arXiv:2601.18450v1 [cs.DM] 26 Jan 2026

关于具有不同的行的二进制矩阵中重列的存在性

Jamolidin K. Abdurakhmanov 副教授、博士
软件工程系 安集延国立大学 乌兹别克斯坦安集延
jamolidinkamol@gmail.com, abduraxmanov@adu.uz
ORCID：0009-0008-0067-0553

摘要。 我们研究具有不同的行的二进制矩阵中重列的存在性。若二进制矩阵的一列中一的数量至少为 $\lceil m/2 \rceil$，则称该列为重列。我们提出两种递归算法， A_1 和 A_2，通过行筛选和列删除来检查所得子矩阵的性质。我们证明，如果算法 A_1 对具有不同的行的二进制矩阵返回 True，那么该矩阵至少包含一个重列（定理 1）。此外，我们证明，如果算法 A_2 对具有不同的行、不同列且不含全零列的二进制矩阵返回 True，那么该矩阵也至少包含一个重列（定理 2）。 A_2 中的关键创新是一个提前终止条件：如果某列中恰好有一行的元素为零，则立即将该列判定为重列。证明采用了一种新颖的论证方法，基于特定列对应的未配对行的存在性，并结合对算法递归结构的细致分析。

2020年数学主题分类。主要分类：05B20；次要分类：05D99，68R05。

关键词和短语。二进制矩阵、重列、递归算法、组合学、多数、投票理论。

1. Introduction

二进制矩阵自然地出现在离散数学和计算机科学的许多领域中，包括编码理论、组合优化、数据库理论和社会选择理论。对二进制矩阵结构性质的研究，尤其是对行或列施加不同性约束的矩阵，有着悠久的历史，可追溯至 Ryser [12, 13] 和 Gale [7], 的经典研究，并不断产生有趣的结果 [4, 3]。

本文探讨一个自然的问题：具有不同的行的二进制矩阵在什么条件下会包含一个重列，也就是说，该列中一的数量至少与零的数量相等？这个问题与投票理论中的问题直接相关：重列对应于获得多数选民认可的候选人 [2], ；它还与组合学中的覆盖问题以及密码学中对布尔函数的分析有关 [5]。

我们通过引入两种递归算法，以算法方式研究这一问题。这些算法系统地检查通过行筛选和列删除得到的子矩阵。我们的主要结果表明，这些算法的返回值为重列的存在提供了充分条件。这种证明方法——基于识别未配对行，并构造经过递归树的特定路径——在二元矩阵理论中似乎尚属首创。

本文结构如下。第 2 节讨论相关研究及我们结果的新颖之处。第 3 节介绍潜在应用。第 4 节给出基本定义和符号。第 5 节介绍算法 A_1 并证明定理 1。第 6 节介绍算法 A_2 并证明定理 2。第 7 节为结论和未解决的问题。

2. Related Work and Novelty

2.1. 二元矩阵的经典结果。 二元矩阵的组合理论历史悠久。奠基性的 **Gale-Ryser 定理** [7, 12]刻画了一对序列何时能够实现为二元矩阵的行和与列和。具体而言，序列 $(r_1, \dots, r_m)$ 和 $(c_1, \dots, c_n)$ 可实现，当且仅当满足某些主序条件。此定理已在诸多方向上得到推广，包括 Barvinok [3] 提出的渐近计数结果。

Sauer-Shelah-Perles引理 [14, 15] 根据VC维数给出了二元向量族的界：如果一个族 $\mathcal{F} \subseteq \{0,1\}^n$不打散任何大小为 $d+1$ 的集合，那么 $|\mathcal{F}| \le \sum_{i=0}^{d} \binom{n}{i}$。尽管该引理关注的是子集关系而非列权重，但它代表了对二进制矩阵结构的另一种经典约束。

Brualdi 的综合性专著 [4] 涵盖了组合矩阵类别，包括 (0,1)-矩阵、永久和拉丁方，为该领域奠定了许多现代术语。Ryser 早期的专著 [13] 仍颇具影响力，因其探讨了矩阵的组合性质。

