杭州电子科技大学

第二十三届校数学建模竞赛

学　　校	杭州电子科技大学
参赛队号	研 2022090
队员姓名	1.马迦南
	2.卫锐萌
	3.马溪彤

杭州电子科技大学

第二十三届校数学建模竞赛

题 目 基于分群思想和贪心算法数据隐藏的策略

摘 要：

伴随着网络化与信息化的发展，信息呈“爆炸式”增长，大数据时代正悄无声息到来。如何在大数据计算环境下保障数据的隐私性的同时，尽可能提高数据可用性和计算的高效性，逐渐成为大数据隐私保护领域的研究热点之一。本文将在上述背景下，将信息抽象为数学问题，从二元到多元，从不限定到限制匹配个数，不断加入新约束新条件，进行数据隐藏策略的研究。

问题一是对例如性别等可简化为只有两个不同选项的数据信息，抽象为处理只含有0，1数据的数学问题。对应信息位不相同的值变更为*号，使变更后的数据总能匹配到至少一条除自身外的其他数据，以到达隐藏数据的目的。我们通过设立表示信息位是否被隐藏的决策变量，建立以最小化隐藏数据量为目标的数学模型。并设计评价指标评估策略好坏，并基于分群思想采用贪心算法求解模型。针对二元1数据集，我们生成隐藏20个数据信息位的方案，共进行8次两两匹配，算法平均用时0.03s。对于二元2数据集，最终生成隐藏361个数据信息位的方案，其中，两两配对共94次，个体与群进行匹配共2次，算法平均用时6.6s。

问题二在问题一的基础上，将数据由二元变为多元，也就是扩大了数据信息位的取值范围。对于多元1数据集，最终生成隐藏634个数据信息位的方案，其中进行了97次两两匹配和2次个体与双人群匹配，算法平均用时6.3s；对于多元2数据，得到隐藏7978个数据信息位的方案，其中400次均为个体间两两匹配，算法平均用时689s。具体匹配操作在文中以及附件中均有详细展示。

问题三延用前两问数据集，并引入p-重保护的概念，即个体能够隐藏在一个至少包含p个个体的数据组之中。我们仍以最小化隐藏数据量为目标，使用基于分群思想的贪心算法在四个数据集上分别计算$p = 3,5,8$时的数据隐藏方案。$p = 3$时，二元1隐藏67个数据信息，平均用时0.053s；二元2隐藏686个数据信息，平均用时6.9s；多元1隐藏927个数据信息，平均用时6.76s；多元2隐藏10115个数据信息，平均用时728.9s。$p = 5$时，二元1隐藏157个数据信息，平均用时0.068s；二元2隐藏116个数据信息，平均用时7.28s；多元1隐藏1165个数据信息，平均用时6.99s；多元2隐藏11230个数据信息，平均用时780.7s；$p = 8$时，二元1隐藏230个数据信息，平均用时0.077s；二

元 2 隐藏 1492 个数据信息，平均用时 7.51s；多元 1 隐藏 1301 个数据信息，平均用时 7.27s；多元 2 隐藏 11646 个数据信息，平均用时 786.2s。具体的方案与性能评估在文中做出呈现。

问题四针对三种不同的情形：限制隐藏模式、限定全发布数据以及用户定制保护力度，提出了进一步的要求。本文通过对情形一和情形二进行分析，并对前文设计的算法进行改进。最终，本文以限定部分数据全发布的多元 1 数据集为例，设计并使用算法进行求解。结果表明，在限定数据全发布且不考虑隐私暴露的情况下，本文算法隐藏 399 个数据信息，完成了数据隐藏任务。在时间性能方面，本文的算法平均用时 6.3s，相比于不限定数据全发布的情况能更快的求出结果。

关键字 贪心算法；分群策略；数据隐藏；评价指标

目录

一. 问题重述

1.1 数据隐藏技术背景

随着云计算与大数据技术的发展，亚马逊、微软、华为与阿里等主流云服务提供商支持云端部署分布式存储和计算框架，主要包括批量计算框架、流式计算框架和机器学习框架，如 TensorFlow[1]等，为用户提供持续可靠、可扩展且高吞吐量的大数据存储和计算服务。但在这种外包的大数据计算环境下，由于数据所有权和使用权的分离，在计算过程涉及的数据输入、计算和输出等阶段都有可能发生隐私数据泄露的风险[2]。纵观近年来频发的隐私数据（private data）泄露事件可以发现，其造成的影响无疑是日益严重的。从泄露的数据类型来看[3]，泄露最多的隐私数据是个人基本信息，其次是用户账号密码信息，再者是个体敏感信息。这样的结果不难理解，因为随着医疗系统、交通枢纽、购物网站等流量平台信息化建设及其在商业调查和科学研究的需要，大量用户数据被调查收集或实时抓取，并在一定范围内进行发布和共享，尽管这些数据中的确包含了用户与平台互动的必要合法信息，但也包括了许多用户个人隐私信息，这些隐私信息在数据被发布和共享时，就会不可避免的被公开。如果这些信息被不法分子或别有用心之人加以利用，进行二次贩卖或非法作业，必然会人们的生活带来无尽的麻烦，甚至是更无妄之灾。

但同时，在大数据背景下，通过信息的共享来避免数据孤岛等问题的出现，又是必然的趋势。因此，如何在大数据计算环境下保障数据的隐私性（privacy），同时尽可能不以牺牲数据可用性（utility）和计算的高效性（efficiency）为代价，成为大数据隐私保护领域的研究热点之一。

1.2 有待解决的问题

为满足数据的隐私性和有效性，在其被发布共享之前，需要考虑以下两个方面的因素：一是保证发布的数据中包含用户或信息采集方隐私的信息不会被泄露；二是保证发布数据的有效性[4]以满足实际应用的需求（例如用于商业调查、医疗预警、科学研究等）。由于数据存在异质性的同时，也拥有一定相似性，例如用户的性别、姓名姓氏、电话号码前几位等，所以通过适当遴选，仅隐藏部分信息即可达成上述目标。同时，考虑研究和应用需求，如何使数据得到隐藏的同时，尽可能减少隐藏信息量成了目前亟待解决的问题。

结合上述课题和附件数据集，本文将解决以下问题：

（1）根据附件中的两组二元数据，以最小化隐藏数据量为目标，以有效性为保证，在每个个体的信息都得到有效保护的基础上，计算出各自隐藏数据量最小的方案，并建立一般数学模型进行概括性描述。

（2）将二元数据扩展到多元，即数据信息取值的范围更广，取值可能性更多，进行数据隐藏的状态空间更高的背景下，进行研究分析，并同样给出隐藏数据量最少的方案，建立一般数学模型进行概述。

（3）引入 p-重保护的概念，使个体能够在 $p \geq 2$ 的情况下，隐藏在一个至少包含 p 个个体的数据组中。题目设定$p = 3,5,8$情形的最佳隐藏方案。通过对问题一和问题二的研究，我们决定仍以最小化隐藏的数据量为目标函数，得出最优方案。问题三将分别给出在$p = 3,5,8$情形下，两组二元和两组多元数据集各自的最佳隐藏方案，共 12 种。

（4）问题四针对三种不同的情形：限制隐藏模式、限定全发布数据以及用户定制保护力度，提出了进一步的要求。针对三种不同情形，设计不同的数据隐藏方案。

二．问题重述

本文针对抽象化为数学问题的数据进行研究，并以“需要隐藏的数据用 * 号替换，其余数据则保持不变”的方式统一描述问题。

2.1 问题一重述

问题一提出，现实中绝大部分信息都可以简化为两个不同的选项，例如性别、是否通过考核、是否绿码等。因此，将首先针对附件中给出的两组二元数据，以最小化隐藏数据量为目标，以有效性为保证，给出最优方案和一般数学模型。

附件的两组二元数据集中，行为个体，列为个体信息，因此两组数据集中的个体数据分别具有 6 个、12 个不同的二元信息。在处理完成后要求达到：以每个个体为模板，在数据集中匹配除自身外的其他所有个体，至少存在一个信息值均相同的其他个体数据。

2.2 问题二重述

问题二在问题一的基础上，将个体的信息数据从二元扩展为多元，即个体的信息取值范围扩大。继而进行与问题一相类似的操作，以隐藏数据量最少为目标，不断进行相同个体匹配、数据隐藏的操作，直至所有个体均匹配成功。

2.3 问题三重述

问题三使用问题一、二的数据集，在问题一、二的基础上引入p-重保护的概念，也就是如果个体能够隐藏在一个至少包含p个个体的数据组中（包括自己），则称该个体得到了p-重保护（$p \geq 2$），现用下图举例说明一个有 2-重保护和一个 4-重保护的数据隐藏示例。

