数独中的数学应用*

王琢†
指导教师：白峰杉教授

摘要

本文以数独为研究对象。数独是一种依据一定规则在九宫格内填写数字的游戏，文中介绍了其背景和近期的研究进展。围绕数独问题，本文提出了几个新问题并提出了一种解决的办法。

本文首先分析了求解数独常用的逻辑算法和机器算法，加以分类整理，并为每一类算法编程，制作了一个简易的数独逻辑求解程序。再利用随机过程方面的知识，对玩家求解数独的过程进行了建模。本文给出的办法，使得我们能够检测玩家求解数独是否异常，评估玩家对于每一类逻辑的熟练程度，并且能够给出一种与玩家水平相关的难度评估办法。

关键词： 数独 逻辑解法 机器解法 异常检测 水平评估 难度评估

主要符号对照表

G	格子(Grid)，数独中最小的单元
G_{mn}	第m行，第n列的格子，也记做$RmCn$
$r(G)$	G所在的行数
$c(G)$	G所在的列数
$b(G)$	G所在的宫数，从左至右，从上至下标记为1-9
$v(G)$	G中所填的数字
R	行(Row)，由横向的一行9个格子组成
R_n	$R_n = \{G\|r(G) = n\}$
C	列(Column)，由纵向的一行9个格子组成

*编者注：收入本刊时删节了原文第二章“数独逻辑解法”.

†基数51

C_n	$C_n = \{G\|c(G) = n\}$
B	宫(box)，由相邻的3×3的9个格子组成
B_n	$B_n = \{G\|b(G) = n\}$
U	泛指某个单元(unit)，可以是R，C或者B之一
N	$N = 1, 2, \ldots, 9$
x, y, z	用来代替某个特定的数字，$x \in \mathbb{N}$
$Cand(G)$	G中当前所有合理候选数(Candidates)组成的集合，$Cand(G) \subseteq \mathbb{N}$
S	$S = G_1, G_2, \ldots, G_k$
$Cand(S)$	$Cand(S) = \bigcup_{G \in S} Cand(G)$
X_k	表示不同的逻辑种类
$D(X_k)$	逻辑X_k的难度，具体函数形式需要设定
T_{X_k}	随机变量，完成逻辑X_k所需要的时间
S_n	表示数独题的局面，可以是未填过的状态，也可以是填完之后的状态，或者填写过程中的任一中间状态
$D(S_n)$	局面S_n的难度，具体函数形式需要设定
T_{S_n}	随机变量，在局面S_n停留的时间

第一章　引言

1.1　数独背景

数独起源于18世纪末的瑞士，最早的雏形由欧拉提出。此后数独在美国发展、又在日本得以发扬光大，因此它的国际通用名称是Sudoku，也就是日文数独的意思。我们最常见的数独盘面是个九宫格，每一宫又分为九个小格。在这81格中给出一定的已知数字和解题条件，主要利用逻辑推理，辅以少量的穷举法，在空格中填入数字1-9。使1-9每个数字在每一行、每一列和每一宫中都只出现一次。

这种游戏全面考验做题者的观察能力和推理能力，被许多人认为是训练头脑、锻炼逻辑能力的绝佳方式。其数字排列方式和解题方法却变化繁复：

2005年，Felgenhauer和Jarvis在[2]中给出数独的总个数为：

$$6,670,903,752,021,072,936,960. \tag{1.1}$$

如果不计重复（例如置换数字，几何对称），那么 [4]声称还剩下的数独个数为：

$$5,472,730,538 \tag{1.2}$$

关于置换数字，几何对称等，在下文的”数独命题”部分(本章第1.5小节)还会有所提及。

1.2 数独定义

定义1. n阶数独题是指一个$n^2 \times n^2$的方形格子S，由n^2个$n \times n$的子方格拼成。在$n^2 \times n^2$个格子中，一些已经预先填好了$1 \sim n$的数字，其他留白。

当$n = 3$时，就是我们通常见到的9×9的数独。对于其他n的情形，仅有$n = 2$或$n = 4$时的迷你数独和大数独题目已在市面上出现，更高阶的数独由于难度过高，所以受玩家关注极少。

定义2. n阶数独题的解是指将$n^2 \times n^2$的方形格子S全部用$1 \sim n^2$填满，对于给定数值的格子，不能改变。剩余的格子必须满足：每行、每列、以及每个$n \times n$的子方格均由$1 \sim n^2$填满，数字不得缺少或重复。

另外还有一个附加性质，对于广义的数独不必满足。但是我们通常遇到的数独题目都被预先假定满足解的唯一性：

定义3. 标准n阶数独题是指其解存在且唯一的n阶数独题。

1.3 数独题求解的计算复杂度

要求解n阶数独，或者判断一个n阶数独的解是否唯一，是一个NP-Complete的问题。这在T.Yato和T.Seta的文章中已经被证明[5]。I.Lynce等在2006年，证明了n阶数独题可以在多项式时间内转化为一个确定规模的适定性问题(Satisfiability Problem, SAT)[3]。然而SAT问题已经被证明是一个NP-Complete的问题，所以n阶数独题也是一个NP-Complete的问题。

此外，n阶数独也可以被转化为精确覆盖问题(Exact Cover Problem, ECP)，和整数线性规划问题(Integral Linear Programming, ILP)。这两种转化方式得到的结论与SAT问题一致，n阶数独是一个NP-Complete的问题。

1.4 数独题解法简介

从现在起，我们将讨论的范围限制在3阶数独，也就是我们通常所说的数独上。对于3阶数独来说，计算机采用暴力破解的办法求解也不过是一瞬间的事情。最简单的一种机器求解办法如下，也即搜索算法：

首先按数独规则整理出每个格子中当前可以用来候选的数字，即填入这个数字不破坏数独题目所要求的定义2。然后逐一尝试每个可能的取值，直到填满所有的格子或者发现矛盾之处（即某个格子不存在任何候选值），则回溯到上一假设并删除相应的候选数值。

