘算法能成功求解一切有解初盘, 应避免算法对布局特征的依赖.

2.2　基本运算

显然, 数独游戏的布局应采用矩阵结构表示. 为描述后继算法, 设计以下基本运算:

function Candidates= Check(*S*, *i*, *j*).

函数 Check计算在布局 *S* 中, 第 *i*行第 *j*列的单元的所有候选数向量 ***Candidates***:

function [*i*, *j*] = Find***Candidate***(*S*).

函数 Find***Candidate*** 在布局 *S* 中, 查找具有最少候选数的单元 [*i*, *j*]. 设 *S* 中空格数为 *k*, 则在Find***Candidate***的执行中, 需要调用 *k*次函数 Check:

function flag= Success(*S*).

函数 Success判断布局 *S* 是否已是终盘:

function flag= Failed(*S*).

函数 Failed判断布局 *S* 是否是死局.

2.3　基于最小候选数的搜索算法

求解初盘的基本方法是, 采用回溯法穷举状态空间. 但在每个步骤中, 对哪个空格试填何值, 将极大地影响状态空间的结构, 进而影响算法的效率. 一般认为求解初盘的状态空间是个深度在 81以内的 9叉树.

笔者采用多叉树遍历算法框架, 结合最小侯选数的计算, 设计了以下算法. 将初盘记作 S_0, 求解过程中第 *k*个布局记作 S_k, 存储所有中间布局的数据结构记作栈 *SS*, 求解 S_k 所有终盘的递归算法 Solve(S_k)描述如下:

① 若 Success(S_k) = = true, 即 S_k 是终盘, 则栈 *SS* 中保留的 S_1, …, S_k 是初盘至一个终盘的求解过程, 返回上层函数调用;

② 若 Failed(S_k) = = true, 即 S_k 是死局, 返回上层函数调用;

③ 查找 S_k 中最少候选数的空格位置 [*i*, *j*] = Find***Candidate***(S_k);

④计算位置 [*i*, *j*]的候选数向量 ***Candidates*** = Check(S_k, *i*, *j*);

⑤依次将 ***Candidates***中的每个候选数填入 S_k 的空格 [*i*, *j*]中, 得到新布局 S_{k+1}, 将 S_{k+1}进栈 *SS*, 递归调用 Solve(S_{k+1})之后, *SS* 执行出栈操作.

本质上, 上述算法是改良的多叉树搜索算法. 改良之处在于, 每次填入数字的空格, 是具有最少候选数的空格, 极大地减少了搜索分支, 达到了修枝限界法的效果.

据数独网站的资料介绍, 目前已知的有唯一解的数独初盘至少有 17个已知数. 笔者搜集到 4万余个

此类数独初盘,将其作为算法测试数据,在普通微机上的实际执行时间约在 ms级别.究其原因是在求解有唯一解的初盘的过程中,每步的最小候选数一般小于等于 2,因此算法的实际性能达到了线性时间复杂度.

2.4 难度评价

对数独的难度评价,常常因玩家的思路不同,而标准各异.笔者以为,难度包括以下 2个方面:

①选择的难度:在每个试填步骤中,具有最少候选数的空格所拥有的候选数个数,决定了搜索的分支数,也考验了玩家的选择灵感.

笔者将求解过程描述为一个判定树,如图 1所示.树中结点中的数字 (m, n), m 表示对应布局中的空格数, n 表示其中最少候选数单元的候选数个数.若 $m > 0$,且 $n = 0$,则表明对应布局为死局;若 $m = 0$,且 $n = 0$,则对应布局为终盘.该判断树表明初盘 S_0 有两个终盘解 S_1 和 S_2, $S_{1,1} \sim S_{1,5}$依次表示求解 S_1 的中间布局, $S_{2,1} \sim S_{2,5}$依次表示求解 S_2 的中间布局.其中 $S_{1,1}$与 $S_{2,1}$相同, $S_{1,2}$与 $S_{2,2}$相同.

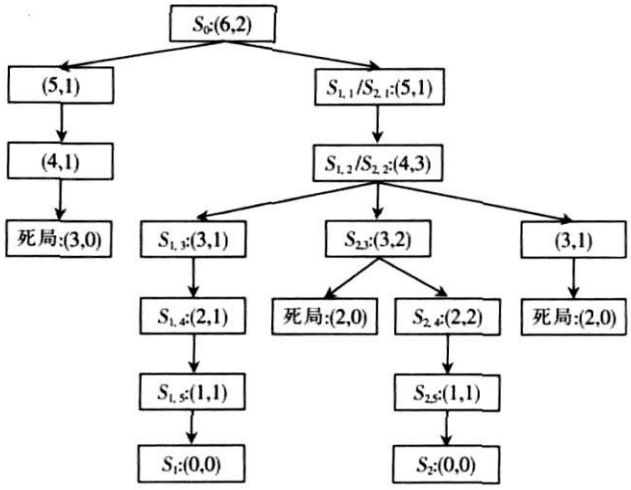

图 1　一个有 6个空格的数独游戏求解过程的判定树

Fig.1　Decision tree of solving Sudoku with spaces

为计算一个判定树中的选择难度,笔者设计了如下方法:

设树中的表示终盘的叶子结点数为 m_1,表示死局的叶子结点数为 m_2,则选择难度为 $m_2/(m_1 + m_2)$.若所有叶子结点都是终盘时,选择难度为 0,图 1中存在 3个死局和 2个终盘,则选择难度为 60%.