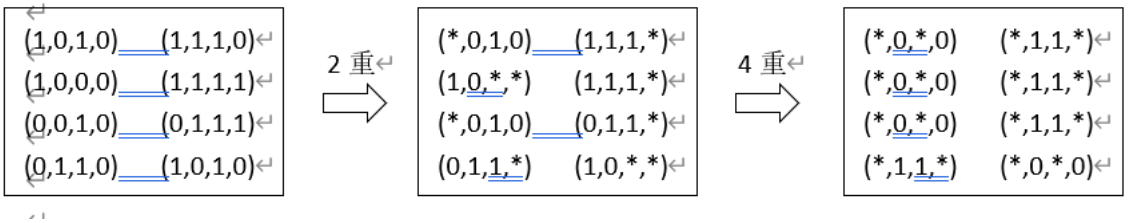


图 2.1 数据隐藏示例

问题三需建立多重保护数据隐藏的数学模型，对附件数据的p-重保护问题进行分析求解，并分别给出$p = 3, 5, 8$情形的最佳隐藏方案。

2.4 问题四重述

问题四在前三问的基础上，引入三种情形：情形一要求考虑对多元数据进行隐藏操作时限定隐藏模式。在隐藏时要求只能进行单隐或全隐；情形二要求从多元高维数据中取出若干维度进行全发布，对其余数据要求达到信息隐藏的效果；情形三要求对每个用户定制不同程度的隐私保护技术，即每个用户可以选择特定的p-重保护要求。对于三个不同的情形，要求给出数据隐藏的方案。

三．基本假设

1. 假设数据不存在异常情况；
2. 假设模型隐藏数据的操作不会出现意外情况；
3. 假设部分文本数据可以抽象为数学问题；
4. 假设隐藏的数据信息位的不同，不会对数据有效性产生较大影响；
5. 假设数据隐藏完成后，不存在被识别的风险。

四．符号说明

符号	说明
x_{ij}	表示处理后第i个个体的第j个信息隐藏情况
f	表示处理整个数据集需要隐藏的信息个数
φ_i	表示处理后是否存在其他个体与个体i相同
m	表示数据集中个体总数
n	表示数据集中每个个体拥有的信息个数
l	表示某次配对操作
T_l	表示某次配对操作的指标
R_l	表示进行配对操作l，从单独个体变为配对成功的个体的数量
U_l	表示进行配对操作l，需要隐藏的信息位数。
A_l	表示配对操作l中两个个体信息不同的位数
B_l	表示操作 l 中要进行隐藏操作的所有数据条的数据值中已经有的*的个数
D_l	表示操作 l 处理的两个群的总个体数
c	表示具体的多元数据的元数
P_i	表示第 i 行数据与其他 $m-1$行数据的不同信息位的个数的集合
S_i	表示集合P_i中所有非零元素组成的集合
V_l	表示配对操作 l 涉及的两个群体中，未满足 p-重保护要求的群的大小之和

注：其它符号将在文中具体说明。

五．第一问模型的建立与求解

5.1 二元数据隐藏模型的建立

5.1.1 模型的准备与思路

问题一的模型目标是最小化隐藏的数据量，并保障每个个体的信息都得到保护。故而需要统计隐藏数据的个数、保证每个个体在数据隐藏结束时，均有除自身外的至少一个个体与之完全匹配。

5.1.2 模型的建立

针对问题一的两组二元数据集进行分析可得，个体的信息仅有 0，1 两种可能取值，目标函数是对隐藏数据量的最小化。因此，需要设置决策变量以便于统计隐藏数据量，同时设置指标，使每一轮匹配操作都执行改变数据量最小的隐藏操作（本文将隐藏操作可视化为原数据变更为 $*$），直至满足约束条件。最终统计总隐藏数据量，即为问题所要求的。

首先，定义个体 i 的第 j 位信息是否进行了隐藏操作，即信息是否被标为 $*$ 号为问题一的决策变量，表示为x_{ij}。对于二元数据集，使用数字 2 标识进行了隐藏操作。当x_{ij}取 0 或 1 时，表示个体 i 的第 j 位信息保持原始值，无需隐藏操作；当x_{ij}取 2 时，表示个体 i 的第 j 位信息被隐藏为$*$。上述可总结为如下公式：

$$x_{ij}=\begin{cases}0,\ \text{个体 } i \text{ 的第}j\text{位信息未隐藏，且原始值为 } 0\\ 1,\ \text{个体 } i \text{ 的第}j\text{位信息未隐藏，且原始值为 } 1\\ 2,\ \text{个体 } i \text{ 的第}j\text{位信息被隐藏为 } *\end{cases} \tag{5-1}$$

根据上述决策变量的取值，我们首先通过计算 $x_{ij}\left(x_{ij}-1\right)$ 的值，来判断个体 i 的第 j 位信息是否发生改变。当 $x_{ij}=0$ 或 1时，该式的值均等于 0，不会对接下来的累加操作产生影响；当 $x_{ij}=2$时，该式的值等于 2。$\sum_{j=1}^{n}x_{ij}\left(x_{ij}-1\right)$ 的作用是统计个体 i 的所有信息位中进行了隐藏操作数量的两倍。$\sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}\left(x_{ij}-1\right)$ 是统计所有 m 条数据的全部 n 位信息中进行了隐藏操作数量的两倍。

我们设 f 为需要变为 $*$ 号进行隐藏的数据信息个数，则$f=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}\left(x_{ij}-1\right)$，目标函数可以表示为：

$$min\, f = min\ \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}(x_{ij}-1) \tag{5-2}$$

由于式（5-2）中的$1/2$不会对优化结果产生影响，故可以将目标函数简化为：

$$min\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}(x_{ij}-1) \tag{5-3}$$

为保证个体均得到有效隐藏，即每个个体均有除自身外与之完全匹配的其他个体，需设置约束条件加以判断。对于两个个体，首先通过计算$(x_{ij}-x_{kj})$来判断他们各个对应位信息是否相同，若结果为0，则表示个体i与个体k的第j位信息相同，进一步计算$\sum_{j=1}^{n}|x_{ij}-x_{kj}|$的值，若结果仍为0，则表示个体$i$与个体$k$的每一位信息均相同，也即个体$i$与个体$k$成功完成了数据隐藏。若存在$j$使得$(x_{ij}-x_{kj})$的计算结果不为0，则表示个体$i$与个体$k$的第$j$位信息不相同，在求和后$\sum_{j=1}^{n}|x_{ij}-x_{kj}|$一定大于0。

将个体i与数据集中其余的$m-1$个个体分别进行上述操作，然后进行累乘计算，即$\prod_{k=1,k\neq i}^{m}(\sum_{j=1}^{n}|x_{ij}-x_{kj}|)$。若乘积为0，则表示个体$i$能够在其余的$m-1$个个体中，匹配到与其自身信息完全一致的个体，从而达成隐藏数据的目的；若乘积不为0，则表示在其余的$m-1$个个体中，没有找到能够与个体i的信息完全匹配的个体，即个体i尚未达成数据隐藏的目标。

设个体维度为n，数量为m，则约束条件可表示为如下公式：

$$\varphi_i=\prod_{k=1,k\neq i}^{m}\left(\sum_{j=1}^{n}|x_{ij}-x_{kj}|\right)=0,\qquad i=1,2,\cdots,n \tag{5-3}$$

其中，φ_i用于表示个体i能否找到一个其他个体与自己相同，若$\varphi_i=0$，则表示个体i能找到一个其他个体与自己相同；否则表示个体i不能找到一个其他个体与自己相同。

综上所述，问题一的二元数据隐藏模型可表示如下：

$$min\, f = min\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}(x_{ij}-1)$$

$$\begin{cases} x_{ij}=\begin{cases}0,\ \text{个体}\ i\ \text{的第}j\text{位信息未隐藏，且原始值为}\ 0\\ 1,\ \text{个体}\ i\ \text{的第}j\text{位信息未隐藏，且原始值为}\ 1\\ 2,\ \text{个体}\ i\ \text{的第}j\text{位信息被隐藏为}\ *\end{cases}\\ \varphi_i=\prod_{k=1,k\neq i}^{m}\left(\sum_{j=1}^{n}|x_{ij}-x_{kj}|\right)=0,\qquad i=1,2,\cdots,n \end{cases} \tag{5-4}$$

5.1.3 模型的求解

本题进行数据隐藏的状态空间高达2^{nm}，在这样的状态下，即使面对很小规模的数据也难以在有限时间内得出满意解。因此，我们将基于数据分群的思想，对群之间的配对设定评价指标，每次使用贪心算法来选择两个评价指标最大群进行配对操作，并最终得到局部最优解。贪心算法的核心思想就是用局部最优解去逼近全局最优解，在寻优过程中总是做出当前情况下的最优选择。

评价指标

本文提出对两类数据进行匹配的配对操作，两类数据在配对完成后形成完全相同的新一类数据。例如，对于数据 001 和数据 000，我们配对的到数据 00*。我们针对配对操作设立统一的评价指标，该指标能够一定程度上判定配对操作的优劣性。因此，在每次执行配对操作前，首先要计算不同配对操作的指标，从而选取指标最优的配对操作。我们将评价指标设定为$T_l = R_l/U_l$，其中，l表示某次配对操作，R_l表示配对操作l中，从单独个体变为配对成功的个体的数量，U_l表示配对操作l中需要隐藏的信息位数。现对该指标进行举例描述如下：