2.2.竞赛矩阵与得分序列。 竞赛矩阵中，元素 $(i, j) = 1$ 表示选手 i 击败了选手j，这是二元矩阵理论的另一个经典背景。**Landau 定理** [9]刻画了可实现的得分序列：序列 $(s_1, \dots, s_n)$ 满足 $s_1 \le \cdots \le s_n$，当且仅当对于所有 k，都有 $\sum_{i=1}^{k} s_i \ge \binom{k}{2}$，它才是某个竞赛的得分序列。Kolesnik [8]近期的研究确定了得分序列的渐近数量。

2.3. 我们结果的新颖性。 尽管相关文献丰富，我们仍未发现任何现有定理能够根据行的互异性，确立保证二进制矩阵存在重列的条件。Gale-Ryser定理讨论了具有指定行列和的矩阵是否存在，但没有考虑矩阵在何种条件下必须 具有权重超过阈值的列。Sauer-Shelah引理对集合族的大小作出限制，但并未直接讨论列权重。

与此最接近的现有研究涉及针对二进制矩阵的鸽巢原理论证，例如证明一个含有 58 个一的 14×14 矩阵必定包含一个全一的 2×2 子矩阵。这些研究采用了相关的双重计数技巧，但探讨的是本质上不同的结构性问题。

我们的工作填补了这一空白：我们证明，行互异性结合特定的算法条件，可以保证重列的存在。此外，这种递归证明技巧——按列值筛选行，并分析所得子矩阵——在用于确立列权重性质方面似乎具有方法上的新颖性。

3. Applications

重列这一概念在多个领域中都有自然的诠释。

3.1. 投票理论与社会选择。 在赞成投票中，选民（行）通过在相应条目中填入 1，表明他们赞成哪些候选人（列）。重列 对应于获得至少半数选民赞成的候选人——这是一种自然的多数支持概念。

合理代表 标准在委员会选举中 [2] 要求，具有共同获批准候选人的足够大选民群体应获得代表。这本质上是一种局部列权重条件。我们的结果给出了充分条件：当选民拥有不同的偏好档案时，至少有一名候选人必须获得多数选民的批准。

3.2. **编码理论**。 在编码理论中，校验矩阵的列决定了错误检测和错误纠正能力。针对 Reed-Muller 码的 Reed 多数逻辑译码 [11] 通过在相互独立的恢复集上进行多数表决，恢复每个消息符号。覆盖半径定理将校验矩阵的列属性与纠错能力直接联系起来。

Lyand Soljanin 的近期研究 [10] 表明，对于 Reed-Muller 码，一步多数逻辑译码器最多只能纠正 $d_{\min}/4$个错误，这凸显了基于多数的分析在编码理论中仍然具有重要意义。

3.3. **密码学**。 密码安全通常要求使用平衡布尔函数——其真值表中的一和零数量相等。不平衡函数容易受到相关攻击。对 k 阶相关免疫性的研究与正交数组结构 [5]有关。

我们关于重列的结果给出了不平衡（即一的数量更多）必然出现的条件，这有助于理解某些密码学漏洞何时不可避免。

3.4. **数据挖掘**。 在事务数据库中，二进制矩阵表示各项在不同事务中的出现情况。某项的支持度 等于其列权重除以事务总数。重列对应于频繁出现的项目，这是 Apriori [1] 等关联规则挖掘算法的出发点。

3.5. **集合覆盖与组合优化**。在集合覆盖问题中，元素构成行，集合构成列，条目 $(i, j) = 1$ 表示集合 j 覆盖元素 i。贪心算法反复选择覆盖最多未覆盖元素的列——即残余矩阵中局部列权重最大的列。Chv´atal [6] 证明了，该算法可达到 ln(最大列权重) 的近似比 $+ 1$。