对于机器来说，用不同种类的回溯算法解决一个3阶数独一般只需要几个毫秒。但是对于玩家来说，回溯却是最后的选择。虽然对于某些很难的题目来说，无论人或计算机都无法回避这种办法。但是对于绝大多数的问题，这种方法却并不是解题所必须的。人们解题的目的并不是仅仅想知道答案，解题的过程才是最大的挑战与享受。

1.5 数独命题

通常数独命题有两大类方法。一类是基于已有的题库，对已经给出的数独问题进行可能的数字置换，镜像、旋转等来变成新的题目[4]。数独上能够进行的变换主要有以下几种：

1. 对数字1 ~ 9作置换。
2. 以三行或三列为单位置换（比如将123列的位置与456列对换，但不能将234列与567列对换）。
3. 在三行或三列内置换（比如对换1列与2列，但不能对换3列与4列）。
4. 对整体做镜像或旋转。

另一类是不基于已有题库的生成方式。其一可以通过往空白的格子里逐个填上一些数字，直到它有唯一解。其二可以从一个已经填好的数独中逐个减少已知数的个数，并保持解的唯一性，直到数字不能再减少或者难度已经达到要求。

1.6 题目难度评估与玩家水平评估

对数独进行难度评估是一项复杂而又艰巨的任务。对计算机而言，数独题目的难度完全可以用解题时间来衡量。但是这种衡量方式并不适用于玩家。正如前面所述，玩家求解数独时更多的是在使用逻辑推理，而不是枚举和回溯。所以评价一个数独的难度应当考虑的是推理的难度和数量，以及推理突破点的寻找的困难程度。

具体一道题目的难度也在一定程度上是因人而异的，对于推理能力相似，但是思维习惯不同的两个人来说，解决同一道题目所花的时间也是不一样的。本文希望从这方面加以突破，得出一种更深入的分析方式，来评估玩家对不同种类逻辑推理的熟练程度。这一方

面也给出了某个玩家在整体上的水平。另一方面，如果得到一些人的平均熟练程度，就可以客观地评价一道题目的难度。

第二章　解题过程的建模

在本章中，我们将对玩家求解数独的过程进行建模。本章将不同的逻辑方式表示为$X_i(i = 1, 2, \ldots, n, \ldots)$。

每种逻辑方式X_i有一个难易程度$D(X_i)$，相互之间有严格的序关系。$D(X_1) < D(X_2)$即表示X_1比X_2容易。但是请注意，这个序关系是因人而异的，也就是说这个序关系是在某一个总体下而言的。这个总体可以是一名玩家，也可以是某一类玩家，也可以是全部玩家的集合。此外，记T_{X_i}为完成逻辑X_i所花费的时间，它是一个取正值的随机变量。

不同的局面被记做S_n，从刚拿到手的数独题目，或者填到一半的数独题，或者全部解出来的数独题，都是不同的局面。如果在某个局面S_1下，可以使用的逻辑多于并且简单于局面S_2，那么我们说局面S_1的难度$D(S_1)$要小于S_2的难度$D(S_2)$。

2.1　背景与基本假设

基于玩家解题时消耗时间的数据，我们有以下几个事实：

a. 如果$D(X_1) < D(X_2)$，那么期望值$E(T_{X_1}) < E(T_{X_2})$。

b. 如果$D(S_1) < D(S_2)$，那么期望值$E(T_{S_1}) < E(T_{S_2})$。

考虑到同一种逻辑X_i的应用范围只是局部的，那么在某个局面下，就可能有多个可以应用X_i的地方。而它们可能都可以推算出某个格子的取值。

c. 如果G_{mn}可以由a个X_1中任意一个推出结果，而G_{pq}可以由b个X_1中任意一个推出结果，且$a > b$，那么下一步推算出G_{mn}的概率要大于下一步推算出G_{pq}。

d. 推算出全部格子所需的时间如图2.1和图2.2分布[1]，且根据题目难度[1]的不同有不同的期望值。由图中可以看出，这两个时间大概是某种指数族的分布。

根据上述事实，我们可以假设T_{X_k}满足指数分布，其参数为λ_{X_k}。

$$T_{X_k} \sim \exp(\lambda_{X_k}) \tag{2.1}$$

而λ_{X_k}的倒数则可以用作衡量逻辑X_k难度的方式：

[1]此处的题目难度是[1]自行分类的结果，分类方法不详

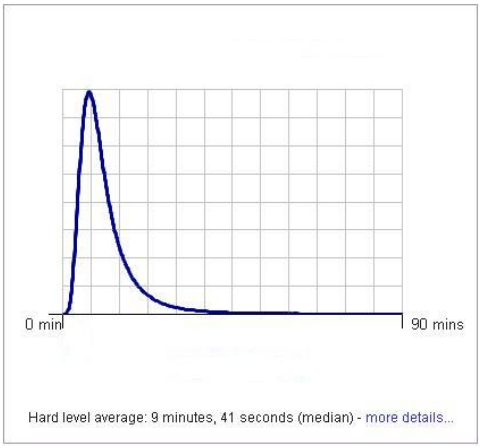

图 2.1: 困难等级题目完成所需时间

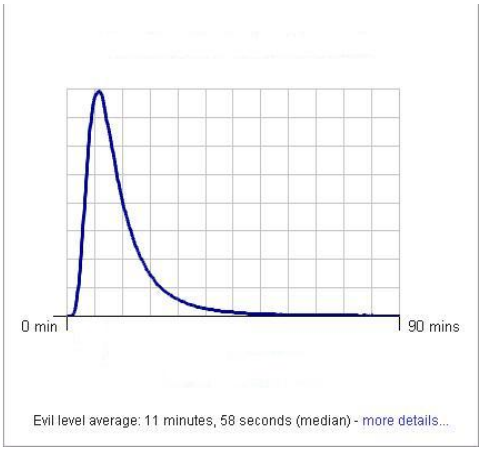

图 2.2: 特难等级题目完成所需时间

$$D(X_k) = \frac{1}{\lambda_{X_k}} = E(T_{X_k}) \tag{2.2}$$

自此，我们假定D是一个难度函数，其量纲为时间，用来表示完成某一步骤、某一逻辑或者完成某一道题目所需要的平均时间。

2.2 单一局面下的模型

2.2.1 逻辑路径L

假设当前局面为S_0，那么我们可以用程序计算出当前所有可能的逻辑方法，以及这些逻辑的推论：下一个可以填入的数字。在填入这些数字后，局面成为S_n。

记L_m为所有可能的逻辑“路径”。不同的L可能使用同样的逻辑X_k，例如图2.3：L_1, L_5, L_6都使用逻辑X_1；不同的L也可能导出同样的结果，例如L_1, L_2都能推出S_1所代表的格子的数值。