表 5-1 评价指标案例（二元数据）

数据名称	二元数据
记录 1	001
记录 2	000
记录 3	111

对表 5-1 中数据进行指标计算，得到结果如下表 5-2 所示：

表 5-2 评价指标案例计算结果（二元数据）

二元数据	001	000	111
001	\	1	1/2
000	1	\	1/3
111	1/2	1/3	\

取表 5-2 中数据 001 与数据 000 的指标结果进行解释：两个数据的指标计算公式为$2/2 = 1$。其中，分子的 2 表示数据 001 和数据 000 两个单独个体，在完成配对后均会变为配对成功个体；分母的 2 表示该配对操作中需要进行隐藏的信息数为 2，即数据 001 中第三位的“1”和数据 000 中第三位的“0”。通过指标计算，我们可以选取数据 000 和数据 001 进行匹配和隐藏操作，以保证操作在我们设定的指标下“最优”。

匹配策略分析

我们引入群的概念，即尚未匹配成功的单个个体、配对成功并聚合在一起的所有个体均称为群，之后的模型中将延用此概念。

由于策略分类较多，我们将引入如下表 5-3 所示的数据，通过具体匹配运算示例来解释不同分类策略的操作过程，并同时加入评价指标计算。

表 5-3 评价指标案例（二元数据）

数据名称	二元数据	群内个体数
多人群 1	00***	3
双人群 2	010**	2
个体群 1	01011	1
个体群 2	10100	1
个体群 3	10110	1
个体群 4	00111	1

我们将匹配策略划分为如下三大类：

（1）第一类策略：尚未匹配成功的单个个体之间进行匹配。该类匹配方案将两个单个个体进行匹配操作。例如表 5-3 中的个体群 2 和个体群 3，计算评价指标为：$2/2 = 1$。一般地，对于两个单个个体群，我们的评价指标计算为：$T_l = 2/(2 \cdot A_l)$，其中A_l表示操作l中两个个体数据值不同的位数。

（2）第二类策略：单个个体与配对成功的多个体群进行匹配。该类又可以细分为单个个体与仅包含两个个体的群进行匹配、单个个体与包含超过两个个体的群进行匹配两小类策略。下面针对这两小类策略进行分析：

① 单个个体与仅包含两个个体的群进行匹配。该策略又可以分为如下两种情况：

1) 情况一是单个个体与从双人群中取出的某个个体进行匹配，与之组成一个新的群，但同时双人群中的另一个个体会从匹配状态变为个体状态，没有起到匹配隐藏的目的，故在本题中不予考虑；
2) 情况二是单个个体加入双人群中，形成包含三个个体的新群。情况二又可继续细化为如下两种情况：
   i. 一是双人群中个体的信息不会发生改动，仅需对单个个体进行部分信息隐藏后即可完成匹配。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(1 \cdot A_l)$。
   ii. 二是个体加入双人群中，双人群中个体与该单个个体均需改变部分信息。这种情况需进行改变的数据信息较多，不是我们最优先考虑的情况。例如表 5-3 中个体群 2 与群 2，计算评价指标为：$1/11$，即个体群 2 与群 2 均需要将其不为 $*$ 的信息均改为 $*$。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(3 \cdot A_l - B_l)$，其中$B_l$表示操作 l 中要隐藏的所有数据条的数据值中已经有的*的个数。

② 单个个体与包含超过两个个体的群进行匹配。该策略也需要分如下情况讨论：

1) 情况一是单个个体直接加入多人群中，例如表 5-3 中个体群 1 加入群 1，计算评价指标为：$1/7$；个体群 4 加入群 1，评价指标为：$1/3$。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(D_l \cdot A_l - B_l)$，其中$D_l$表示操作 l 处理的两个群的总个体数。

2) 情况二是个体从群中抽取一个个体，与之匹配后形成新的群，利用 1）中相同数据，可求得评价指标分别为：$1/5$、$1/3$。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(2 \cdot A_l - B_l)$。

（3）第三类策略：配对成功的多个体群之间进行匹配。结合第一问的需求，此时群中的个体数均大于 2，已满足题目要求，无需继续进行额外的数据隐藏。因此在二重保护的二元数据隐藏中不予考虑。

算法步骤

Step1：读取附件中二元数据集，并对数据集中的全部数据进行初始分群操作，即数据中全部信息位均相同的归入同一群体，不完全相同的数据各自单独成群；

Step2：计算群与群之间的评价指标。根据群与群之间的具体情况，选择上述策略中的一种，并根据对应的指标公式计算两个群体间进行配对操作的指标。

Step3：选取 Step2 中评价指标结果最优的两个群体进行匹配操作。

Step4：判断所有群体是否满足题目的隐藏数据要求，若不满足，则跳转回 Step2，继续计算和配对操作；若满足，则跳转至 Step5。

Step5；输出当前方案。

根据上述算法步骤，绘制如下群与群之间的匹配过程图 5.1：

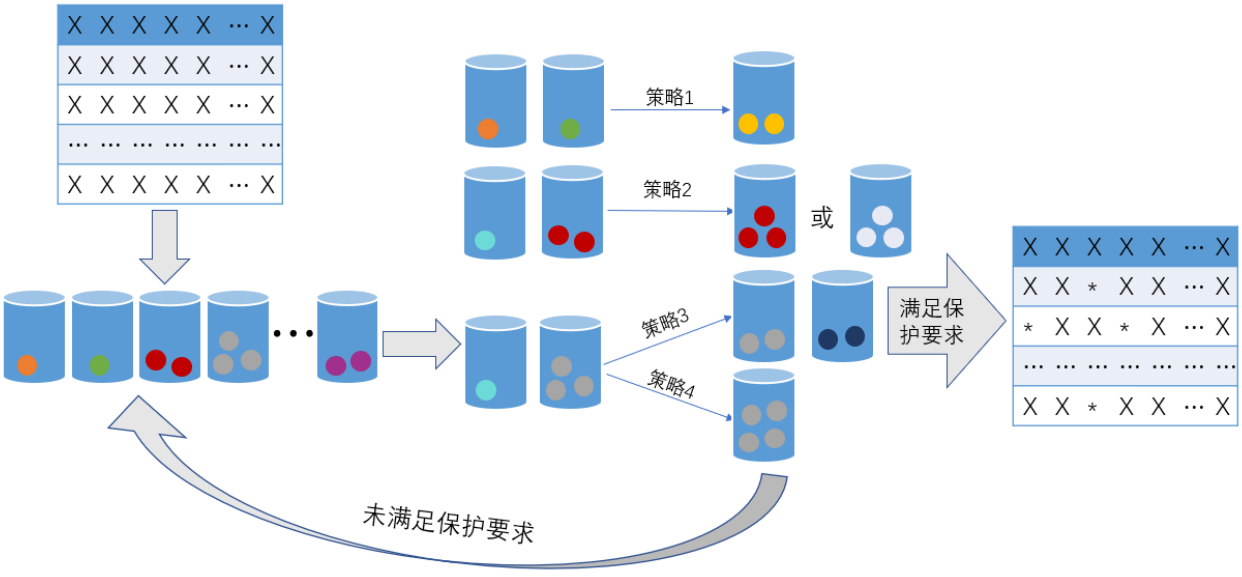
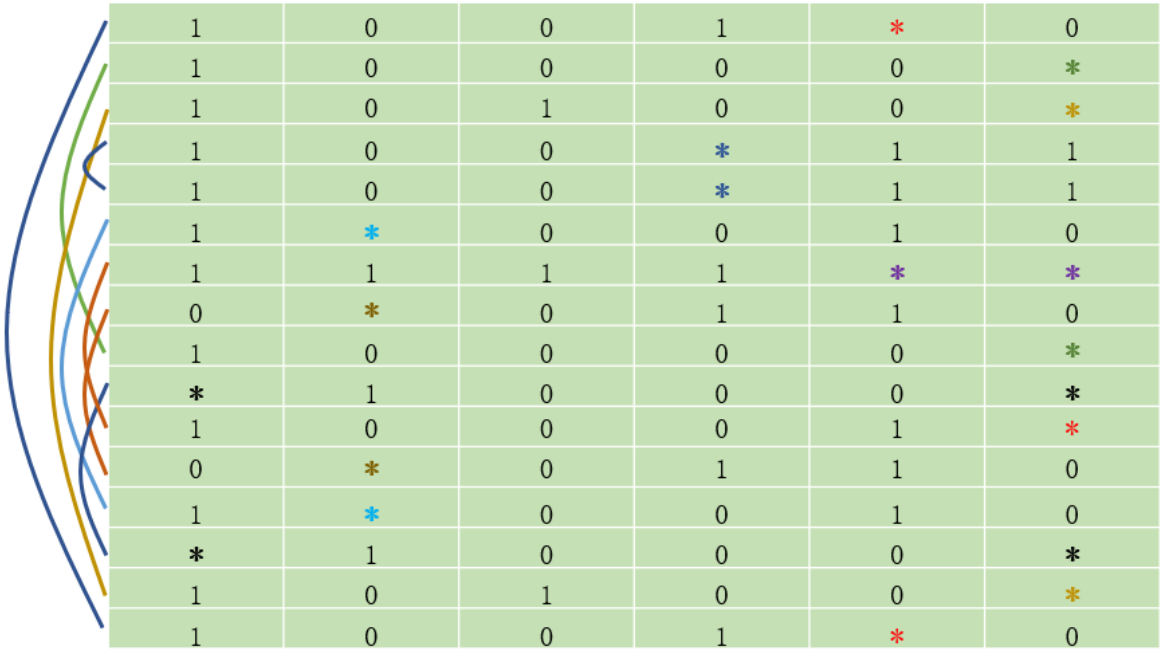


