第 38 卷 第 10 期
Vol.38 No.10

计 算 机 工 程
Computer Engineering

2012 年 5 月
May 2012

·人工智能及识别技术· 文章编号：1000—3428(2012)10—0161—03 文献标识码：A 中图分类号：O157

数独难度级别划分的权重法研究

吴林波，肖华勇，杨宗泽

(西北工业大学理学院，西安 710129)

摘 要：给出候选数模式下模仿人工智能求解数独的一系列填数及删减规则，在此基础上提出模仿人工智能的求解算法及数独难度衡量方法。从数独博士 5 个难度级别中随机抽取各 100 道题目，采用难度衡量标准重新分级，并将结果与数独博士等级划分标准做相关性检验，得到 Goodman-Kruskal 相关系数 r=0.82，说明该标准与数独博士的难度划分标准有较强的相关性，并给出随机生成数独题目的算法。通过难度衡量方法与生成算法，可以随机生成 5 个不同难度的数独谜题。

关键词：数独；填数及删减规则；人工智能；难度衡量；相关系数；随机生成

Research of Weight Method with Sudoku Difficulty Level Partition

WU Lin-bo, XIAO Hua-yong, YANG Zong-ze

(School of Science, Northwestern Polytechnical University, Xi'an 710129, China)

【Abstract】This paper gives a few filling and deleting rules to imitate artificial intelligence to solve Sudoku under the candidate mode, and based on the proposed artificial intelligence rules, it constructs the difficulty scale and the algorithm of artificial intelligence solving. Validating the scale with a sample of 500 Dr.Sudoku puzzles rated externally into five gradations of difficulty. Goodman-Kruskal r coefficient is 0.82, indicating significant correlation the two scales. It gives an algorithm that can generate five different levels of Sudoku puzzles randomly.

【Key words】Sudoku; filling and deleting rules; artificial intelligence; difficulty measure; correlation coefficient; random generation

DOI: 10.3969/j.issn.1000-3428.2012.10.049

1 概述

数独名字来源于日本语 Sudoku，是 18 世纪瑞士数学家欧拉发明的。后在美国发展，并在日本得以流行。Sudoku 的规则很简单，就是在 9×9 的九宫格(区块)里面填写数字，每个方格中填入合适的数字使其满足每行、每列以及每个九宫格都要包含从 1~9 的数字各 1 次。数独的玩法逻辑简单，而且数字排列的方式千变万化。很多教育者认为数独是锻炼思维的好方法。谜题中会预先填写若干数字，而其他方格为空白，玩家要依据谜题中的数字分布情况，逻辑地推敲出剩下的空格里是什么数字。 由于规则简单的原因，在推敲的过程中完全不必用到数学计算，而只需运用逻辑推理能力，因此无论男女老幼都可以玩，而且非常容易上手，容易入迷。

目前已有很多的求解数独的算法：递归法求解数独[1]，回溯候选数法求解数独[2]，枚举算法求解数独[3]，遗传算法求解数独[4]，几何粒子群算法求解数独等。这些求解数独的算法已经很系统全面了，但是在数独的难度级别划分与生成数独算法方面的研究还很少。

本文给出建立生成数独及其难度衡量的一套算法。首先提出候选数模式下的模仿人工智能的一系列填数及候选数删减规则：显式唯一候选数法(Naked Single)，隐式唯一候选数法(Hidden Single)，显式二数集删减法(Naked Pair)，隐式二数集删减法(Hidden Pair)，显式三数集删减法(Naked Triplet)，隐式三数集删减法(Hidden Triplet)，显式四数集删减法(Naked Quad)，隐式四数集删减法(Hidden Quad)，区块删减法(Intersection Removel)，矩形对角线删减法(X-wing)，三链数删减法(Swordfish)，XY 形态匹配法(XY-wing)，XYZ 形态匹配法(XYZ-wing)，WXYZ 形态匹配法(WXYZ-wing)，试探法等。然后，在这些规则之上，建立纯粹的模仿人工智能的求解算法及难度衡量标准，并给出了生成 5 个难度级别的划分成算法。