4. Preliminaries

在全文中，我们考虑二进制矩阵，即其条目属于集合 $\{0, 1\}$ 的矩阵。

定义 1 （重列）。 设 M 是一个大小为 $m \times n$、元素取自 $\{0, 1\}$的二进制矩阵。若某列中的一的数量至少与零的数量相同，即一的数量至少为 $\lceil m/2 \rceil$，则称 M 的一列为重列 。

定义 2（行索引）。 对于一个有 n 列的二进制矩阵 M，设 M的一行 t，我们用 $t[k]$ 表示第 t 行、第 k列的元素，其中 $k \in \{1, 2, \ldots, n\}$。

定义 3（行数记法）。 对于任意矩阵 X，我们用 $|X|$表示 X的行数。

定义 4（子矩阵集合 S_k^0、 S_k^1和 S_k）。 设 M是一个大小为 $m \times n$的二进制矩阵。对于固定的列 $k \in \{1, 2, \ldots, n\}$，我们定义以下集合：

- 如果列 k 不含零，则 $S_k^0 = \varnothing$。否则，令M_k^0 为由 M 中所有满足位置 k 上的元素等于零的行构成的子矩阵。令 M_k^{0-} 为从 M_k^0 中删除列 k （该列全为零）后得到的矩阵。在这种情况下，$S_k^0 = \{M_k^{0-}\}$。
- 如果列 k 不含一，则 $S_k^1 = \varnothing$。否则，令 M_k^1为由 M 中所有满足位置 k 上的元素等于一的行构成的子矩阵。令 M_k^{1-} 为从 M_k^1 中删除列 k （该列全为一）后得到的矩阵。在这种情况下，$S_k^1 = \{M_k^{1-}\}$。
- 集合 S_k 定义为 $S_k = S_k^0 \cup S_k^1$。

定义 5（共轭行）。 设 M 是一个有 n列的二进制矩阵。 M 中的两个不同的行 t 和 s 称为关于列 k 共轭，当且仅当它们仅在位置 k上不同，即：

- $t[k] \neq s[k]$，并且
- $t[j] = s[j]$ 对所有 $j \neq k$均成立。

t 与 s 之间的汉明距离等于 1，且唯一不同的位置为 k。

定义 6（未配对行）。 矩阵 M 的行 r 称为关于列 k 未配对，如果 $r[k] = 0$，且在 M 中不存在与 r 关于列 k共轭的行。

定义 7（矩阵约简）。 设 M 为一个二进制矩阵， k 为一个列索引，$b \in \{0, 1\}$ 为一个值。将矩阵 M按列 k 和值 b进行约简，记为 M_k^{b-}，定义如下：

(1) 选取 M 中位置 k 的元素等于 b 的所有行。(2) 从所得子矩阵中删除列 k。

如果列 k 中没有等于 b的元素，则约简 M_k^{b-} 未定义（或视为空矩阵）。

定义8 (顺序约简**).** 设 M 是一个大小为 $m \times n$的二进制矩阵，设 r 是 M中的一行固定行，且 $l \in \{1, \ldots, n\}$ 是一个指定列索引。

矩阵 关于行 并保留列 的顺序约简定义如下:

设 $\{k_1, k_2, \ldots, k_{n-1}\} = \{1, \ldots, n\} \setminus \{l\}$ 是除 l 以外所有列的任意一种排列。定义矩阵序列:

- $M^{(0)} = M$
- $M^{(i)} = (M^{(i-1)})_{k_i}^{r[k_i]-}$ for $i = 1, 2, \ldots, n-1$

其中每一步都按列 k_i 和值 $r[k_i]$进行约简。

矩阵 $M^{(n-1)}$ 称为顺序约简结果，其大小为 $m' \times 1$，其中 $m' \geq 1$是某个值。

定义 9（一致行集合）。设 M 是一个大小为 $m \times n$ 的二进制矩阵， r 是 M 的一行， l 是一个固定列。与行 r 在列 l 以外一致的 M 行集合 定义如下:

$$\mathcal{C}(M, r, l) = \{t : t \text{ is a row of } M \text{ and } t[j] = r[j] \text{ for all } j \neq l\}.$$