图 5.1 二元群匹配过程图

5.2 二元数据隐藏模型的结果呈现

我们首先使用上述算法对二元 1 数据进行求解，最终得到操作方案见附件。二元 1 数据共含数据 80 条，最终我们一共隐藏了 20 个数据信息。由于数据量大，难以在论文中全部呈现，且结果方案中仅对 16 条数据进行了隐藏操作，其余数据均保持原始数据不变，故我们提炼出这 16 条数据进行如下图 5.2 所示：



1	0	0	1	*	0
1	0	0	0	0	*
1	0	1	0	0	*
1	0	0	*	1	1
1	0	0	*	1	1
1	*	0	0	1	0
1	1	1	1	*	*
0	*	0	1	1	0
1	0	0	0	0	*
*	1	0	0	0	*
1	0	0	0	1	*
0	*	0	1	1	0
1	*	0	0	1	0
*	1	0	0	0	*
1	0	1	0	0	*
1	0	0	1	*	0

图 5.2 二元 1 方案

对于二元 2 数据，我们同样采用上述算法进行求解，最终得到操作方案见附件。二元 2 数据共含数据 200 条，最终我们一共隐藏了 361 个数据信息。对二元 1 和二元 2 数据的最终处理结果如下表 5-4 所示。

表 5-4 二元数据处理结果

数据名称	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
二元 1	80*6	2	40	8 次策略 1	20
二元 2	200*12	2 3	92 2	94 次策略 1， 2 次策略 2	361

处理结果显示，对于二元 1 和二元 2 中的数据，本文的算法主要使用策略 1，少部分情况下使用了策略 2。这是因为处理后每个个体只需要找到一个其他个体与自己相同即可。而策略 1 正好能将两个单独个体匹配成一个双人群。而策略 3 未被使用的原因是，一般情况下一个单独个体会选择策略 2，与一个双人群匹配而非使用策略 3 与一个多人群（对于同一单独个体，选择策略 3 需要的隐藏个数通常高于策略 2）。

5.3 算法评估

我们对本文提出的基于分群思想和贪心算法的处理策略进行性能分析。由于在最坏情况下，每轮配对操作结束可能只有一个个体完成了匹配和隐藏，而匹配的流程中，复杂度最高的操作是求群之间的指标，该操作的复杂度为 $m^2 \cdot n$，则配合 m 个个体总数，得出算法总时间复杂度为$O(m^3 \cdot n)$。

我们对本文提出的算法进行了时间性能的实验分析，具体结果如表 5-5 所示。根据表 5-5 可以得出，我们的算法可以在短时间内针对二元数据集高效地求出结果。

表 5-5 二元数据算法评估

	二元 1 数据维度	二元 2 数据维度
m	80	200
n	6	12
时间复杂度	运行时间(s)	
$O(m^3 \cdot n)$	0.03s	6.6s

其中，每组数据的运行时间均为运行 100 次后取平均值得到。

六．第二问模型的建立与求解

6.1 多元数据隐藏模型的建立

6.1.1 模型的准备与思路

问题二的模型目标同样为最小化隐藏的数据量，并保障每个个体的信息都得到保护。与第一问的主要区别就在于，问题二使用的数据是多元的，即数据信息取值的范围更广，取值可能性更多，进行数据隐藏的困难度更高。

6.1.2 模型的建立

针对问题二的两组多元数据集进行分析可得，个体的信息取值范围为 $[0, c-1]$，其中 c 表示具体的多元数据的元数，目标函数是最小化隐藏数据量。

首先，类似于问题一中的决策变量，我们用x_{ij}定义个体 i 的第 j 位信息是否进行了隐藏操作，即信息是否被标为 $*$ 号。由于 $*$ 在程序中不便于计算，故使用数字 c 对其标识，使用 c是为了保证该数值不与个体信息的取值冲突。

当x_{ij}在区间 $[0, c-1]$ 中取值时，表示个体 i 的第 j 位信息保持原始值，无需隐藏操作；当x_{ij}取 c 时，表示个体 i 的第 j 位信息被隐藏为$*$。上述可总结为如下公式：

$$x_{ij}=\begin{cases} 0, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为 } 0 \\ 1, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为 } 1 \\ \cdots\cdots \\ c-1, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为} c-1 \\ c, \text{ 个体 } i \text{ 的第} j \text{位信息被隐藏为 } * \end{cases} \tag{6-1}$$

第二问中数据信息位的取值范围较第一问有所扩充，即通过 $\prod_{t=0}^{c-1}(x_{ij}-t)$ 来判断个体 i 的第 j 位信息是否发生改变。根据上述决策变量的取值，我们首先通过计算 $\prod_{t=0}^{c-1}(x_{ij}-t)$ 的值，来判断个体 i 的第 j 位信息是否发生改变。当 $x_{ij}\in[0, c-1]$时，该式的值均等于 0，不会对接下来的累加操作产生影响；当 $x_{ij}=c$时，该式的值等于$c!$。$\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$ 的作用是统计个体 i 的所有信息位中进行了隐藏操作数量的$c!$倍。$\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$ 是统计所有 m 条数据的全部 n 位信息中进行了隐藏操作数量的$c!$倍。

设 f 为需要变为 $*$ 号进行隐藏的数据个数，即$f=\frac{1}{c!}\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$，则目标函数可以表示为：

$$minf = \min \frac{1}{c!}\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t) \tag{6-2}$$

由于式（6-2）中的$1/c!$不会对优化结果产生影响，故可以将目标函数简化为：

$$min\sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}(x_{ij}-1) \tag{6-3}$$

为保证个体均得到有效隐藏，即每个个体均有除自身外与之完全匹配的个体，需设置约束条件加以判断。与问题一中的约束条件相同，问题二也采用按信息位求差再累加，再将每一次累加结果进行累乘操作，得到最终解。若结果为 0，则表示至少存在一个与该个体完全匹配的个体；若结果不为 0，则需要进行数据隐藏，而后继续匹配。

设个体维度为 n，数量为 m，则约束条件可表示为如下公式：

$$\varphi_i = \prod_{k=1,k\neq i}^{m}\left(\sum_{j=1}^{n}|x_{ij}-x_{kj}|\right)=0, \qquad i=1,2,\cdots,n \tag{6-4}$$

遍历数据集中的所有个体，延用群的概念，将完全匹配的个体划归为同一群体，没有匹配项的个体则单独成群。

综上所述，问题二的多元数据隐藏模型可表示如下：

$$\min \quad c!\cdot f = min\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$$

$$\begin{cases} x_{ij}=\begin{cases} 0, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为 } 0 \\ 1, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为 } 1 \\ \cdots\cdots \\ c-1, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为} c-1 \\ c, \text{ 个体 } i \text{ 的第} j \text{位信息被隐藏为 } * \end{cases} \\ \varphi_i = \prod_{k=1,k\neq i}^{m}\left(\sum_{j=1}^{n}|x_{ij}-x_{kj}|\right)=0, \qquad i=1,2,\cdots,n \end{cases} \tag{6-5}$$

6.1.3 模型的求解

由于问题二是在问题一的基础上，对个体信息取值范围进行了扩展，所以将参考问题一的模型求解思路进行模型设计。本问仍采用贪心算法、设置评价指标，并针对具体匹配策略进行具体分析。

评价指标

问题二使用与问题一相同的评价指标，即 $T_l = R_l/U_l$，来判定每次隐藏操作的优劣性，以保证操作在我们设定的指标下“最优”。

匹配策略分析

由于策略分类较多，我们将引入如下表所示的数据，通过具体匹配运算示例来解释不同分类策略的操作过程，并同时加入评价指标计算。

表 6-1 评价指标案例（多元数据）

数据名称	二元数据	群内个体数
群 1	02***	3
群 2	310**	2
个体群 1	31031	1
个体群 2	02000	1
个体群 3	02010	1
个体群 4	00231	1

假设表 6-1 中均为四元数据。

我们仍将匹配策略按照如下三大类进行划分：

（1）第一类策略：尚未匹配成功的单个个体之间进行匹配。该类匹配方案将两个单个个体进行匹配操作。例如表 6-1 中的个体群 2 和个体群 3，计算评价指标为：$2/2 = 1$，即仅需隐藏个体群 2 第四位的“0”和个体群 3 第四位的“1”。一般地，对于两个单个个体群，我们的评价指标计算为：$T_l = 2/(2 \cdot A_l)$，其中A_l表示操作l中两个个体数据值不同的位数。