②计算的难度:将计算一个空格的候选数视作一条基本运算,判定树中的每个结点的计算量是 m (m 是结点中的空格数),进而可以得到游戏的全部计算量.例如图 1中的全部计算量是 46,为所有节点的 m 值的总和.

虽然玩家在游戏过程未必经历所有分支的计算,但这种计算量还是比较客观,值得参考的.

整个初盘的难度应是选择难度与计算难度的调和函数.至于权重的选择,应由游戏组织者根据游戏目标加以设定.

3 生成初盘

3.1 生成有解初盘

一般说来,生成有解初盘的方法分成两步:得到合理的终盘;从终盘删除若干数字作为初盘.第一步并非易事,在 9×9的单元网格,可以穷举出 9^{81}种状态,据维基百科中数独条目的介绍,德国数学家 Bertram Felgenhauer在 2005年给出数独终盘有 $9! \times 72^2 \times 2^7 \times 27$,计 704 267 971种.

分析初盘与终盘的关系,可以发现:一个初盘中的已知数字,其本质是对终盘解的若干约束.若初盘中没有已知数字,即约束为空,则其对应的解就是全部可能的终盘.遗憾的是,这个求解初盘的算法在没有任何约束条件之下,表现出来的时间复杂度是指数级的.因此,笔者提出了以下改进的生成有解初盘的算法:

①在 9×9的单元网格,随机填写 k 个 1~ 9之间的数字,得到初盘 S_0;

②检测 S_0 中每个行、列、宫中, 是否出现重复数字, 若出现, 则转①, 否则转③;

③利用 Solve算法求解初盘 S_0, 若 S_0 无解, 则转①;

④若 S_0 有 k 个终盘解, 记作 S_1、S_2、…、S_k, 在 S_0 与 S_i ($i \in [1, k]$)之间的所有中间布局记作 S_0、$S_{i\ 1}$、$S_{i\ 2}$、…、S_i, 则算法结果是 S_0 和全部的布局 S_{ij}.

算法中 k 的取值值得商榷. 若较大值, 可能增加步骤②的计算量及①②的重复次数; 若取较小值, 可能导致较多的终盘解, 增加步骤③④的计算量. 笔者建议 k 取值在 17~45之间.

若 S_0 有多个终盘解, 其解空间的结构如图 1所示.

3.2　生成有唯一解初盘

生成有唯一解初盘更有应用意义. 考察图 1中的判定树, 沿叶子节点 S_1 上溯, 从 $S_{1\ 5}$到 $S_{1\ 3}$都只最终导致一个终盘, 所以都是有唯一解的初盘. 同理, $S_{2\ 5}$至 $S_{2\ 3}$也是有唯一解的初盘. 但 $S_{2\ 1}$和 $S_{2\ 2}$是有两个解的初盘.

据上分析, 笔者提出以下算法, 分析 3.1的算法结果, 以获得有唯一解的初盘.

①根据 3.1的算法结果, 构造判定树;

②对每个表示终盘解的叶子结点 S_i 做如下处理:

③沿 S_i 上溯, 依次分析其祖先结点, 直至某祖先结点的子树中存在多个终盘解. 其间的所有分支结点都是有唯一解的初盘.

4　结语

本文提出的“基于最小候选数的搜索算法”, 可以求解出初盘的所有终盘解. 该算法针对一般游戏中的初盘, 性能可以近似达到线性时间复杂度. 依据求解初盘算法中的判定树结构, 给出了游戏中的计算量指标和选择难度指标的计算方法. 基于求解初盘算法, 提出了一个生成有解初盘的算法. 基于判定树结构, 给出了生成有唯一解的初盘的算法.

[参考文献] (References)

[1] 雷蕾, 沈富可. 关于数独问题的算法的设计与实现[J]. 电脑知识与技术, 2007, 2(2): 481-482.
Lei Lei, Shen Fuke. The design and implementation of the algorithm about Sudoku[J]. Computer Knowledge and Technology, 2007, 2(2): 481-482. (in Chinese)

[2] 李盘荣. “数独”游戏的算法研究与实现[J]. 电脑知识与技术, 2008, 3(8): 1715-1717.
Li Panrong. The research and implementation of the algorithm about Sudoku[J]. Computer Knowledge and Technology, 2008, 3(8): 1715-1717. (in Chinese)

[3] 赵志芳, 郭静鑫, 杨璐. 生成 Sudoku的算法探究[J]. 内江科技, 2008(7): 22-23.
Zhao Zhifang, Guo Jingxing, Yang Lu. Research of algorithm for generation of Sudoku[J]. Neijiang Science and Technology, 2008(7): 22-23. (in Chinese)

[4] Timo Mantere, Janne Koljonen. Solving, rating and generating sudoku puzzles with GA[C]//2007 IEEE Congress on Evolutionary Computation. Singapore, 2007.

[5] Timo Mantere, Janne Koljonen. Solving and analyzing Sudokus with cultural algorithms[C]//2008 IEEE Congress on Evolutionary Computation. Hong Kong, 2008.

[责任编辑: 严海琳]