2.2.2 逻辑路径L_m的先验概率$P(L = L_m)$

假设我们知道每种逻辑X_k的难度参数λ_{X_k}，那么每一步逻辑推断X_k所需要的时间为：

$$T_{X_k} \sim Exp(\lambda_{X_k}) \tag{2.3}$$

所以不同逻辑路径需要的时间T_{L_m}为：

$$T_{L_m} = T_{X(L_m)} \tag{2.4}$$

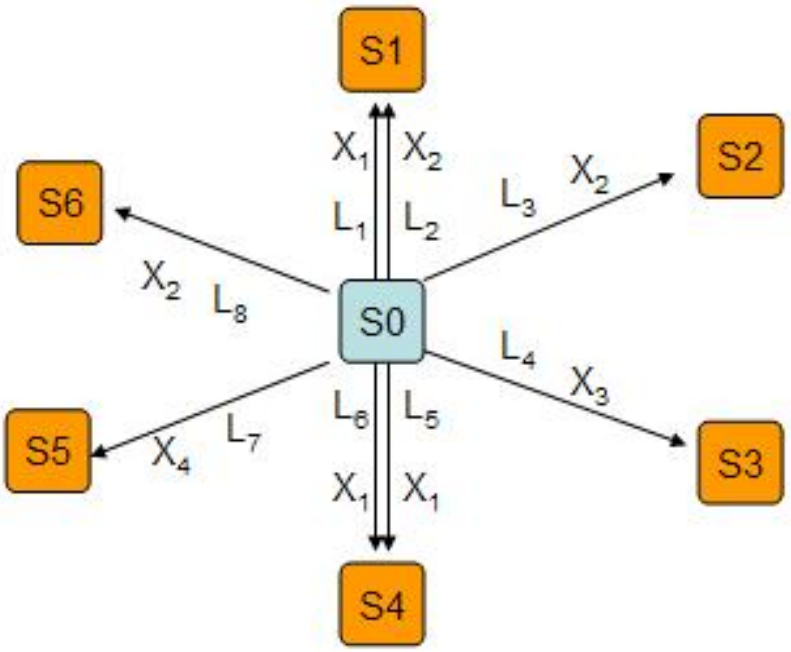

图 2.3: 单一局面

由于T_{L_m}是指数分布的随机变量(见式2.1)，那么L_i在所有L_m中最先发生的概率为：

$$P(L = L_i) = \frac{\lambda_{X(L_i)}}{\sum_{m=1}^{M} \lambda_{X(L_m)}} \tag{2.5}$$

2.2.3 逻辑路径L_m在时间$T = t_0$时的后验概率$P(L = L_m|T = t_0)$

如果已知在当前状态停留的时间$T = t_0$，那么不同逻辑路径发生的概率则会有所变化。由Bayesian公式：

$$\begin{aligned} P(L = L_i|T = t_0) &= \frac{P(L = L_i)P(T = t_0|L = L_i)}{\sum_{m=1}^{M} P(L = L_m)P(T = t_0|L = L_m)} \\ &= \frac{P(L = L_i)P(T_{L_i} = t_0)}{\sum_{m=1}^{M} P(L = L_m)P(T_{L_m} = t_0)} \\ &= \frac{P(L = L_i)f_{L_i}(t_0)}{\sum_{m=1}^{M} P(L = L_m)f_{L_k}(t_0)} \end{aligned} \tag{2.6}$$

式中$P(L = L_i)$即为第2.2.2小节所计算的先验概率(式2.5)。$f_{L_i}(t)$是随机变量T_{L_i}的概率密度函数(Probability Density Function, PDF)。

在计算出这个后验概率之后，就可以计算在$S = S_0$状态下，经过时间$T = t_0$后，填出

格子S_n的概率：

$$P(S = S_n|T = t_0) = \sum_{L_m \text{leads to} S_n} P(L = L_m|T = t_0) \tag{2.7}$$

2.3 完整的数独解题模型

在讨论完单一局面下的解题模型之后，我们可以开始研究整体求解数独的模型。求解数独的过程就是不断处理单一的状态。每填入一个数字之后即从原先的状态转移到一个新状态，如图2.4所示

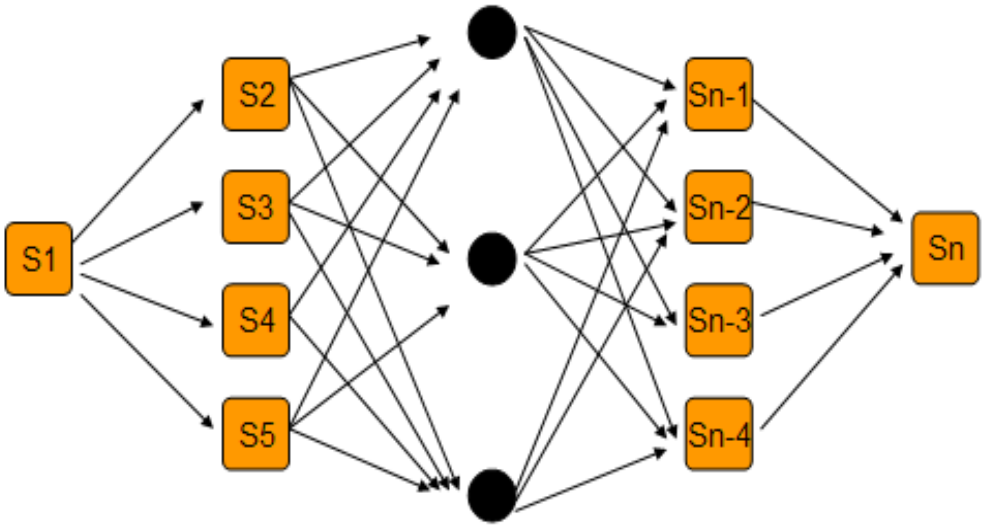

图 2.4: 完整的解题过程

但是对于一个数独题目来说，存在的状态实在太多，比如一道数独题，有50个空格，那么所有可能的状态就有2^{50}个。要求遍历每个状态的运算进行起来将非常困难，我们采用两种方法来利用这个模型。

