基于深度优先搜索的 Kakuro 剪枝求解算法

摘要

Kakuro 问题是已被证明的经典 NP 完全问题，暴力求解线性方程组整数无重复解计算难度很大。本文通过更为巧妙的方法简化了解题难度，使机器快速求解成为可能。并且据此算法提出了一套解题复杂度的表示公式。

针对问题一：我们首先建立了一套求解算法。利用深度优先搜索进行前序遍历和利用数学逻辑知识进行合理剪枝简化计算过程。对于每一组连续待填写方格，提前找到所有可能的组合可以将局部计算复杂度由暴力搜索的 9^n 简化到平均 6-7 组，极大地简化搜索流程。

针对问题二：我们提出了根据空白格总个数与连续空白格组合的个数的比值来作为判定难度的标准，由于空白格的个数代表着未知数的个数，连续空白格组合的个数表示方程的个数，根据线性代收的有关知识，我们知道此比例越大表明难度越大，将此比例作为难度系数，以此作为难度评估的标准，并将难度系数在小于 1 的数独定义为简单，难度系数在 1。

针对问题三：基于第二问的结论，我们基于先保证有解再保证有唯一解的思路进行数独的生成。为了获得不同难度的数独，我们先确定空白格个数与斜线格的个数，确定他们的位置，然后在连续空白格中随机生成 1 到 9 的不重复的数字，这就保证了数独一定有解。然后再利用第一问的模型搜索该数独是否有唯一解，如果有，则构造完成。

关键词　深度优先搜索; 剪枝; Kakuro 数迷

1 问题重述

1.1 问题背景

Kakuro 数独是日本 Nikoli 公司开发的一款逻辑解谜问题，其与传统数独玩法相近但求解难度更大。Kakuro 数独的具体规则如下：在空格中只能填入整数数字 1-9，数字 0 不能出现；带斜线的方格，斜线上方的数字等于该方格右面对应的一组水平空格里的数字之和；斜线下方的数字，等于该方格下面对应一组垂直空格里的数字之和；同一数字在每组水平（垂直）空格里只能出现一次。当所有空格都填写完毕且满足上述条件时，问题即被解决。

1.2 需要解决的问题

在假定所有 kakuro 数独都为 8*10 的规格下，我们需要建立数学模型解决以下问题

- 对于任意给定的 kakuro 数独，建立一个求解模型，并提出一套合理可行的表示题目复杂度的计算方式。
- 在问题一复杂度讨论的基础上给出划分 Kakuro 数独难度的一种方式，并给出在此种复杂度计算规则下不同难度题目实例
- 结合问题二、三，找到不同难度 Kakuro 问题的特点并据此给出不同难度级别的 kakuro 数独的一种生成方式。讨论判定 Kakuro 数独只有唯一解的判定条件。

2 问题分析

2.1 kakuro 数独求解模型的建立及复杂度分析

查阅文献可知，判定 Kakuro 数独是否有解是一个 NPC 问题。（参考文献给出了将 Kakuro 规约为 3-SAT 问题的方法，并给出了一种解法。）则给出其任意一组解这一更强的命题也至少是一个 NPC 问题，否则就给出了一个多项式时间内的构造性解法，而这与前面的结论是矛盾的。

为进行 Kakuro 数独的求解，一个自然的想法是将需要填写的空白格作为变元、将斜线格中的约束条件作为方程关系，建立一个系数仅为 0 或 1 ，取值为 1 到 9 的线性方程组，这个方程组还具有一类变量中不能出现重复数字的约束。那么解出这样一种特殊的线性方程组的过程即为解出对应的 Kakuro 数独的过程。

但是，基于前面的论述，即使运用这种办法，也不可能得到多项式解法。我们做一个简单的分析，由于建立的线性方程组未必满秩，所以通过解线性方程组，我们只能得到各个变量与自由变量的关系，而考虑到游戏规则中要求的每个空格中只能填 1 到 9 的整数且一组连续空格中不能有重复元素，故在解出变量之间的关系后，求出最终的解的

过程和整数规划相比有更强的结构。我们没有很好的手段同时将组合学问题和代数学问题完美的结合在一起。从这种观点出发，这一条路很难给我们带来实际上的复杂度上的优势，作为判定唯一性的方法也难以实现，因而我们放弃这种想法。

联想数独等问题的解决方案、不妨从搜索入手，解决这个问题。我们对空白格进行搜索，在不做任何优化的情况下，当搜索量达到 9 的 n 次方时，一定能找到答案，但显然单纯的搜索复杂度表现自然不好，对此我们提出通过考虑连续空格组的唯一分解方式、重叠组等剪枝的方案来优化搜索的次数，此即我们的 kakuro 求解模型。