这是与 r 在除位置 l之外的所有位置上都相同的 M 的所有行的集合。

5. Algorithm A_1 and Theorem 1

现在我们介绍第一个递归算法，并证明其返回正值即可保证重列的存在。

算法1：递归算法 $A_1(M, m, n)$
输入: 大小为 $m \times n$ 的二进制矩阵 M。
输出: True 或 False。

1.如果 $n = 1$ **那么2. 如果** 单列 $\geq$ 中一的数量为零的数量**那么3. 返回** True 4. **否则**5. **返回**False 6. **结束如果**7.**结束如果**8. **对于** 每一列 $k \in \{1, 2, \ldots, n\}$ **执行**9. 构造集合 S_k10. **如果** $\exists$ 矩阵 $K \in S_k$ 的所有列中零的数量都多于一的数量**那么**11. **返回**False 12. **结束如果**13. **对于每个** 含有 d 行和 $n-1$ 列的矩阵 $K \in S_k$ **执行** 14. $B \leftarrow A_1(K, d, n-1)$

15. **如果** B =False **则返回** False **结束如果**16.
结束循环 17. **结束循环** 18. **返回** True

注意，子矩阵中的行仍然互不相同，因为原矩阵的行互不相同。算法 $A_1(M,\ m,\ n)$ 递归检查子矩阵的性质，并在 $n = 1$ 时终止。第 8 行中处理列索引 $k \in \{1, 2, \ldots, n\}$ 的顺序可以任意。

定理 1. 设 M 是一个大小为 $m \times n$ （$m \geq 1, n \geq 1$）且各行互不相同的二进制矩阵。如果 A_1（M, m, n） =为真，则 M 至少包含一列重列。

证明。我们通过反证法证明该定理，并明确考察基本情形。

基本情况 1： $n = 1$。根据算法 A_1 的定义，当$n = 1$时，该算法当且仅当这一列中一的数量不少于零的数量时返回 True，也就是说，该列是重列。因此，该定理直接成立：如果 $A_1(M,\ m,\ 1)$ =为真，则存在一列重列（也是唯一的一列）。

基本情况 2： $m = 1$。矩阵只有一行，因此显然各行互不相同。对于任意列 k，其元素不是 0 就是 1。如果元素为 1，则该列是重列（有一个 1，没有 0）；如果元素为 0，则该列不是重列（有一个 0，没有 1）。算法递归检查子矩阵：对于含有 1 的列， M_k^1有 1 行， M_k^0为空；对于含有 0 的列，则相反。缩减后的矩阵有 $n-1$ 列，且经过筛选后只剩一行。最终，递归会到达满足$n = 1$的基本情形。如果所有列都不是重列（所有元素均为 0），那么某条递归路径会到达一个仅含一列且非重的子矩阵，从而得到 A_1 = False。因此，如果 A_1 =True，就必定至少存在一列重列（其元素为 $= 1$），这与一般定理一致。

这些基本情形说明了该算法如何确保递归树叶节点处满足这一性质。现在，对于一般情况（m >1，$n > 1$），我们通过反证法进行证明，并借助基本情形得出最终矛盾。

假设 $A_1(M,\ m,\ n)$ =为真，但 M没有重列；也就是说，对于每一列 k，其中一的数量都小于 $\lceil m/2 \rceil$ （零的数量多于一的数量）。

由于各行互不相同，因此它们是 $\{0,1\}^n$ 中互不相同的向量，所以$m \leq 2^n$。

任取一行 $r \in M$，其元素为 b_1、b_2、.....、$b_n \in \{0, 1\}$。

引理 1（共轭行引理）。设 M 是一个行互不相同的二进制矩阵。若对于每一列 k 以及每一行 t（满足 $t[k]=0$），都存在一行 s，使其相对于列 k（按定义 *5* 的意义）与 t 共轭，则每一列中零的数量都不超过一的数量。