2.3.1 单链模型

既然无法对全局进行计算，那么计算玩家求解数独的过程，只需要计算大约50个状态的详细数据，也就是图2.4中从S_1到S_n任意的一条链。用这种方法可以进行求解数独过程的异常检测，而且可以实时进行。

每一步填入某个格子的概率可以被计算出来。为了比较的需要，这个概率的绝对数值没有太大的意义，考虑它的似然比对于我们的检验更加方便。详细过程可以参见3.1。

2.3.2 近似模型

无法承受过于庞大的运算，可以在有条件近似的前提下大幅削减计算量。比如在评估题目难度的时候，可以用近似的办法计算出题目难度的一个下界。

题目的难度应该为每条链难度$D(Chain)$的加权平均值：

$$D(S_0) = \sum_{Chain} P(Chain)D(Chain) \tag{2.8}$$

但是每条链必然包含每个空格一次，所以难度还有另外一种求和方法：

$$\begin{aligned} D(S_0) &= \sum_{G} D(G) \\ &= \sum_{G}(\sum_{k} P(X_k|G)D(X_k)) \end{aligned} \tag{2.9}$$

于是可以将最简单的逻辑X_1 [2] 能推断出来的空格G直接先填妥，其他的部分再用其他逻辑方法遍历来确定。如果在S_0状态能用X_1填出n个空格，并设填好后的中间状态为M，其难度为$D(S_M)$

$$D(S_0) \geq nD(X_1) + D(S_M) \tag{2.10}$$

再重复对S_M状态做类似分析，即可估算出数独题目难度的一个下界。具体的算例请参见第3.3节。

2.3.3 参数λ_{X_k}

第2.2节我们讨论了在已知参数$\lambda_{X_k}, (k = 1, 2, \ldots)$以及停留时间$T = t_0$的情况下如何计算填入每一个格子的先验概率(式2.5)和后验概率(式2.6)。下面我们将要推导如何根据已有样本来确定这组衡量了逻辑难度的参数λ_{X_k}。

我们能够利用的每个样本是这样的，给定当前状态S_j，同时知道玩家下一步填写的格子G_j和所用时间t_j。首先我们可以分析出G_j 能用到的逻辑的种类和个数。例如图2.3中，填出S_1格子可以用1个X_1或1个X_2逻辑推断出来。

为了计算方便，我们将逻辑的种类限定为两种：X_A和X_B。它们需要的时间为T_A和T_B，分别服从参数为λ_A和λ_B的指数分布。对于逻辑种类更多的情形，我们一会将会看到其并不影响最终解的形式。

在S_j状态，填出G_j所需要的时间为T_j，可能的逻辑种类为a_j种X_A和b_j种X_B。那么我们希望选取最好的参数λ_A和λ_B来使得$\prod_j f_j(t_j)$取到极大值。

$$T_j \sim \exp(\lambda_j) \tag{2.11}$$

[2]此处最简单的逻辑可能因样本来自的群体而不同

其中：

$$\lambda_j = \frac{a_j\lambda_A + b_j\lambda_B}{a_j + b_j} \tag{2.12}$$

同时我们有：

$$\frac{\partial\lambda_j}{\partial\lambda_A} = \frac{a_j}{a_j + b_j} \tag{2.13}$$

$$\frac{\partial\lambda_j}{\partial\lambda_B} = \frac{b_j}{a_j + b_j} \tag{2.14}$$

下面我们计算概率密度函数$f_j(t_j)$及其对λ_j的偏导数：

$$f_j(t_j) = \lambda_j e^{-\lambda_j t_j} \tag{2.15}$$

$$\frac{\partial f_j(t_j)}{\partial\lambda_j} = (1 - \lambda_j t_j)e^{-\lambda_j t_j} \tag{2.16}$$

如此即可计算$f_j(t_j)$对λ_A和λ_B的偏导数：

$$\frac{\partial f_j(t_j)}{\partial\lambda_A} = \frac{\partial\lambda_j}{\partial\lambda_A}\frac{\partial f_j(t_j)}{\partial\lambda_j} = \frac{a_j}{a_j + b_j}(1 - \lambda_j t_j)e^{-\lambda_j t_j} \tag{2.17}$$

$$\frac{\partial f_j(t_j)}{\partial\lambda_B} = \frac{\partial\lambda_k}{\partial\lambda_B}\frac{\partial f_j(t_j)}{\partial\lambda_j} = \frac{b_j}{a_j + b_j}(1 - \lambda_j t_j)e^{-\lambda_j t_j} \tag{2.18}$$

需要假设$f_j(t_j) \neq 0$对于任意的j。那么$\max\prod_j f_j(t_j)$就等价于：

$$\frac{\partial}{\partial\lambda_A}\prod_j f_j(t_j) = 0 \tag{2.19}$$

$$\frac{\partial}{\partial\lambda_B}\prod_j f_j(t_j) = 0 \tag{2.20}$$

由求导的链法则，在两边同时除以$\prod_j f_j(t_j)$，就得到：

$$\sum_j \frac{\frac{\partial}{\partial\lambda_A}f_j(t_j)}{f_j(t_j)} = 0 \tag{2.21}$$

$$\sum_j \frac{\frac{\partial}{\partial\lambda_B}f_j(t_j)}{f_j(t_j)} = 0 \tag{2.22}$$

这等价于

$$\sum_j \frac{a_j}{a_j + b_j}(\frac{1}{\lambda_j} - t_j) = 0 \tag{2.23}$$

$$\sum_j \frac{b_j}{a_j + b_j}(\frac{1}{\lambda_j} - t_j) = 0 \tag{2.24}$$

这个方程的形式无关于逻辑的种类是A, B，还是$X_1, \ldots, X_k$。计入多少种逻辑就可以得到多少个未知数和方程。用数值解法求解这个方程组就可以得到参数λ_A和λ_B的最优估计。