2.2 给出划分 kakuro 数独难度的一种方式，并给出实例

在第一问算法的实现中，我们发现在未知数给定的情况下，方程数越多，搜索的用时越短；类似的，在方程数给定的情况下，未知数越多，搜索的用时越长。我们用线性代数的观点同样能得出类似的结论，当未知数给定时，方程数越多，已知信息越多，方程组解起来越容易；当方程数给定，未知数越多，确定每个未知数的值越困难。因而我们可以用方程数的个数（即所有斜线格中的数字个数之和）与未知数的个数（即空白格）的比值来衡量给定的 Kakuro 数独的难度，比值越小，数独越困难；比值越大，数独越容易。

2.3 给出不同难度级别的 kakuro 数独的一种生成方式，并保证生成的数独问题为唯一解

结合前两问，有一种比较自然的想法，就是先确定方程（斜线格中的数字）的个数和未知数（即空白格的个数）确定难度，然后产生一个有可行解的数独，再次搜索除该可行解外的其他解，如果有则不是唯一解的数独，通过这种方式来保证产生的数独为唯一解。

3 模型假设与约定

假设一: 假设图形均为 8X10

假设二: 给定的需求解的 Kakuro 数独最外一层不为阴影，即给定的 kakuro 整体不能化为更小规模的 kakuro

4 符号说明及名词含义

名词	含义
组	夹在包含对角线格子和同行同列最近的灰色终止格子之间的连续格子个数
解生成树	生成本题结果过程中产生的树状数据结构
儿子节点	某一个节点对某组做出判断后产生的所有下一步可能情况
层	图中每少一个空组即解生成树向下延伸一层
可行组合	符合题目条件的能完全填充满整个图的空白格子填充方案
难度系数	kakuro 数独中空白格个数与斜线格中数字个数之和的比值
N	每个图中组的个数
n	每个图中空白格子总个数

5 模型建立与求解

5.1 kakuro 数独求解模型的建立

我们构造的 kakuro 数独的求解模型基于有剪枝的搜索，下面首先要说明其依据的剪枝原理。

5.1.1 唯一分解方式（非唯一分解方式）

对于一定的组和及组中连续空白块的个数，该组和只有有限种固定的分解方式，详见附录一。

5.1.2 重叠组

如果两个正整数的某个分划有相同的正整数，那么 Kakuro 布局上两个正整数所代表的 ”空档” 的交叉的取值一定取这些相同的正整数。下面举个例子。

我们观察两组被黄色标注的连续空白格组，由于 29 分为 4 个部分具有唯一的分解方式，即 29=5+7+8+9，而 6 分为两个部分的分解方式有两种：6=1+5, 6=2+4。由于这两组连续空白格组具有交叉点，所以交叉点必定为 29 与 6 分解后元素的公共元素，即为 5.