2 算法及难度衡量

2.1 候选数模式下的填数及删减规则

以下规则大体上是按多数人们认知的从容易到难的顺序排列的。

规则 1 显式唯一候选数法(Naked Single)：若某行，或列，或九宫格中的某个空格候选数唯一，那么该候选数就填该空格。

规则 2 隐式唯一候选数法(Hidden Single)：若某个候选数只出现在某行，或列，或九宫格的某个空格中，那么该空格就填该候选数。

规则 3 显式二数集删减法(Naked Pair)：若某行，或列，或九宫格中的某 2 个空格的候选数恰好为 2 个，那么这 2 个候选数可以从该行，或列，或九宫格的其他空格候选数中删除。

规则 4 隐式二数集删减法(Hidden Pair)：若某 2 个候选数只出现在某行，或列，或九宫格中的某 2 个空格中，那么这 2 个空格中不同于这 2 个候选数的其他候选数可删除。

规则 5 显式三数集删减法(Naked Triplet)：若某行，或列，或九宫格中的某 3 个空格的候选数恰好为 3 个，那么这 3 个候选数可以从该行，或列，或九宫格的其他空格候选数中

基金项目：西北工业大学大学生创新基金资助项目“数独难度级别划分与生成算法研究”

作者简介：吴林波(1986－)，男，硕士，主研方向：人工智能；肖华勇，副教授、博士；杨宗泽，本科生

收稿日期：2011-06-28 **E-mail：**15091899361@126.com

删除。

规则 6 隐式三数集删减法(Hidden Triplet)：若某 3 个候选数只出现在某行，或列，或九宫格中的某 3 个空格中，那么这 3 个空格中不同于这 3 个候选数的其他候选数可删除。

规则 7 显式四数集删减法(Naked Quad)：若某行，或列，或九宫格中的某 4 个空格的候选数恰好为 4 个，那么这 4 个候选数可以从该行，或列，或九宫格的其他空格候选数中删除。

规则 8 隐式四数集删减法(Hidden Quad)：若某 4 个候选数只出现在某行，或列，或九宫格中的某 4 个空格中，那么这 4 个空格中不同于这 4 个候选数的其他候选数可删除。

规则 9 区块删减法(Intersection Removel)

区块对行的影响：在某一区块中，当所有可能出现某个数字的单元格都位于同一行时，就可以把这个数字从该行的其他单元格的候选数中删除。

区块对列的影响：在某一区块中，当所有可能出现某个数字的单元格都位于同一列时，就可以把这个数字从该列的其他单元格的候选数中删除。

行或列对区块的影响：在某一行(列)中，当所有可能出现某个数字的单元格都位于同一区块中时，就可以把这个数字从该区块的其他单元格的候选数中删除。

规则 10 矩形对角线删减法(X-wing)：如果一个数字正好出现且只出现在某 2 行(列)相同的 2 列(行)上，则这个数字就可以从这 2 列(行)上其他单元格的候选数中删除。

规则 11 三链数删减法(Swordfish)：如果某个数字在某 3 列(行)中只出现在相同的 3 行(列)中，则这个数字将从这 3 行(列)上其他的候选数中删除。

规则 12 XY 形态匹配法(XY-wing)：若 XY 和 YZ 同在一个区块但不同行(列)中，而 XZ 和 XY 在同一行(列)，但在不同区块中。那么在 XY 所在区块中与 XY 所在行(列)交集空格中应该删除候选数 Z，并且在 XZ 所在区块与 YZ 所在行(列)交集的空格中应该删除候选数 Z。其中，XY、YZ、XZ 分别是三空格的候选数，并且这三空格没有其他候选数。