具体的计算样例请参见第3.2节。

第三章　模型的应用

3.1　解题过程的异常检验

这个模型最简单的应用就是针对玩家解题过程的异常检测。玩家在解数独题过程中，通常都是采用符合逻辑的方式。那么只要鉴定出玩家的每一步是否有背后的逻辑作支撑，就可以从某种程度上检测出异常。

玩家填写每一步所消耗的时间也是一个重要的因素。如果玩家用较短的时间填写了一个需要逻辑难度较高的格子，或者用较长的时间填写了一个非常容易就能推算出的格子，在某种程度上来说都是异常的表现。

3.1.1　单步过程的异常检验

如果要分析完整的路线是否异常，首先需要分析单步的过程是否存在异常。我们任意选取一个数独题，如图3.1所示。我们直接分析第一步在什么时间，填在何处最为合理。

首先我们将计算机搜索的逻辑范围限定在Singles[1]，即不考虑其他可能的逻辑方式。并且假定我们已知参数：

$$\lambda_A = \frac{1}{15}\mathrm{s}^{-1} \tag{3.1}$$

$$\lambda_B = \frac{1}{10}\mathrm{s}^{-1} \tag{3.2}$$

我们用程序分析图3.1中的这个局面，得到了所有可以使用的逻辑，如下表所示：

[1]编者注: Singles及下文的Hidden Single, Naked Single, Hidden Pair为原文第二章“数独逻辑解法”中介绍的逻辑解法

Number	Row	Column	Logic Type	Remark
1	2	6	Hidden Single	R
2	2	6	Hidden Single	B
3	3	4	Hidden Single	C
4	3	4	Hidden Single	B
5	4	6	Naked Single	
6	8	4	Naked Single	

表中的每一行代表了一条逻辑路径。例如第一行就表示：在R2C6格，有一个候选数在该行是唯一候选数。

现在我们一共得到了6条逻辑路径$L_1, \ldots, L_6$。如果未知时间T，那么我们可以计算L_i的先验概率$P(L_i) = \frac{\lambda_{L_i}}{\sum_{i=1}^{6} \lambda_{L_i}}$：

$$P(L_i) = 0.1875, i = 1, 2, 3, 4. \tag{3.3}$$

$$P(L_i) = 0.1250, i = 5, 6. \tag{3.4}$$

假设已知时间$T = t_0$，那么我们就可以计算L_i的后验概率。例如$t_0 = 10\mathtt{s}$，我们得到如下结果：

$$P(L_i|T = 10\mathtt{s}) = 0.1908, i = 1, 2, 3, 4. \tag{3.5}$$

$$P(L_i|T = 10\mathtt{s}) = 0.1184, i = 5, 6. \tag{3.6}$$

可以据此计算出填出每个格子的概率：

$$P(R2C6|T = 10\mathtt{s}) = P(L_1|T = 10\mathtt{s}) + P(L_2|T = 10\mathtt{s}) \tag{3.7}$$

$$P(R3C4|T = 10\mathtt{s}) = P(L_3|T = 10\mathtt{s}) + P(L_4|T = 10\mathtt{s}) \tag{3.8}$$

$$P(R4C6|T = 10\mathtt{s}) = P(L_5|T = 10\mathtt{s}) \tag{3.9}$$

$$P(R8C4|T = 10\mathtt{s}) = P(L_6|T = 10\mathtt{s}) \tag{3.10}$$

数值上：

$$P(R2C6|T = 10\mathtt{s}) = P(R3C4|T = 10\mathtt{s}) = 0.3816 \tag{3.11}$$

$$P(R4C6|T = 10\mathtt{s}) = P(R8C4|T = 10\mathtt{s}) = 0.1184 \tag{3.12}$$

绘制成图的结果参见图3.2。

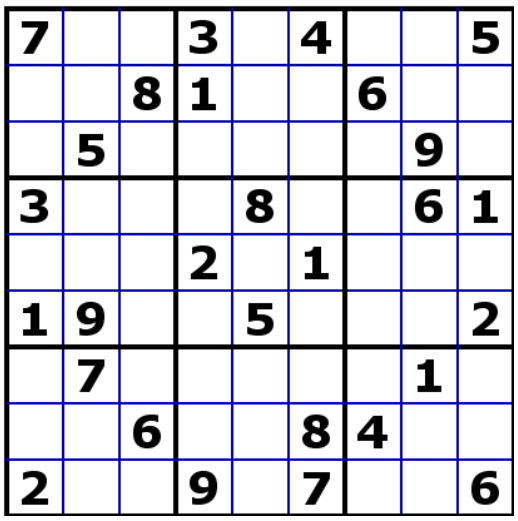

图 3.1: 例题1

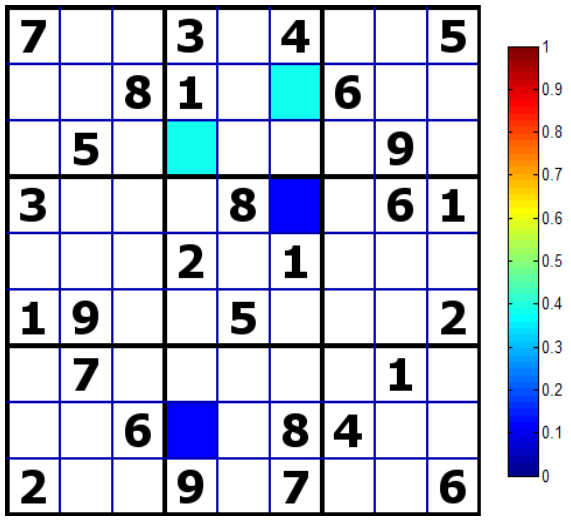

图 3.2: 填出每个格子的概率$t_0 = 10\text{s}$

类似地，我们还可以计算出任意时间下，填出这几个格子的概率。以$T = 30\text{s}$和$T = 120\text{s}$为例：

$$P(R2C6|T = 30\text{s}) = P(R3C4|T = 30\text{s}) = 0.3117 \tag{3.13}$$

$$P(R4C6|T = 30\text{s}) = P(R8C4|T = 30\text{s}) = 0.1883 \tag{3.14}$$

$$P(R2C6|T = 120\text{s}) = P(R3C4|T = 120\text{s}) = 0.0381 \tag{3.15}$$

$$P(R4C6|T = 120\text{s}) = P(R8C4|T = 120\text{s}) = 0.4619 \tag{3.16}$$

绘制成图的结果参见图3.3和3.4

3.1.2 完整过程的异常检验

我们选取一个难度为简单的数独题，如图3.5所示。请一位玩家现场做题，我们用程序记录下玩家每一次正确填入的格子，和所花掉的时间。对于那些错误的填入，不计入考虑。同时将计算机搜索的逻辑范围限定在Singles。