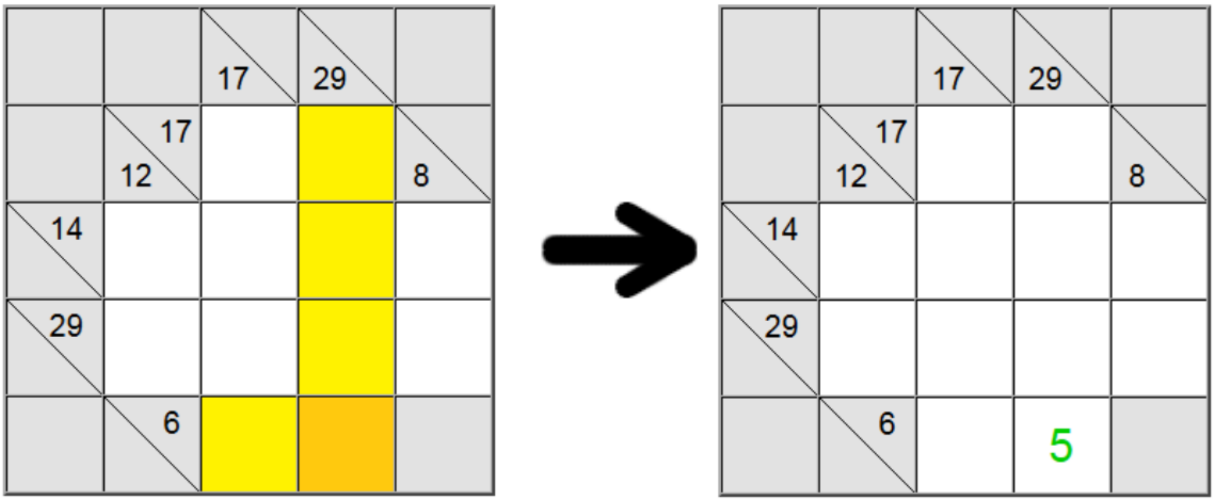

图 1: 重叠组

5.1.3 算法介绍

1. 从左到右遍历所有列中的“组”，对每个组判断可能的组合，特别的，若组和及其对应的分解元素的个数与 5.1.1 中的对应，则其分解方式唯一；每种组填充方式作为此节点的一个儿子节点，构造解生成树。

2. 对于解生成树第二层每一个节点，填好相应的第一个组的填充情况，寻找此时可能情况最小的组向下延伸，不断递归直到填满整个图或者无法填充，此节点作废。接下来向上回溯并向后遍历，亦即，采用前序遍历进行深度优先搜索。

3. 对于前两问，只要能够解出一组解即可，遍历到可行组合后程序终止。对于第三问，我们改用广度优先搜索，计算并储存完整个层的所有情况后再计算下一层，遍历得到此图的所有解。

4. 对于有交叉的横纵组，由于交叉点上元素必然相同，如果有相同元素则相同元素必在交叉点上。在向下构造结果生成树时可以利用这个特点进行剪枝加快程序运行。

5.1.4 算法复杂度分析

对于一段连续空白格，长度记为 n，完全暴力的、不考虑剪枝的搜索需要 9 的 n 次方，但利用剪枝 1、剪枝 2，我们可将 9 的 n 次方化为 $n!(n <= 9)$（对于有唯一分解方式的组），这大大提高了搜索的效率，但对于 kakuro 数独整体，由于其运行时间会根据有没有剪掉枝而浮动，我们能够确定一个大致的复杂度浮动范围。根据附录一我们可以看到，每一组大概有 5-8 种组合，因此估计复杂度在 $5^n - 8^n$ 之间。

5.2 kakuro 数独难度评估

由于在未知数给定的情况下，方程数越多，搜索的用时越短；类似的，在方程数给定的情况下，未知数越多，搜索的用时越长，所以我们用连续空白格组的个数与空白格个数的比值来衡量游戏难度，当比值较小时，难度较高，当比值较大时，难度较低。不过这与我们一般意义上的数值越大难度越高、数值越低难度越低的认知是不一致的，故我们取其倒数，即空白格个数与连续空白格组的个数的比值作为衡量难度的标志，称其为难度系数。注意到，在假设 2 的保证下，我们所研究的各个 kakuro 规模相近，故这种相对难度的评估方式可以表示 kakuro 问题的难度。

下面举两个例子来说明这一点。第一个例子为样例，第二个例子来自于 google play 的一个 kakuro app 前者可有 45 个未知数，构造方程 27 个，其难度系数为 1.67，后者有 43 个未知数，可构造方程 30 个，其难度系数为 1.43，故前者比后者难度高。