规则 13 XYZ 形态匹配法(XYZ-wing)：若某区块某空格候选数为 XYZ，在该同区块但不同列(行)的某空格候选数为 YZ，且与 XYZ 所在空格同列(行)但不同区块某空格候选数为 XZ，那么 XYZ 所在区块与 XZ 所在列(行)的交集中的空格候选数不能为 Z。

规则 14 试探法：若某个空格的候选数只有 2 个时，进行试探填写其中一个候选数，若填写成功则该试探成功，若导致矛盾则另外一个候选数应填该空格。

2.2 难度衡量标准

在建立难度衡量标准[5]之前，假设人工求解算法充分使用简单规则 i 直到此简单规则不再起作用时才使用下一条更高级的规则 $i+1$。

假设 n_i 为规则 i 的使用次数，w_i 为规则 i 的权重。规则 $i+1$ 要比规则 i 难度大，并且对于同一个规则 i，使用 2 次要比 1 次难度大。所以，为了体现数独题目使用不同规则及其相应次数的复杂度，试图给不同规则加上适当的权重，进而用规则的使用次数和相应的权重衡量数独题目的难度。在实际应用中，一般难度级别较高的规则要比难度级别较低的规则使用次数要相应少一些，然而这些难度级别较高的规则在该题目难度等级衡量中占了很重要的分量。 因此，对于难度级别较高的规则，相应给予他们较高的权重。本文中对于权重系统 $w_i(1\leqslant i\leqslant 14)$，取 $w_1=1$，$w_2=5$，$w_3=10$，$w_4=15$，$w_5=20$，$w_6=25$，$w_7=30$，$w_8=35$，$w_9=40$，$w_{10}=500$，$w_{11}=1\,000$，$w_{12}=3\,000$，$w_{13}=5\,000$，$w_{14}=8\,000$。对于上述权重系统，按照随规则递增权重递增以及尽可能使得不同规则间的权重差异足够大的原则进行取值。虽然权重的取值有一定的随意性，但实验结果表明这样的权重系统能够很好地区分不同难度等级的数独题目。

建立一个简单的难度评价函数：

$$D(n_1,n_2,\cdots,n_{14})=\sqrt{\frac{\sum_{i=1}^{14}w_in_i}{\sum_{i=1}^{14}n_i}}$$

例如：成功求解一个难度等级为较难的数独题目使用了 55 次规则 1，6 次规则 3，17 次规则 6，88 次规则 9，4 次规则 10，1 次规则 13，1 次规则 14。那么该题目的难度系数：

$$D=\sqrt{\frac{1\times55+10\times6+25\times17+40\times88+500\times4+5\,000\times1+8\,000\times1}{55+6+17+88+4+1+1}}\approx 10.5$$

难度评价函数中的开二次方根的操作是为了使难度评价函数的值所属区间尽可能小。

从 KLSuduku 软件中的容易、普通、困难、极难和骨灰等 5 个不同难度级别中随机抽取了各 100 道数独题目，然后用本文的人工智能求解算法求解它们，并分别记录下求解 5 个不同难度级别中数独题目所使用的各规则的平均次数及相应级别中的所有题目平均难度系数 D，如表 1 所示。

表 1 不同难度级别使用各规则平均次数相关信息

规则	容易	普通	困难	极难	骨灰
1	37.8	60.0	61.6	55.3	53.2
2	0.0	0.0	0.0	0.0	0.0
3	0.0	1.8	3.0	1.9	1.8
4	0.0	0.4	1.0	0.6	0.6
5	0.0	0.4	1.0	0.3	0.2
6	0.0	3.7	10.7	11.7	13.8
7	0.0	0.0	0.1	0.0	0.0
8	0.0	0.1	0.6	0.4	0.5
9	0.0	14.5	47.8	57.0	37.0
10	0.0	0.0	0.1	2.4	3.4
11	0.0	0.0	0.1	0.5	0.4
12	0.0	0.0	0.0	0.2	0.9
13	0.0	0.0	0.1	0.7	1.0
14	0.0	0.0	0.1	0.5	1.3
D	1.0	3.0	5.1	9.7	13.9