如此我们得到如附录A所示的一个记录文件。

对于每个单步，我们计算第2.2.3节所分析的后验概率$P(L = L_i|T = t_0)$，并计算在t_0时刻填出每个格子的概率：$P(S = S_n|T = t_0)$，和填出每个格子的似然比(Likelyhood Ratio)$LR(S = S_n|T = t_0)$。其中:

$$LR(S = S_n|T = t_0) = \frac{P(S = S_n|T = t_0)}{\max_n P(S = S_n|T = t_0)} \tag{3.17}$$

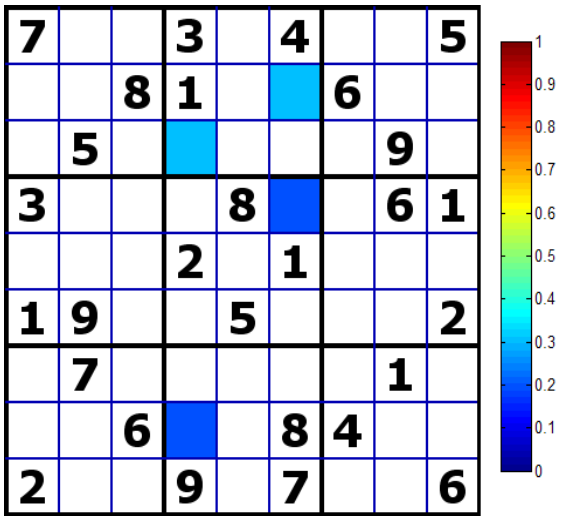

图 3.3: 填出每个格子的概率$t_0 = 30$s

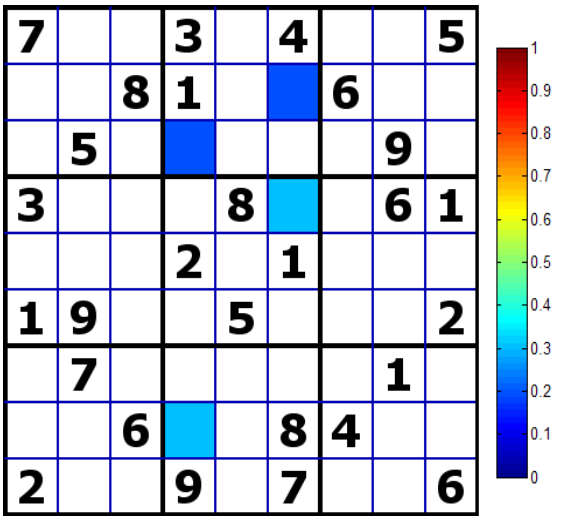

图 3.4: 填出每个格子的概率$t_0 = 120$s

5		3				7		
			7	9			3	
7			8			2		6
	1	6	9		7			
	4						9	
			6		3	8	1	
1		7			9			4
	3			6	4			
		9				5		8

图 3.5: 例题2

以第一步为例，得到如图3.6和图3.7所示的结果：

核对附录A中的记录，第一步填入的格子为R3C5，于是第一步的似然比为：

$$LR_1 = 1 \tag{3.18}$$

类似地，我们可以求出每一步的似然比LR_n，于是我们可以绘制出LR_n关于n的图像，如图3.8所示。几乎全部的步骤都有较大的似然比，所以这是一个相对正常的解题过程。在第42步出现了$LR_{42} = 0$，这种情况出现只有两个原因，一是由于玩家采用了没有被电脑计