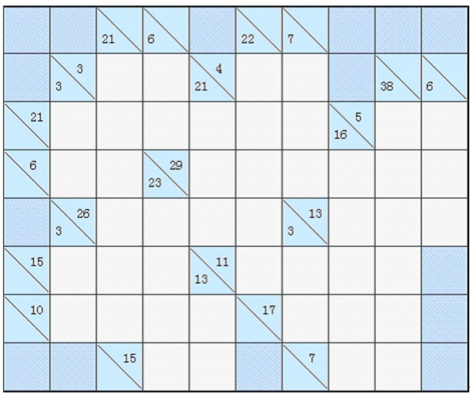

图 2: 较难的图

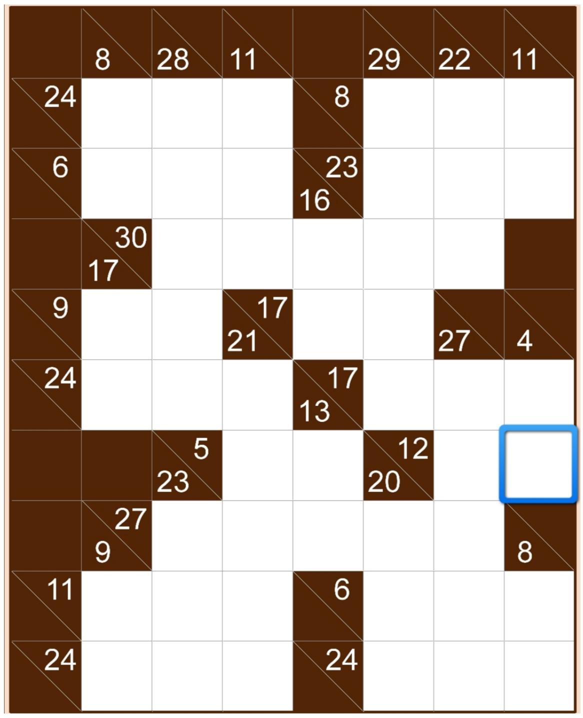

图 3: 较简单的图

根据上述评估规则，可知前者比后者更难。事实上，经程序验证，前者用时比后者长，这也验证了这一点。

在给定数独正确的条件下，如果线性方程组的个数大于等于未知数个数时，则必定有唯一解，此时难度系数小于等于 1，将小于等于 1 的定义为简单；将难度系数在 1 到 1.5 范围内的数独难度定义为中等；将难度系数大于 1.5 的数独的难度定义为困难，则上面两例中前面一例的难度为困难，后一例的难度为中等。

5.3 不同难度级别的 kakuro 数独的生成及唯一解的保证

由于存在上述的困难，我们可以说没有很好的一个 Kakuro 数独有唯一解的充分必要条件（线性方程组满秩并不是 kakuro 数独有唯一解的充要条件，因为还有整数约束条件，且当线性方程组不满秩，即方程组个数小于未知数的个数时，由于整数及一组连续空格中元素互不相同的约束条件的存在，仍可能存在唯一解）。因而更进一步的，我们也没有办法构造性的生成一类 Kakuro 数独。即使存在这样的方式，也因为它不是多项式时间内可解的，意义并不很大，不容易完成我们期望的目标。

我们认为，唯一可行的方式，仍然是根据前文所归纳的求解条件，随机生成若干个潜在的数独方案，随后完全求解，看其是否具有多组解，保留具有唯一解的数独方案。

6 模型检验与评价

6.1 模型检验

利用上述模型，我们得到了样例的结果，经验证，它能准确迅速得到正确结果。

```
ans =

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

图 4: 正确的结果

6.2 模型优缺点

6.2.1 优点

- 使用了基于深度优先搜索的剪枝求解算法，能够在可以接受的时间内给出给定的 Kakuro 数独的解法。
- 能对给定的 Kakuro 数独给予定性的难度评估。
- 能够给出一种不同难度级别的 Kakuro 数独的生成方式。

6.2.2 缺点

- 搜索时间波动较大
- kakuro 数独的难度评估具有一定的局限性，只能分析规模相仿的 kakuro 数独，对于规模具有差异的 kakuro 数独，不能给出一个恰当的难度评估。
- 模型中给出的不同难度的 kakuro 数独的生成方式尽管可以达到效果，但缺乏目的性，且效率较低，不能视作一个有效的生成方式。

7 结论

事实上，出于问题本身的 NPC 性质，我们事实上较为缺乏十分有力的问题求解手段。在解决一般问题中常用的利用一些数学上的性质，逐步规约复杂度到多项式级别的方式，在这一问题上一开始就不可能适用。我们实际上只有唯一的一条道路，即搜索-剪枝的道路，只是这条道路如何去走来或多多少压低一点平均情况（甚至很难说压低最坏情况），多从这条道路上压榨出一些性能，压榨出一些额外的性质的问题。

我们认为，单从问题求解的角度出发，最好的方法仍然是使用最朴素的搜索和剪枝，在这里进行模型转换的代价很大（建立一个 K 组方程的 N^2 个变量的方程组并求解本身代价已经足够巨大），也会使得我们失去一些易于实现的性质。毕竟如果在另一个模型中，性质本身的判断代价增大了，那么这样的模型转换是不存在问题求解上的意义的。

参考文献

[1] 姜启源，谢金星，叶俊. 数学模型 (第四版)[M]. 北京: 高等教育出版社,2011.

[2] 司守奎，孙兆亮. 数学建模算法与应用 [M]. 国防工业出版社.2015

[3] Simonis H.Kakuro as a constraint problem[J].Proc.seventh Int.Works.on Constraint Modelling and Reformulation.2008.

[4] Ruepp O.Holzer M.The computational complexity of the KAKURO puzzle.revisited[C].International Conference on Fun with Algorithms.Springer.Berlin.Heidelberg.2010:319-330.