引理 *1* 的证明。固定任意一列 k。考虑映射φ_k，它将每个满足 $t[k]=0$ 的行 t 映射到其共轭行 s，其中$s[k]=1$。

我们证明 φ_k 是单射。假设对于某些不同的行 t_1，t_2，且满足 $t_1[k]=t_2[k]=0$，有 $\varphi_k(t_1)=\varphi_k(t_2)=s$。那么：

- $t_1[j]=s[j]$ 对所有 $j\neq k$ 均成立（根据共轭的定义），
- $t_2[j]=s[j]$ 对所有 $j\neq k$ 均成立。

因此，$t_1[j]=t_2[j]$对所有 $j\neq k$ 均成立，且 $t_1[k]=t_2[k]=0$。于是 $t_1=t_2$，这与它们互不相同相矛盾。

因此，φ_k 是一个从第k 个位置为零的行集合到第 k个位置为一的行集合的单射。因此，第 k 列中零的数量不超过一的数量。□

由于根据假设，所有列中的零都比一多，根据引理 1 的逆否命题：

存在一列 l 和一行 r，使得 r 相对于列 l 未配对（按照定义 6 的含义）。

固定这样的 l 和 r。我们在该算法的递归树中构造一条路径：按照任意顺序移除除 l 以外的所有列（例如，从 1 到 n，但不包括 l）。每一步移除第$i\neq l$列时，我们选择子矩阵 $M_i^{b_i-}$（按 $b_i=r[i]$筛选）。

由于筛选依据的是行 r 自身的值，该行在每一步都会保留，因此中间子矩阵中的行数至少为 1。

删除 $n-1$次后，我们得到一个大小为 $m''\times 1$（$m''\geq 1$）的子矩阵 S，其中该列是第 l 列在剩余行上的限制。

在所有已删除的列（$n-1$）中，剩余行都与 r 相同。由于不存在仅在 l 处与 r 不同的 s（即 $s[l]=1$ 且其他位置一致），所有剩余行在位置 l 上的值均为 0。

因此，在 S 中：一的数量为 $=0$，零的数量为 $=m''\geq 1$。该列不是重列。

根据 $n=1$ 的基本情形（上文已明确说明），$A_1(S,\ m'',\ 1)$ =False。由于 S 是递归树中特定路径上到达的子矩阵，而算法要求所有递归调用（包括沿此路径的调用）都返回 True，以避免提前返回 False，因此这与 $A_1(M)$ = True 矛盾。

因此，该假设不成立，且 M 至少有一列重列。□

6. Algorithm A_2 and Theorem 2

现在我们介绍第二种递归算法，它与A_1 的区别在于增加了提前终止条件。关键创新是第 13–15 行中的条件：如果第k列中恰好有一行的元素为零，算法便立即返回 True，因为该列必定是重列。

算法 2：递归算法 $A_2(M, m, n)$
输入：大小为 $m \times n$ 的二进制矩阵 M。
*Outp*ut：真或假。0. 如果 $m = 1$ 且 $n > 1$ 则 1. 返回真 2. 结束如果 3. 如果 $n = 1$ 则 4. 单列 $\geq$ 中的一的数量大于零的数量，则 5. 返回真 6. 否则 7. 返回 False 8. 结束如果 9. 结束如果 10. 对于 $k = 1$ 到 n 执行 11. 创建（形式） M_k^{0-} // 参见定义 4 12. 创建（形式） M_k^{1-} // 参见定义 4 13. 如果 $|M_k^0| = 1$则 // 关键条件 14. 返回真 15. 结束如果 16. $S_k \leftarrow \varnothing$17. 如果 M_k^{0-}是非空矩阵，则 $S_k \leftarrow S_k \cup \{M_k^{0-}\}$结束如果 18. 如果 M_k^{1-}是非空矩阵，则 $S_k \leftarrow S_k \cup \{M_k^{1-}\}$结束如果 19. 对每个 $K \in S_k$ 执行 20. 如果 K 的所有列中零的数量都多于一的数量，则 21. 返回假 22. 结束如果 23. 结束循环 24. 对每个 $K \in S_k$ 执行 25. $d \leftarrow |K|$ // K中的行数 26. $B \leftarrow A_2(K, d, n-1)$