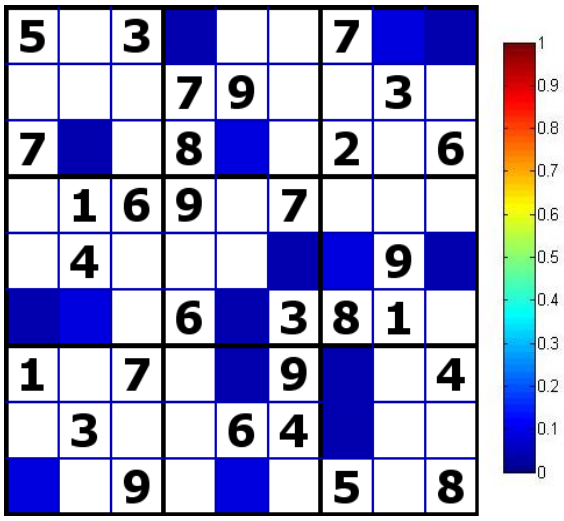

图 3.6: 填出每个格子的概率

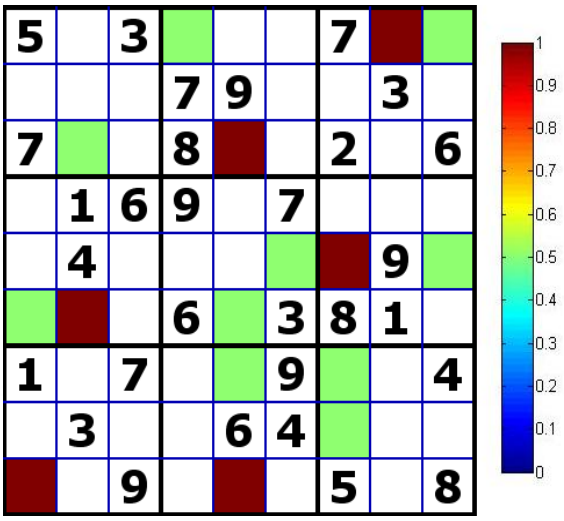

图 3.7: 填出每个格子的似然比

入搜索范围的逻辑方法，另外一个就是确实发生了异常，例如玩家偷看答案等。

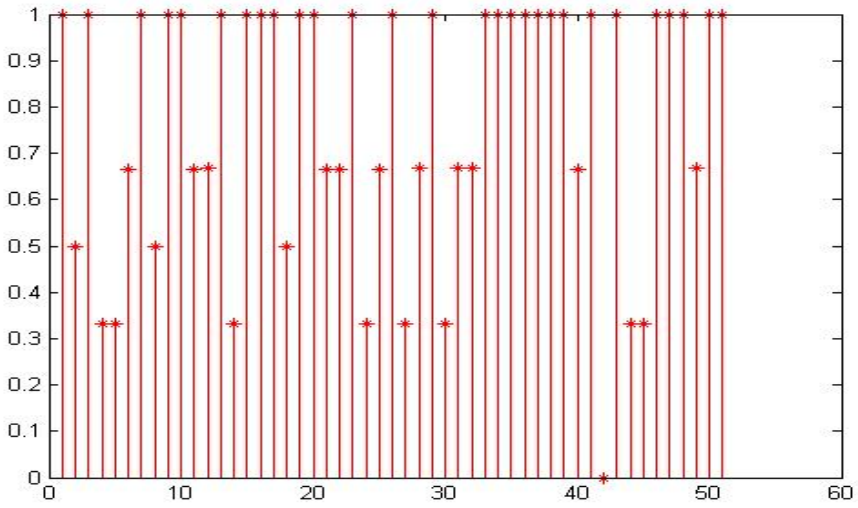

图 3.8: 正常过程对应的LR图

对于非正常的过程，我们也选取了两个样本，一个是附录A记录文件打乱次序，另外一个是请人隔一定的时间按照答案往上随机抄写一个数字。两种方法得到的似然比结果如图3.9和图3.10所示。

由图3.9和图3.10可以看出，在解题的早期，出现了较多不合逻辑的行为。因此很容易

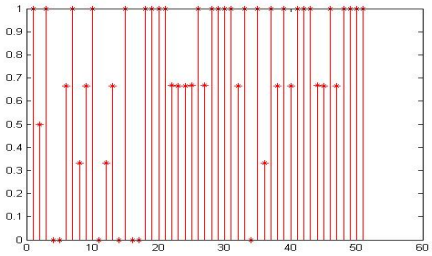

图 3.9: 打乱次序对应的LR图

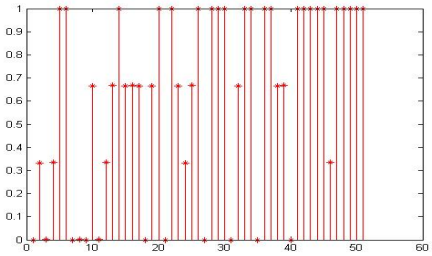

图 3.10: 抄写答案对应的LR图

判断出这两个都是异常的进度。

这种方法可以在一定程度上检测出解题过程的异常，但是仍然有它的缺陷，因为许多能通过这个检验的进程其实也可能存在不同程度的异常。例如图3.10中，对于后期的步骤，无法检测出这样的异常，只要抄写答案的次序合理，也会被这种检验方式判定为正常过程。

3.2 逻辑难度评估

由第2.3.3节提供的方法，我们可以计算出每种逻辑对于一个玩家或者一类玩家的难易程度。以下我们仍然将逻辑范围限定在Singles，利用与附录A中同时分析获得的数据来估算Naked Single和Hidden Single两种逻辑所需要的时间。

需要注意的是，评估逻辑水平的数据不一定要来自同一道数独题，也不一定来自同一名玩家。如果所有数据来自同一名玩家的解题过程，那么这个分析得到的结果就是各种逻辑对该玩家而言的难易程度。不同玩家的数据可能得到完全不同的结果。而如果数据来自全体玩家，那么这个分析得到的结果就是针对全体玩家而言。某种程度上说，这个结果应当就是这种逻辑的真实难度。

我们以两种逻辑对应的参数λ_A和λ_B为变量，代入方程2.23 以及2.24。得到如下结果：

$$r_1(\lambda_A, \lambda_B) = \sum_j \frac{a_j}{a_j + b_j}(\frac{1}{\lambda_j} - t_j) \tag{3.19}$$

$$r_2(\lambda_A, \lambda_B) = \sum_j \frac{b_j}{a_j + b_j}(\frac{1}{\lambda_j} - t_j) \tag{3.20}$$

式子中，λ_j的定义为：

$$\lambda_j = \frac{a_j\lambda_A + b_j\lambda_B}{a_j + b_j} \tag{3.21}$$

为了使得$r_1(\lambda_A, \lambda_B)$和$r_2(\lambda_A, \lambda_B)$更接近0，我们设：

$$r(\lambda_A, \lambda_B) = r_1^2(\lambda_A, \lambda_B) + r_2^2(\lambda_A, \lambda_B) \tag{3.22}$$

并取合适的范围绘制图像，求$r(\lambda_A, \lambda_B)$的极小值点。我们选取合适的范围，按次序计算三次，所得图像见图3.11，3.12和3.13。

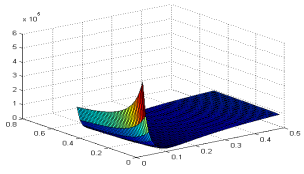

图 3.11: 第一次网格抽样

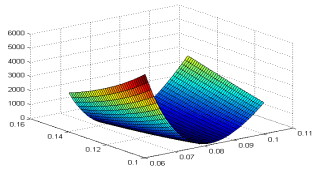

图 3.12: 第二次网格抽样

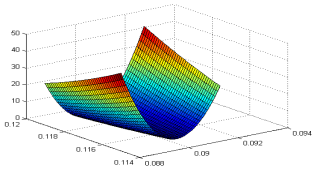

图 3.13: 第三次网格抽样

第一次抽样的范围我们依据数据定为0.04-0.50s，步长设置为0.01s。而三次抽样的得到的最优解和最优值分别为：

第一次	第二次	第三次
$\lambda_A = 0.08$	$\lambda_A = 0.090$	$\lambda_A = 0.0904$
$\lambda_B = 0.12$	$\lambda_B = 0.117$	$\lambda_B = 0.1169$
$r = 29.0853$	$r = 0.0553$	$r = 0.0056$