（2）第二类策略：单个个体与配对成功的多个体群进行匹配。对该类进一步划分为如下两小类策略进行分析：

① 单个个体与仅包含两个个体的群进行匹配。该策略又可以分为如下两种情况：

1) 情况一是单个个体从双人群中取出某个个体进行匹配。同问题一中解释，该情况在本题中不予考虑；
2) 情况二是单个个体加入双人群中，形成包含三个个体的新群。此类情况又可继续细化为如下两种：

i. 一是群中个体的信息不会发生改动，仅需对单个个体进行部分信息隐藏后即可完成匹配。例如表 6-1 中的个体群 1 仅改变自身的最后两位信息，便可成功与群 2 匹配，此时评价指标为：$1/2$；一是双人群中个体的信息不会发生改动，仅需对单个个体进行部分信息隐藏后即可完成匹配。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(1 \cdot A_l)$。

ii. 二是个体加入双人群中，群中个体与该单个个体均需改变部分信息。由

于问题二扩大了数据信息的取值范围，故需要隐藏的数据信息将变得更多，因此该方法根据指标计算结果通常不是我们最优先考虑的策略。例如表 6-1 中个体群 3 想要与群 2 匹配，双方都需要对自身的信息位进行更改，此时计算评价指标为：$1/11$。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(3 \cdot A_l - B_l)$，其中$B_l$表示操作 l 中要隐藏的所有数据条的数据值中已经有的*的个数。

② 单个个体与包含超过两个个体的群进行匹配。该策略也需要分如下情况讨论：

1) 情况一是单个个体直接加入多人群中，例如表 6-1 中个体群 4 加入群 1，计算评价指标为：$1/7$；个体群 3 加入群 1，评价指标为：$1/3$。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(D_l \cdot A_l - B_l)$，其中$D_l$表示操作 l 处理的两个群的总个体数。
2) 情况二是个体从多人群中抽取一个个体，与之匹配后形成新的群，利用 1）中相同数据，可求得评价指标分别为：$1/5$、$1/3$。一般地，对于该情况，我们的评价指标计算为：$T_l = 1/(2 \cdot A_l - B_l)$。

（3）第三类策略：配对成功的多个体群之间进行匹配。结合第一问的需求，此时群中的个体数均大于 2，已满足题目要求，无需继续进行额外的数据隐藏。因此在二重保护的多元数据隐藏中不予考虑。

算法步骤

Step1：读取附件中多元数据集，并对数据集中的全部数据进行分群操作，即数据中全部信息位均相同的归入同一群体，不完全相同的数据各自单独成群；

Step2：计算群与群之间的评价指标。根据群与群之间的具体情况，选择上述策略中的一种，并根据对应的指标公式计算两个群体间进行配对操作的指标。

Step3：选取 Step2 中评价指标结果最优的两个群体进行匹配操作。

Step4：判断所有群体是否满足题目的隐藏数据要求，若不满足，则跳转回 Step2，继续计算和配对操作；若满足，则跳转至 Step5。

Step5：输出当前方案。

根据上述算法步骤，绘制如下群与群之间的匹配过程图 6.1：

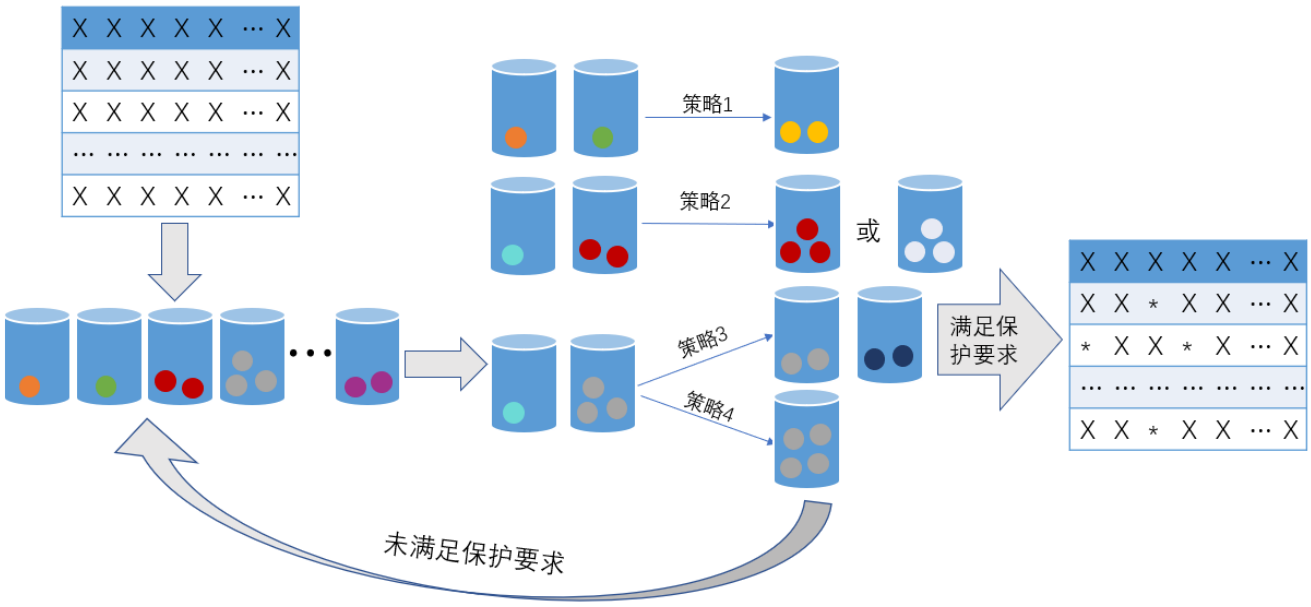


图 6.1 多元群匹配过程图

6.2 多元数据隐藏模型的结果呈现

我们使用上述算法对多元 1 数据进行求解，最终得到操作方案见附件。多元 1 数据共含数据 200 条，每条包含 8 个数据信息。最终，我们一共隐藏了 634 个数据信息。

对于多元 2 数据，我们同样采用上述算法进行求解，最终得到操作方案见附件。多元 2 数据共含数据 800 条，每条包含 16 个数据信息。最终我们一共隐藏了 7978 个数据信息。对多元 1 和多元 2 数据的最终处理结果如下表 6-2 所示。

表 6-2 多元数据处理结果

数据名称	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
多元 1（元=5）	200*8	2 3	97 2	97 个策略 1，2 个策略 2	634
多元 2（元=10）	800*16	2	400	400 个策略 1	7978

处理结果显示，对于二元 1 和二元 2 中的数据，本文的算法主要使用策略 1，少部分情况下使用了策略 2。这是因为处理后每个个体只需要找到一个其他个体与自己相同即可。而策略 1 正好能将两个单独个体匹配成一个双人群。而策略 3 未被使用的原因是，一般情况下一个单独个体会选择策略 2，与一个双人群匹配而非使用策略 3 与一个多人群（对于同一单独个体，选择策略 3 需要的隐藏个数通常高于策略 2）。

6.3 算法评估

针对多元数据，我们对本文提出的算法进行了时间性能的实验分析，具体结果如表6-3 所示。根据表 6-3 可以得出，对于多元 1 数据我们的算法可以在短时间内针对二元数据集高效地求出结果；对于规模较大的多元 2 数据，我们的算法也能在一定时间内得出结果。

表 6-3 多元数据算法评估

	多元 1 数据维度	多元 2 数据维度
m	200	800
n	8	16
时间复杂度	运行时间(s)	
$O(m^3 \cdot n)$	6.3s	689s

其中，多元 1 数据的运行时间为运行 100 次后取平均值得到，多元 1 数据的运行时间为运行 10 次后取平均值得到。

将多元数据下的运行时间与二元数据的运行时间进行对比发现，对于$m = 200$，$n = 12$的二元数据，平均运行时间为 6.6s，而$m = 200$，$n = 8$的多元数据平均运行时间为 6.3s。可以说明我们提出的算法运行时间不会随着数据取值范围的扩大而明显增长。

七．第三问模型的建立与求解

7.1 多元数据隐藏模型的建立

7.1.1 模型的准备与思路

问题三在一二问的基础上，加入 p-重保护的概念，即当 $p \geq 2$ 时，个体能够隐藏在一个至少包含 p 个个体的数据组中。题目要求给出$p = 3,5,8$情形的最佳隐藏方案。通过对问题一和问题二的研究，我们决定仍以最小化隐藏的数据量为目标函数，得出最优方案。

问题将分别要求给出在$p = 3,5,8$情形下，两组二元和两组多元数据集各自的最佳隐藏方案，共 12 种。

7.1.2 模型的建立

在问题一和问题二中，我们仅需保证存在除自身外的一个个体，与之完全匹配即可满足要求，置于问题三的背景下，可以说前两问仅需满足 2-重保护即可。而问题三则对数据保护的多重性提出要求，因此我们需要对与数据相匹配的数据量进行统计，引用群的概念即为，统计群众包含的个体个数。