27. 如果 $B =$False **则**
28. 返回 False 29. **结束如**
果30. **结束循环**31. **结束循**
环32. **返回**真

定理 2. 设 M 是一个大小为 $m \times n\ (m \geq 1, n \geq 1)$的二进制矩阵，其行各不相同、列各不相同，且没有全零列。如果$A_2(M, m, n) =$ True，则 M 至少包含一列重列。

证明. 我们证明其逆否命题：若 M 没有重列，则$A_2(M, m, n) =$ False。

基本情况 1： $n = 1$。若 M 没有重列，则该单列中零比一多。算法在第 6–7 行返回 False。

基本情况 2： $m = 1$ **且** $n > 1$。如果没有重列，那么每列都只包含 0（因为一个 1 就足以使该列成为重列）。但这与没有全零列的条件矛盾。因此，这种情况不可能发生。

基本情况 3： $|M_k^0| = 1$ **对于某一列** k。如果算法在第 13–14 行返回 True，我们就验证 M 确实有一列重列。

如果恰有一行在第 k 列中为 0，则有 $m - 1$ 行在第 k 列中为 1。第 k 列有 $m - 1$ 个一和 1 个零。对于 $m \geq 2$:

$$m-1 \geq \lceil m/2 \rceil \iff m-1 \geq \frac{m+1}{2} \iff 2m-2 \geq m+1 \iff m \geq 3.$$

对于 $m = 2$: $m - 1 = 1 \geq \lceil 2/2 \rceil = 1$。因此，对于所有 $m \geq 2$，第 k 列都是重列。

一般情况： $m \geq 2$**、** $n \geq 2$**，且对于所有** k**，都有** $|M_k^0| \geq 2$。

由于 M 没有重列，对于每一列 k，零的数量都多于一的数量。特别是，$|M_k^0| > |M_k^1|$，这意味着 $|M_k^0| \geq 2$（因为 $|M_k^0| + |M_k^1| = m$ 且 $|M_k^0| > m/2$）。

因此，该算法不会在第 13–15 行返回 True。

根据引理 1 的逆否命题，由于所有列中的零都多于一，因此存在一个相对于某列l的未配对行 r。这意味着 $r[l] = 0$，并且 M 的任何一行都不与 r 相对于 l共轭。

构造一条通向False**的路径**。

我们构造一条通向 False 的特定递归树路径。

考虑相对于行 r 对 M 进行的顺序约简，同时保留列 l。也就是说，对于每一列 $k \neq l$ （按某种顺序），我们按值 $r[k]$ 进行筛选，并移除列 k。

在每一步中，我们处理列 k ，并选择包含行 r的子矩阵：

- If $r[k] = 0$, we take M_k^{0-} (which contains row r).
- If $r[k] = 1$, we take M_k^{1-} (which contains row r).

行 r 在每一步中都得以保留，因为我们始终根据该行自身的值进行筛选。

经过 $n-1$ 次这样的约简后，我们得到一个大小为 $m'' \times 1$的矩阵 S（其中 $m'' \geq 1$），它只包含列 l。

断言： S 的所有元素均为 0。

断言的证明： S 的行恰好是 M 中与 r 在所有位置 $k \neq l$上都相同的行，即定义 9 中的集合 $\mathcal{C}(M, r, l)$。

假设某一行 $t \in \mathcal{C}(M, r, l)$ 具有 $t[l] = 1$。由于对所有 $k \neq l$ 和 $t[l] = 1 \neq 0 = r[l]$, $t[k] = r[k]$ 均成立，因此该行 t 只在位置 l 上与 r 不同。根据定义 5, t 关于 l 与 r 共轭。但这与 r 关于 l 未配对这一事实矛盾。