从表 1 可以看出，从 KLSuduku 软件中随机抽取的各不同难度级别的 100 道数独题目，使用各规则的平均次数及平均难度系数。例如求解 100 道难度级别为普通的数独题目，使用规则 1 的平均次数为 60 次，使用规则 3 的平均次数为 1.8 次，使用规则 4 的平均次数为 0.4 次，使用规则 5 的平均次数为 0.4 次，使用规则 6 的平均次数为 3.7 次，使用规则 8 的平均次数为 0.1 次，使用规则 9 的平均次数为 14.5 次，使用其余规则的平均次数为 0 次。根据 KLSuduku 软件中该 100 道难度级别为普通的题目使用的各规则平均次数，可以用上面的难度评价函数计算得到这些题目的平均难度系数：

$$D=\sqrt{\frac{1\times60+10\times1.8+15\times0.4+20\times0.4+25\times3.7+35\times0.1+40\times14.5}{60+1.8+0.4+0.4+3.7+0.1+14.5}}\approx 3.0$$

利用各平均难度系数，构造 5 个相连接的分别包含其相应平均难度系数的不相同的 5 个区间，从而建立起难度衡量标准：

(1)容易：$D\in[1,1.05)$。一个难度 $D=1.0$ 的数独题目，使用了 37 次规则 1。

(2)较容易：$D\in[1.05,4.5)$。一个难度 $D=1.2$ 的数独题目，使用了 54 次规则 1，3 次规则 3。

(3)适中：$D\in[4.5,6)$。一个难度 $D=4.8$ 的数独题目，使用了 55 次规则 1，3 次规则 3，9 次规则 6 和 73 次规则 9。

(4)困难：$D\in[6,13)$。一个难度 $D=10.7$ 的数独题目，使用了 62 次规则 1，26 次规则 6，88 次规则 9，2 次规则 14。

(5)非常困难：$D\geqslant 13$ 或者用完所有规则无法求解。一个难度 $D=14.2$ 的数独题目，使用了 53 次规则 1，7 次规则 3，18 次规则 6，1 次规则 8，94 次规则 9，1 次规则 10，2 次规则 11，3 次规则 12，1 次规则 13 和 2 次规则 14。

这样的划分标准使得 KLSuduku 软件各难度等级中尽可能多(75%以上)的数独题目落到了相应难度级别中。

算法伪代码如下：

1. 输入数独题目

2. 用规则 1 求解数独，若成功则计算难度系数 D 并输出结果，否则转 3

3. for(i=2; i<=14; i++)

4. 用规则 i 求解数独，判断是否起作用，若起作用让该规则 i 的使用次数 n_i 加 1，然后转 2

5. end for

6. 输出结果并计算难度系数 D

2.3　结果分析

从数独博士软件的 5 个级别中分别随机抽取了 100 道题目，然后用难度衡量标准将其重新分级，结果如表 2 所示。

表 2　数独博士各等级题目的重新分级信息

难度等级	博士入门等级	博士初级	博士中级	博士高级	博士骨灰级
容易	98	0	0	0	0
较容易	1	84	38	0	0
适中	1	16	57	7	0
困难	0	1	2	59	46
非常困难	0	0	3	34	54

对该列联表做卡方独立性检[6]。计算卡方值：

$$K=\sum_{i=1}^{5}\sum_{j=1}^{5}\frac{(n_{ij}-N\times n_{i\cdot}\times n_{\cdot j})^2}{N\times n_{i\cdot}\times n_{\cdot j}}\approx 1039.42$$

其中，N 为样本总频数；$n_{i\cdot}$ 为列联表第 i 行总频数；$n_{\cdot j}$ 为列联表第 j 列总频数；n_{ij} 为列联表中 i 行 j 列的实际频数。

卡方分布的自由度 df=(5－1)×(5－1)=16。查卡方临界值表知道自由度为 16 且显著性水平 a=0.01 时的卡方值为 32.00。由于 1 039.40 远远大于 32.00，因此有 99%的把握认为难度等级划分标准与数独博士等级划分标准是相关的。