针对问题三的四组数据集进行统一性建模，设个体的信息取值范围为 $[0, c-1]$，其中 c 表示具体的数据元数，目标函数则是最小化隐藏数据量。

首先，类似于问题一二中的决策变量，我们用x_{ij}定义个体 i 的第 j 位信息是否进行了隐藏操作，即信息是否被标为 $*$ 号。由于 $*$ 在程序中不便于计算，故使用数字 c 对其标识，使用 c是为了保证该数值不与个体信息的取值冲突。

当x_{ij}在区间 $[0, c-1]$ 中取值时，表示个体 i 的第 j 位信息保持原始值，无需隐藏操作；当x_{ij}取 c 时，表示个体 i 的第 j 位信息被隐藏为$*$。上述可总结为如下公式：

$$x_{ij} = \begin{cases} 0, \text{ 个体 } i \text{ 的第}j\text{位信息未隐藏，且原始值为 } 0 \\ 1, \text{ 个体 } i \text{ 的第}j\text{位信息未隐藏，且原始值为 } 1 \\ \cdots\cdots \\ c-1, \text{ 个体 } i \text{ 的第}j\text{位信息未隐藏，且原始值为} c-1 \\ c, \text{ 个体 } i \text{ 的第}j\text{位信息被隐藏为 } * \end{cases} \tag{7-1}$$

与第二问相类似，我们通过 $\prod_{t=0}^{c-1}(x_{ij}-t)$ 来判断个体 i 的第 j 位信息是否发生改变。根据上述决策变量的取值，我们首先通过计算 $\prod_{t=0}^{c-1}(x_{ij}-t)$ 的值，来判断个体 i 的第 j 位信息是否发生改变。当 $x_{ij} \in [0, c-1]$时，该式的值均等于 0，不会对接下来的累加操作

产生影响；当 $x_{ij}=c$时，该式的值等于$c!$。$\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$ 的作用是统计个体 i 的所有信息位中进行了隐藏操作数量的$c!$倍。$\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$ 是统计所有 m 条数据的全部 n 位信息中进行了隐藏操作数量的$c!$倍。

设 f 为需要变为 $*$ 号进行隐藏的数据个数，即$f=\frac{1}{c!}\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$，则目标函数可以表示为：

$$minf=\min\frac{1}{c!}\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t) \tag{7-2}$$

由于式（7-2）中的$1/c!$不会对优化结果产生影响，故可以将目标函数简化为：

$$min\,f=min\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t) \tag{7-3}$$

为保证个体均得到有效隐藏，问题三首先设置与问题二相同的约束条件如下：

$$\varphi_i=\prod_{k=1,k\neq i}^{m}\left(\sum_{j=1}^{n}|x_{ij}-x_{kj}|\right)=0,\qquad i=1,2,\cdots,n \tag{7-4}$$

在此基础上，问题三还需满足p-重保护的概念。在此，我们定义集合$P_i=\left\{\sum_{j=1}^{n}|x_{ij}-x_{kj}|\,|k=1,2,\cdots,i-1,i+1,\cdots,m\right\}$，其中$i=1,2,\cdots,m$。该集合 P_i 表示第 i 行数据与其他 $m-1$行数据的不同信息位的个数。继续定义集合$S_i\subseteq P_i$，对$s_{ir}\in S_i,r=1,2,\cdots,|S_i|$，满足条件$\sum_{k=1,k\neq i}^{m}p_{ik}=\sum_{r=1}^{|S_i|}s_{ir}$，且$\prod_{r=1}^{|S_i|}s_{ir}\neq 0$。条件$\sum_{k=1,k\neq i}^{m}P_{ik}=\sum_{r=1}^{|S_i|}s_{ir}$保证集合$P_i$中的所有非零元素都存在于$S_i$中；条件$\prod_{r=1}^{|S_i|}s_{ir}\neq 0$保证集合$P_i$中的所有零元素都不存在于$S_i$中。那么，只需要计算$|P_i|-|S_i|$，就可以得到与个体$i$相同的其他个体的数量。因此，条件$|P_i|-|S_i|\geq p-1$就可以保证集合 P_i 满足至少包含 $p-1$ 个0元素。若对于每一个$i=1,2,\cdots,m$，该条件都满足，则表示处理后的数据满足p-重保护（$p\geq 2$）的要求。

该约束条件可表示如下：

$$|P_i|-|S_i|\geq p-1,\qquad \sum_{k=1,k\neq i}^{m}P_{ik}=\sum_{r=1}^{|S_i|}s_{ir},\prod_{r=1}^{|S_i|}s_{ir}\neq 0 \tag{7-5}$$

综上所述，问题三的多元数据隐藏模型可表示如下：

$$minf = min\sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{t=0}^{c-1}(x_{ij}-t)$$

$$\begin{cases} x_{ij} = \begin{cases} 0, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为 } 0 \\ 1, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为 } 1 \\ \cdots\cdots \\ c-1, \text{ 个体 } i \text{ 的第} j \text{位信息未隐藏，且原始值为} c-1 \\ c, \text{ 个体 } i \text{ 的第} j \text{位信息被隐藏为 } * \end{cases} \\ \varphi_i = \prod_{k=1,k\neq i}^{m}\left(\sum_{j=1}^{n}|x_{ij}-x_{kj}|\right) = 0, \quad i = 1,2,\cdots,m \\ \begin{cases} |P_i| - |S_i| \geq p-1 \\ \sum_{k=1,k\neq i}^{m} P_{ik} = \sum_{r=1}^{|S_i|} s_{ir} \\ \prod_{r=1}^{|S_i|} s_{ir} \neq 0 \end{cases}, \quad i = 1,2,\cdots,m \end{cases} \tag{7-6}$$

7.1.3 模型的求解

评价指标

对于 p-重保护，我们使用评价指标 $T_l = V_l/U_l$。其中，l 表示某次配对操作，V_l表示配对操作 l 涉及到的群体中，未满足 p-重保护要求的群的大小之和（可能为一个群的大小，也可能为两个群的大小之和），U_l表示配对操作 l 需要隐藏的信息位数。我们将对 $p = 4$的情况进行举例描述如下：

表 7-1 评价指标案例（多元数据）

数据名称	多元数据	群内个体数
群 1	003	2
群 2	01 *	3
群 3	000	4

对表 7-1 中数据进行指标计算，得到如下结果：

表 7-2 评价指标案例计算结果（多元数据）

二元数据	2 × (003)	3 × (01 *)	4 × (000)
2 × (003)	\	5/7	2/12
3 × (01 *)	5/7	\	3/18
4 × (000)	2/12	3/18	\

取表 7-2 中数据$2\times(003)$与数据$3\times(01*)$的指标结果进行解释：两个数据的指标计算公式为$5/7$。其中，分子的 5 表示数据$2\times(003)$和数据$3\times(01*)$ 两个群，在完成配对后均会变为配对成功群；分母的 7 表示该配对操作中需要进行隐藏的信息数为 7，即数据$2\times(003)$中第二、三位“0”、“3”和数据$3\times(01*)$中第二位的“1”。通过指标计算，我们可以选取数据$2\times(003)$和数据$3\times(01*)$进行匹配和隐藏操作，以保证操作在我们设定的指标下“最优”。

匹配策略分析

由于策略分类较多，我们将引入如下表所示的数据，通过具体匹配运算示例来解释不同分类策略的操作过程，并同时加入评价指标计算。

我们仍将匹配策略按照如下三大类进行划分：

（1）第一类策略：尚未匹配成功的单个个体之间进行匹配。该类匹配方案将两个单个个体进行匹配操作。一般地，对于两个单个个体群，我们的评价指标计算为：$T_l=2/(2\cdot A_l)$，其中A_l表示操作l中两个个体数据值不同的位数。

（2）第二类策略：单个个体与包含p个及以上个体的群进行匹配。将单个个体的通过隐藏数据信息等操作后，加入群中。一般地，对于该情况，我们的评价指标计算为：$T_l=1/(D_l\cdot A_l-B_l)$，其中D_l表示操作 l 处理的两个群的总个体数。

（3）第三类策略：单个个体与包含p个以下个体的群进行匹配。将单个个体的通过隐藏数据信息等操作后，加入群中。一般地，对于该情况，我们的评价指标计算为：$T_l=D_l/(D_l\cdot A_l-B_l)$，其中$D_l$表示操作 l 处理的两个群的总个体数。