因此，对于所有 $t \in \mathcal{C}(M, r, l)$, $t[l] = 0$ 均成立，并且 S 的所有元素都是 0。 □

由于 S 有 $m'' \geq 1$ 个零和 0 个一，这唯一的一列不是重列。根据基本情况 1, $A_2(S, m'', 1) = \text{False}$。

False 的传播。

我们必须验证所构造的路径有效，即算法在到达 S 之前，不会在第 20–22 行返回 False。

在每个中间步骤，我们都有一个矩阵 $M^{(i)}$，其中有 $n-i$ 列（包括列 l）。对于我们处理的每一列 k，我们选择沿着分支 $M_k^{r[k]-}$ （其中包含行 r）继续。

假设在某一步，算法在第 20–22 行针对子矩阵 $K = M_k^{r[k]-}$ 返回 False。这意味着 K 的所有列中，零都比一多。

但是， K 包含行 r （或其投影）。如果 K 的所有列中，零都比一多，那么特别地，K 中列 l 的投影中，零也比一多。由于行 r 在列 l 中贡献了一个 0 （因为 $r[l] = 0$），并且 K 的所有行在迄今处理过的列上都与 r 一致，这与我们的构造相符。

然而，关键在于，算法会按顺序检查从 1 到 n 的所有 k，并且对于每个 $K \in S_k$，它会先检查第 19–23 行，然后再进行第 24–30 行中的递归调用。

由于我们要证明逆否命题（如果没有重列，那么$A_2 = \text{False}$），因此需要证明该算法最终 会返回 False。出现这种情况的原因可能是：

(1) 对于某个 $K \in S_k$，在第 20–22 行；或 (2) 当递归调用返回 False 时，在第 27–29 行。

我们的构造表明，沿着由行 r 确定的路径，最终会到达 S，其中 $A_2(S) = \text{False}$。这个 False 值在第 7 行（基本情形）返回，并通过第 28 行逐层传播回来。

矩阵 S 是通过一系列特定的递归调用到达的：

$$A_2(M) \to A_2(M^{(1)}) \to A_2(M^{(2)}) \to \cdots \to A_2(S) = \text{False},$$

其中，每个 $M^{(i)}$ 都是相应层级某个 S_k 中的子矩阵。

由于 $A_2(S) = \text{False}$，且只要任何递归调用返回 False，算法就会返回 False（第 27–29 行），因此 False 值会沿着递归过程逐层传播回来。

因此，$A_2(M,\ m,\ n) = \text{False}$。

结论： 我们已经证明，如果 M 没有重列，那么$A_2(M) = \text{False}$。根据逆否命题，如果 $A_2(M) = \text{True}$，那么 M 至少有一列是重列。□

备注。 定理 2 的条件不可或缺：

- **不同列：** 若没有这一条件，一个具有 $m = 1$、$n > 1$且所有 $[0]$ 列都相同的矩阵将没有重列，但A_2 会在第 0–1 行返回 True。
- **没有全零列：** 若没有这一条件，基本情形 $m = 1$可能不成立，而未配对行的论证也可能无法得出有效的 False 路径。
- **不同的行：** 这确保引理 1 适用，并确保在没有重列时存在未配对行。

7. Conclusion

我们已经证明了两个定理，将递归算法 A_1 和 A_2 的返回值与具有互异性约束的二进制矩阵中重列的存在联系起来。两个证明的关键技巧都是识别未配对行，并在算法的递归树中构造特定路径，从而导出矛盾。