取最后一次得到的数值作为近似解。我们可以计算出Naked Single(A)和Naked Single(B)相对该玩家而言的难度：

$$D(A) = E(A) = \frac{1}{\lambda_A} = 11.0619\mathtt{s} \tag{3.23}$$

$$D(B) = E(B) = \frac{1}{\lambda_B} = 8.5543\mathtt{s} \tag{3.24}$$

对于多于2种逻辑的情况，仍然可以使用近似数值求解的办法，找到$r(\lambda_{X_1}, \ldots, \lambda_{X_k})$的极小值点。不过无法像本例一样绘制出曲面的三维图像。

3.3 题目难度评估

作者编写了一个小程序，用来分析最简单的解题路线需要多少种、什么类型的逻辑方式。同时考虑上述3.2节的分析，再进行题目难度评估就并不算困难了。我们仍然以图3.5中的例题来举例。经过分析，这个题目可以全部用Hidden Single这种逻辑方式按一定先后顺序完成。因此，这道题目难度的下界就是(式2.10)：

$$D \geq 51D(\texttt{Hidden Single}) = 476\mathtt{s} \tag{3.25}$$

对于其他情况，问题并没有如此简单。例如图3.14中所示的数独题目，其难度的下界是：

$$D \geq 50D(\texttt{Hidden Single}) + D(\texttt{Hidden Pair}) \tag{3.26}$$

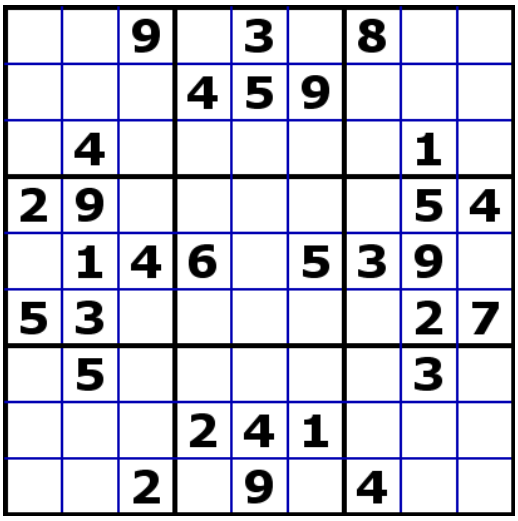

图 3.14: 例题3

第四章　结论

在这篇文章中，我们以数独这种数字游戏为背景，对玩家解题的过程进行了建模。按照本文提出的模型，我们可以达到以下几个目的：

a. 分析玩家填写过程，侦测异常过程。

b. 依据玩家填写过程的数据，评估不同逻辑的难度。

c. 能够因人而异的评估题目的难度。

其中a和b在之前的研究中没有被成体系地建立过，是本论文的创新之处。同时本论文还有许多未尽与不足之处，有待进一步研究与改进：

在侦测异常过程时，我们提出的办法把玩家填写所消耗的时间作为已知量来分析。然而每一步需要的时间本身就是可以作为异常检测的根据。所以可以考虑将时间因素一并加以分析。

在评估题目难度的过程中，由于硬件条件的限制无法综合考虑到所有可能的路线，所以这套近似的评估体系还并不算成熟。在使用效果上也不如其他分类方法，例如直接按某一道题目的平均解题时间来分类，或者许多现有的数独题目生成器，依据特有的生成方式可以自动产生特定难度的题目。如果第3.3节中所使用的难度评估方式可以进一步改进，得到的结果应该能够更加理想。

本文所用全部程序与数据可以向作者来函索取：jswangzhuo@gmail.com

附录 A　记录文件LOG FILE

请注意，此表中的行数和列数为$0, 1, \ldots, 8$，分别对应正文中的$1, 2, \ldots, 9$。

Number	Row	Column	Time spent(sec)	Number	Row	Column	Time spent(sec)
1	2	4	5.883000	27	8	7	19.924000
2	6	6	9.765000	28	6	7	10.269000
3	8	3	9.062000	29	8	1	6.390000
4	0	3	11.195000	30	2	5	7.865000
5	5	4	18.620000	31	1	8	3.976000
6	8	0	8.249000	32	2	7	5.469000
7	4	6	17.978000	33	3	6	5.382000
8	1	0	8.398000	34	1	2	5.998000
9	0	5	6.864000	35	1	1	3.585000
10	5	1	16.413000	36	6	4	6.943000
11	4	8	3.931000	37	4	5	11.367000
12	8	4	6.380000	38	3	7	18.322000
13	7	7	3.458000	39	3	8	14.039000
14	0	8	17.711000	40	5	8	13.763000
15	7	6	4.599000	41	4	0	24.749000
16	2	1	13.131000	42	3	4	21.632000
17	5	0	7.100000	43	3	0	3.050000
18	7	8	13.804000	44	7	0	4.575000
19	1	6	3.237000	45	6	1	8.508000
20	2	2	4.618000	46	7	2	3.698000
21	0	4	3.841000	47	6	3	7.979000
22	8	5	6.674000	48	7	3	6.616000
23	4	3	6.808000	49	4	4	5.027000
24	1	5	10.387000	50	4	2	5.475000
25	0	1	1.991000	51	5	2	5.119000
26	0	7	3.660000				

参考文献

[1] http://www.websudoku.com, 2009.2.14.

[2] B. Felgenhauer and F. Jarvis. Mathematics of sudoku i. *Mathematical Spectrum*, 39(1):15–22, 2006.

[3] I. Lynce and J. Ouaknine. Sudoku as a SAT problem. In *9th Internatioanl Symposium on Artificial Intelligence and Mathematics*, 2006.

[4] E.D. Russell and F. Jarvis. Mathematics of sudoku ii. *Mathematical Spectrum*, 39(2):54–58, 2007.

[5] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 86(5):1052–1060, 2003.