（4）第四类策略：配对成功的多个体群之间进行匹配。无论群的规模大小，均执行合并操作，生成新的群。

算法步骤

Step1：设定保护重数p。

Step2：读取附件中多元数据集，并对数据集中的全部数据进行分群操作，即数据中全部信息位均相同的归入同一群体，不完全相同的数据各自单独成群；

Step3：计算群与群之间的评价指标。根据群与群之间的具体情况，选择上述策略中的一种，并根据对应的指标公式计算两个群体间进行配对操作的指标。

Step4：选取 Step2 中评价指标结果最优的两个群体进行匹配操作。

Step5：判断所有群体是否满足题目的隐藏数据要求和设定的保护重数，若不满足，则跳转回 Step2，继续计算和配对操作；若满足，则跳转至 Step6。

Step6：输出当前方案。

根据上述算法步骤，群与群之间的匹配过程如图 7-1 所示：

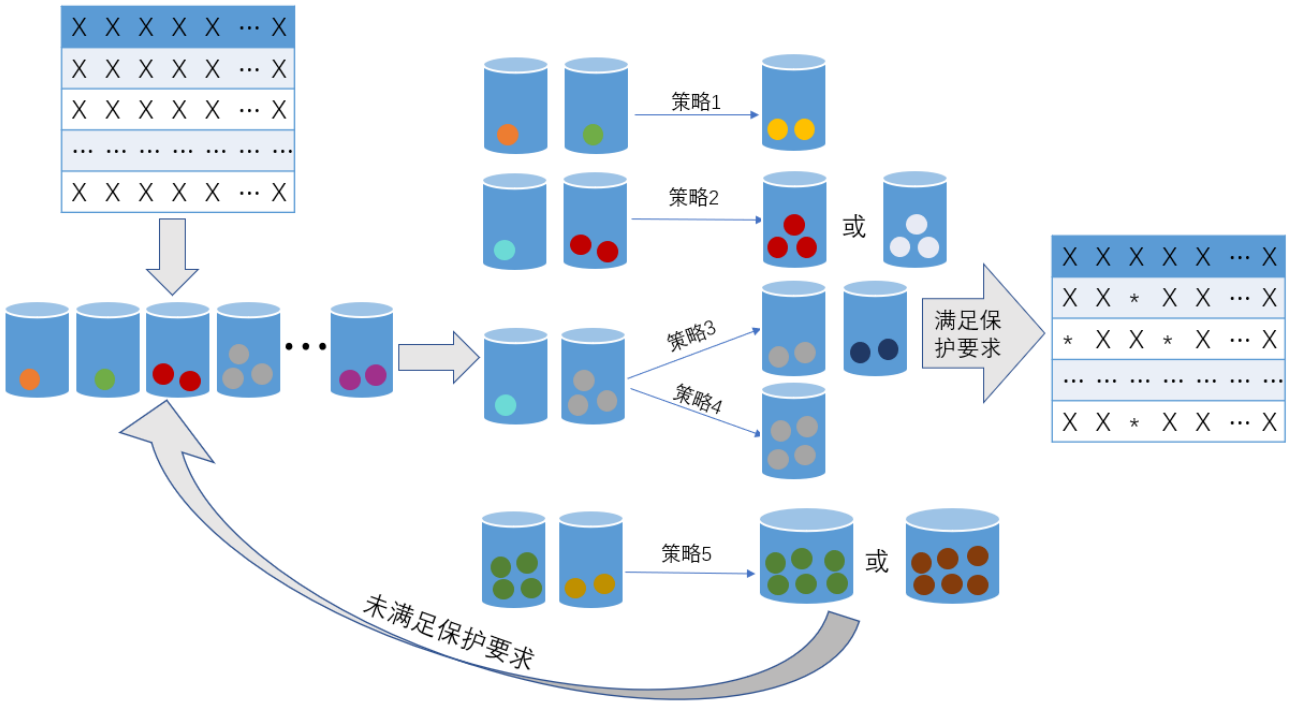


图 7-1 p-重保护匹配过程图

7.2 多重保护下的数据隐藏模型结果呈现

首先，在$p=3$的情况下，我们使用上述算法对二元 1 和二元 2 数据进行求解，最终得到的具体操作方案见附件。对于二元 1 数据，我们通过隐藏 67 个数据信息，达到了$3-$重保护的要求，最终产生 16 个大小为 3 的群，8 个大小为 4 的群。在处理过程中，一共执行了 3 次策略 1，10 次策略 2 和 7 次策略 5。对于二元 2 数据，在隐藏 686 个数据信息后达到了$3-$重保护的要求，最终产生 56 个大小为 3 的群，8 个大小为 4 的群和。在处理过程中，一共执行了 3 次策略 1，10 次策略 2 和 7 次策略 5。$p=3$的情况下二元数据的处理结果如表 7-3 所示。

表 7-3 $3-$重保护条件下二元数据的处理结果

$p-$重保护	数据	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
3	二元 1	80*6	3 4	16 8	3 次策略 1，10 次策略 2，7 次策略 5	67
	二元 2	200*12	3 4	56 8	67 次策略 1，56 次策略 2，8 次策略 5	686

其次考虑$p=5$的情况下二元 1 和二元 2 数据的处理，最终得到的具体操作方案见附件。对于二元 1 数据，我们通过隐藏 157 个数据信息，达到了$5-$重保护的要求，最

终产生 7 个大小为 5 的群，5 个大小为 6 的群，1 个大小为 7 的群，1 个大小为 8 的群。在处理过程中，一共执行了 4 次策略 1，3 次策略 2，5 次策略 4 和 18 次策略 5。对于二元 2 数据，在隐藏 1116 个数据信息后达到了5 −重保护的要求，最终产生 26 个大小为 5 的群，7 个大小为 6 的群和 4 个大小为 7 的群。在处理过程中，一共执行了 64 次策略 1，38 次策略 2，24 次策略 4，32 次策略 5。$p = 5$的情况下二元数据的处理结果如表 7-4 所示。

表 7-4 5 −重保护条件下二元数据的处理结果

$p-$重保护	数据	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
5	二元 1	80*6	5	7	4 次策略 1， 3 次策略 2，5 次策略 4，18 次策略 5	157
			6	5		
			7	1		
			8	1		
	二元 2	200*12	5	26	64 次策略 1，38 次策略 2，24 次策略 4，32 次策略 5	1116
			6	7		
			7	4		

随后考虑$p = 8$的情况，最终得到对二元 1 和二元 2 数据的具体处理具体操作方案见附件。对于二元 1 数据，我们以隐藏 230 个数据信息的方案，达到了8 −重保护的要求，最终分别生成 3、5、1 个对应大小为 8、9、11 的群。在数据处理过程中，一共执行了 2 次策略 1，7 次策略 2，5 次策略 4 和 21 次策略 5。对于二元 2 数据，我们以隐藏 1492 个数据信息的方案达到了8 −重保护的要求，最终分别产生 18、4、2 个对应大小为 8、9、10 的群。在处理过程中，一共执行了 60 次策略 1，26 次策略 2，44 次策略 4，41 次策略 5。$p = 8$的情况下二元数据的处理结果如表 7-5 所示。

表 7-5 8 −重保护条件下二元数据的处理结果

$p-$重保护	数据	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
8	二元 1	80*6	8	3	2 次策略 1， 7 次策略 2，5 次策略 4，21 次策略 5	230
			9	5		
			11	1		
	二元 2	200*12	8	18	60 次策略 1， 26 次策略 2，44 次策略 4，41 次策略 5	1492
			9	4		
			10	2		

下面对多元数据进行求解。

在$p = 3$的情况下，我们使用上述算法对多元 1 和多元 2 数据进行求解，最终得到的具体操作方案见附件。对于多元 1 数据，我们通过隐藏 927 个数据信息，达到了3 −重保护的要求，最终产生 62 个大小为 3 的群，2 个大小为 4 的群和个 6 元群。在处理过程

中，一共执行了 68 次策略 1，63 次策略 2，1 次策略 4 和 3 次策略 5。对于二元 2 数据，在隐藏 686 个数据信息后达到了3 −重保护的要求，最终产生 258 个大小为 3 的群，4 个大小为 4 的群和 2 个五元组。在处理过程中，一共执行了 266 次策略 1，264 次策略 2 和 7 次策略 5。$p = 3$的情况下二元数据的处理结果如表 7-6 所示。

表 7-6 3 −重保护条件下多元数据的处理结果

数据名称	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
3-多元 1（元=5）	200*8	3 4 6	62 2 1	68 次策略 1， 63 次策略 2，1 次策略 4，3 次策略 5	927
3-多元 2（元=10）	800*16	3 4 5	258 4 2	266 次策略 1， 264 次策略 2，4 次策略 4，2 次策略 5	10115

在$p = 5$的情况下对多元 1 和多元 2 数据进行处理，最终得到的具体操作方案见附件。对于二元 1 数据，我们通过隐藏 1165 个数据信息，达到了5 −重保护的要求，最终产生 35 个大小为 5 的群，3 个大小为 6 的群，1 个大小为 7 的群。在处理过程中，一共执行了 46 次策略 1，39 次策略 2，69 次策略 4 和 7 次策略 5。对于二元 2 数据，在隐藏 11230 个数据信息后达到了5 −重保护的要求，最终产生 152 个大小为 3 的群，2 个大小为 6 的群和 4 个大小为 7 的群。在处理过程中，一共执行了 162 次策略 1，158 次策略 2，318 次策略 4，4 次策略 5。$p = 5$的情况下二元数据的处理结果如表 7-7 所示。

表 7-7 5 −重保护条件下多元数据的处理结果

数据名称	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
5-多元 1（元=5）	200*8	5 6 7	35 3 1	46 次策略 1， 39 次策略 2，69 次策略 4，7 次策略 5	1165
5-多元 2（元=10）	800*16	3 6 7	152 2 4	162 次策略 1，158 次策略 2，318 次策略 4，4 次策略 5	11230