7.1. **贡献总结**。

(1) **定理 1** 确立了这样一个结论：对于具有不同的行的二进制矩阵，如果算法 A_1 返回 True，则该矩阵至少包含一个重列。

(2) **定理 2** 确立了算法 A_2 的类似结论，但附加了不同列且不存在全零列的约束。A_2 中的关键创新是提前终止条件：如果某列 k 中恰好有一行的值为零，那么列 k 就有 $m-1$ 个一，因此对于 $m \geq 2$ 而言是重列。(3) 这种**证明技巧**基于共轭行、未配对行和顺序约简，在二元矩阵理论领域似乎是全新的，并且可能适用于其他结构性问题。(4) 这些结果在投票理论（多数候选项）、编码理论（多数解码）、密码学（平衡函数）和数据挖掘（频繁项）中都有**自然的解释**。

7.2. **未解决问题**。未来的研究仍有几个问题尚未解决：

(1) **算法构造**。能否修改算法 A_1 和 A_2，使其真正找到一个重列（而不仅仅是证明其存在）？(2)**计算复杂度**。算法 A_1 和 A_2的时间复杂度是多少？递归结构表明，其最坏情况可能呈指数级增长，但平均情况或受限实例可能更易处理。(3)**组合刻画**。是否存在自然的组合条件，能够等价于 $A_1(M) = $ 为真或$A_2(M) = $ 为真？这类刻画可以更深入地揭示重列必然存在的情形。(4) **推广**。能否应用类似技术研究具有其他权重性质的列的存在性，例如权重恰好为 $\lfloor m/2 \rfloor$ 的列，或权重处于指定范围内的列？(5) **定量界**。当算法返回 True 时，我们能否界定重列的数量，或刻画哪些列必然是重列？(6) **与其他问题的联系**。我们的条件与Gale-Ryser定理或绍尔-谢拉引理等经典结果之间的确切关系是什么？

Acknowledgments

作者感谢安集延国立大学的同事们进行的有益讨论。

References

[1] R. Agrawal 和 R. Srikant，快速挖掘关联规则的算法，《超大型数据库》第 20 届国际会议论文集（VLDB），1994 年，第 487–499 页。[2] H. Aziz、M. Brill、V. Conitzer、E. Elkind、R. Freeman 和 T. Walsh，基于赞成票的委员会投票中的合理代表，《社会选择与福利》48（2017），461–485。[3] A. Barvinok，关于具有给定行和与列和且元素为 0-1 元素的矩阵数量及随机矩阵，《数学进展》224（2010），316–339。[4] R. A. Brualdi，《组合矩阵类》，《数学及其应用百科全书》第 108 卷，剑桥大学出版社，剑桥，2006。[5] C. Carlet，用于密码学和纠错码的布尔函数，载于：《数学、计算机科学与工程中的布尔模型与方法》（Y. Crama 和 P. L. Hammer 编），剑桥大学出版社，2010 年，第 257–397 页。[6] V. Chv´atal，集合覆盖问题的贪心启发式，《数学运筹研究》4（1979），233–235。[7] D. Gale，网络流的一个定理，《太平洋数学杂志》7（1957），1073–1082。[8] J. Kolesnik，得分序列的渐近数量，《组合数学》43（2023），1–20。[9] H. G. Landau，关于支配关系与动物社会的结构。III. 得分结构的条件，《数学生物物理学通报》15（1953），143–148。[10] T. Ly 和 E. Soljanin，关于 Reed-Muller 码的多数逻辑译码，《IEEE 信息论汇刊》，2025 年（即将刊出）。[11] I. S. Reed，一类多重纠错码及其译码方案，《IRE 信息论汇刊》4（1954），38–49。[12] H. J. Ryser，零一矩阵的组合性质，《加拿大数学杂志》9（1957），371–377。[13] H. J. Ryser，《组合数学》，Carus 数学专著丛书第 14 卷，美国数学协会，1963。[14] N. Sauer，关于集合族的密度，《组合理论杂志》A 辑 13（1972），145–147。[15] S. Shelah，一个组合问题；无穷语言中的模型与理论的稳定性与序，《太平洋数学杂志》41（1972），247–261。