进一步计算得到 Goodman-Kruskal 相关系数：

$$r=\sqrt{\frac{kf}{kf+N}}=\sqrt{\frac{1039.40}{1039.40+500.00}}\approx 0.82$$

由于 r 已经很大，因此有理由认为难度衡量标准与数独博士的难度衡量标准具有较强的相关性，进而证实了难度衡量标准的合理性。

3　数独题目的随机生成算法

3.1　初始数独棋盘的产生

理论上，可以从空的棋盘开始遍历每一种可能的情况，产生初始数独棋盘[7]。但是总的完整(所有空格都已经填写合法数字的)数独大概有 6.671×10^{21} 个[8]，所以这样做相当费时。本文采用如下算法产生初始棋盘：从空的棋盘开始，随机把 1~9 的数字填写到空棋盘从左上方起点开始的斜对角线上的 3 个区块中，直到这 3 个区块数字都已填满且无矛盾。然后，用递归算法求解该区块随机填满数字的数独，并且它的解可能不止一个(假设 n 个)。最后，在这些解中随机选取一个作为初始数独棋盘。

3.2　随机生成算法

从 2.1 节中产生的初始数独棋盘开始，随机地抽取该棋盘一个格子号将该格子的数字删掉，然后用递归求解算法判断当前数独棋盘是否有唯一解。如果唯一则重复进行下一次操作；若不唯一则恢复该格子的数字，然后重复进行下一次操作。一直进行上面的操作直到剩下没有删除的格子足够少(一般少于 30)。

3.3　随机生成算法的伪代码

随机生成算法的伪代码如下：

(1)导入空棋盘。

(2)对空棋盘斜对角线 3 区块随机填数。

(3)用递归算法求解该 3 区块已填数的数独棋盘的解(n 个解)，随机选取一个解作为生成数独的初始棋盘。

(4)重复步骤(5)和步骤(6)的操作直到没有删除的格子数足够少。

(5)随机抽取该初始棋盘一个其位置上的值不为 0 的格子号，删掉该位置上的数字。

(6)用递归法判断当前棋盘是否有唯一解，若有唯一解则转步骤(5)。

若没有唯一解则恢复该格子的原来数字，转步骤(5)。

(7)输出最终生成的数独题目。

3.4　结果分析

按照上述算法思想，用 C 语言编写程序实现了 2.3 节中的算法，结果表明该套算法是可行的。表 3 是随机生成的 100 道数独题目中的难度级别，及其相应难度级别对应的题目数量结果。这 100 道数独题目的平均生成时间为 700 ms。

表 3　各难度等级数独题目的数量分布信息

题目难度	题目数量
容易	45
较容易	8
适中	6
困难	22
非常困难	19

从表 3 可以看到，用生成算法随机产生的 100 道数独题目中，容易题目数 45，较容易题目数 8，适中题目数 6，困难题目数 22，非常困难题目数 19。从各难度等级来看，该 100 道题目的难度分布中，容易、困难和非常困难的题目数偏多，而较容易和适中的题目数偏少。这是由于算法的随机性造成的，为此可以生成大量数独题目(例如 10 000 个)，然后将其按不同难度等级分类，这样可以使得产生的数独题目不同难度级别分布更均匀些。

4　结束语

本文首先提出了模仿人工智能求解数独题目的一系列规则，并给出了建立在该系列规则上的人工求解算法及建立在加权基础上的难度衡量标准。然后，给出了随机生成数独的算法。最后通过 C 语言程序实现了这一整套算法，实验结果表明，该套算法能够合理地评价题目难度，以及在可接受的时间内随机产生大量(理论上 $6.67\times10^6\times n$，其中 n 是 2.2 节算法中多解的数目)的 5 种不同难度级别的数独谜题，从而验证了该套算法的可行性。然而该算法也有不足之处，例如：难

(下转第 167 页)