最后考虑$p = 8$的情况，得到对多元 1 和多元 2 数据的具体操作方案见附件。对于多元 1 数据，我们以隐藏 1301 个数据信息的方案，达到了8 −重保护的要求，最终分别生成 21、2、1 个对应大小为 8、9、14 的群。在数据处理过程中，一共执行了 33 次策略 1，24 次策略 2，110 次策略 4 和 9 次策略 5。对于多元 2 数据，我们以隐藏 11646 个数据信息的方案达到了8 −重保护的要求，最终分别产生 94、1、1、1 个对应大小为 8、10、12、14 的群。在处理过程中，一共执行了 103 次策略 1，102 次策略 2，924 次策略 4，41 次策略 5。$p = 8$的情况下二元数据的处理结果如表 7-8 所示。

表 7-8 8－重保护条件下多元数据的处理结果

数据名称	数据规模	群体大小	群的个数	采取策略	数据隐藏个数
8-多元 1（元=5）	200*8	8 9 14	21 2 1	33 次策略 1，24 次策略 2，110 次策略 4，9 次策略 5	1301
8-多元 2（元=5）	800*16	8 10 12 14	94 1 1 1	103 次策略 1，102 次策略 2，492 次策略 4，5 次策略 5	11646

我们对表 7-3、7-4、7-5、7-6、7-7 和 7-8 的数据进行总结。发现在规定p重保护的条件下，经过本文提出的算法处理，得到的分群大小大部分接近p，少数情况下才会出现远大于p的群体出现。这一现象体现了本文的贪心算法运行中，以尽量使配对后产生的新群体刚好满足p重保护的要求为目的，尽可能减少了隐藏数据的个数。

对于算法处理后的具体方案，我们选取分群数较少的 8 重保护要求下的多元 1 数据进行呈现。多元 1 数据处理后具体的分群情况见表 7-9 所示。

表 7-9 8重保护条件下多元 1 数据的处理结果

生成群体大小	群内数据
8	**31****、0*****4*、3*****3*、4*****4*、*0****1*、******03、**2*2***、*3*****2、*****43*、*11*****、3*1*****、4*******、0*******、***4**0*、*4******、**0*1***、*****3**、**2*****、******4*、*****2**、*****0**
9	***2**2*、*****1**
14	********

7.3 算法评估

针对p重保护条件下的二元数据和多元数据，我们对本文提出的算法进行了时间性能的实验分析，具体结果如表 7-10 所示。根据表 7-10 可以得出，对于$p=3$，5，8的不同取值，我们的算法可以在短时间内针对二元 1、二元 2 和多元 1 数据集在短时间内高效地求出结果；对于规模较大的多元 2 数据，我们的算法也能在一定时间内得出结果。

表 7-10 p-重保护算法评估

p－重保护	二元 1 运行时间(s)	二元 2 运行时间(s)	多元 1 运行时间(s)	多元 2 运行时间(s)
3	0.053	6.90	6.76	728.9
5	0.068	7.28	6.99	780.7
8	0.077	7.51	7.27	786.2

p-重保护运行时间均为多次运行后取得的平均，其中二元 1、2，多元 1 取 100 次运行后的均值，多元 2 取 10 次运行后的均值。

将多元数据下的运行时间与二元数据的运行时间进行对比发现，对于$m = 200$，$n = 12$的二元数据、$m = 200$，$n = 8$的多元数据平均运行时间相似。可以说明我们提出的算法运行时间不会随着数据取值范围的扩大而明显增长，且基本符合时间复杂度$O(m^3 n)$。

八．第四问问题分析与求解

8.1 问题重述

问题四在前三问的基础上，引入三种情形：情形一要求考虑对多元数据进行隐藏操作时限定隐藏模式。在隐藏时要求只能进行单隐或者全隐藏；情形二要求从多元高维数据中取出若干维度进行全发布（全发布的数据不考虑隐私暴露），对于其余维度的数据要求达到信息隐藏的效果；情形三要求对每个用户定制不同程度的隐私保护技术，即每个用户可以选择特定的p-重保护要求。

8.2 问题分析与算法提出

首先对情形 1 进行分析。对于情形 1，由于隐藏手段只允许单隐或者全隐，即限定了(*, x, x, x), (x, *, x, x), (x, x, *, x), (x, x, x, *)以及(*, *, *, *)五种隐藏模式，并且 x 表示相同的数据。基于此，我们对于个体的信息位进行分析后得知，若信息位中有 2 种数据（例如 0010 只有“0”与“1”），并且其中一种数据只出现 1 次，或信息位中只有 1 种数据（例如 1111），那么可以进行匹配操作：若存在只有出现 1 次的数据位不匹配，其余完全匹配的个体（如 0010+0020），则进行单隐；其余情况（例如 0120 且无完全匹配个体），则进行全隐。

基于上文分析，我们提出算法步骤如下：

算法步骤

Step1：读取附件中多元数据集，并对数据集中的全部数据进行分群操作，即数据中全部信息位均相同的归入同一群体，不完全相同的数据各自单独成群；

Step2：对个体的信息位进行检索，对信息位的种类>2的个体进行“全隐”；

Step3：对个体的信息位进行检索，对信息位中只出现 1 次的数字位置与其他个体满足信息位中只出现 1 次的数字位置相同，则对其进行匹配操作，进行“单隐”；

Step4：判断所有群体是否满足题目的隐藏数据要求，若不满足，则对其余个体进行“全隐”。

Step5；输出当前方案。

对于情形 2，需要若干维度的数据全发布（假设全发布的数据不构成隐私暴露），那么进行数据隐藏时，不需要考虑全发布位，只需考虑其他位的匹配即可。如第 6 章所述，评价指标使用$T_l = R_l/U_l$，来判定每次隐藏操作的优劣性，以保证操作在我们设定

的指标下“最优”。

同样，可以采取的匹配策略分为三大类：

（1）第一类策略：尚未匹配成功的单个个体之间进行匹配。

（2）第二类策略：单个个体与配对成功的多个体群进行匹配。

（3）第三类策略：配对成功的多个体群之间进行匹配。

基于上述匹配策略，针对情形 2，提出算法如下：

算法步骤

Step1：读取附件中多元数据集，提取所有无需全发布的数据组成新数据集；

Step2: 对新数据集中的全部数据进行分群操作，即数据中全部信息位均相同的归入同一群体，不完全相同的数据各自单独成群；

Step3：计算群与群之间的评价指标。根据群与群之间的具体情况，选择上述策略中的一种，并根据对应的指标公式计算两个群体间进行配对操作的指标。

Step4：选取 Step2 中评价指标结果最优的两个群体进行匹配操作。

Step5：判断所有群体是否满足题目的隐藏数据要求，若不满足，则跳转回 Step2，继续计算和配对操作；若满足，则跳转至 Step5。

Step6：输出当前方案。

8.3 结果呈现

对于情形 2，我们选取多元 1 数据作为处理对象。同时，假定数据集中的第一列数据与第四列数据进行全发布，且不考虑隐私保护。我们使用上文所述的算法对该数据集进行处理。得到结果：最终对 399 个数据信息进行隐藏,满足了在第一列数据和第四列数据全发布情况下的隐私保护要求。其中，策略 1 共使用 97 次，策略 3 使用 1 次，策略 4 使用 5 次。具体方案见附件。

在算法运行时间方面，由于全发布操作减少了需要处理的数据维度，即n的大小，在该情形下，算法平均 4.63s 即可求解出结果，相比于不考虑全发布的多元 1 数据处理所需的平均 6.3s，运行时间有所减少。

九．总结

本文研究了如何通过隐藏尽可能少的数据信息达成数据隐藏的效果。通过对二元、多元数据分析，明确最小化数据隐藏量的目标建立数学模型。并采用分群思想，借助评价指标进行匹配策略的筛选，最后使用贪心算法求解出模型优解。通过以上章节中对算法的性能分析可以得出，我们的算法能够在有限时间内，高效得出符合目标的优解。

十．参考文献

[1] Abadi M, Agarwal A, Barham P, et al. Tensorflow:Largescale machine learning on heterogeneous distributed systems.arXiv preprint arXiv:1603. 04467,2016.

[2] 钱文君, 沈晴霓, 吴鹏飞, 董春涛, 吴中海.大数据计算环境下的隐私保护技术研究进展[J].计算机学报, 2022, 45(04):669-701.

[3] 天枢实验室。

http://blog. nsfocus. net/inventory-of-data-breaches-at-home-and-abroad-in-2019/

[4] MACHANAVAJJHALA A，GEHRKE J，KIFER D.1-diversity：privacy beyond K-anonymity[C].Atlanta： Proceeding of the 22nd Internaional Conferenceon Data Engineering，IEEE Computer Society，2006：24-35.

十一．附录

附录一：第一问算法（python）

```
import pandas as pd
import numpy as np
import random

# da = pd.read_excel('test.xlsx', header=None)
da = pd.read_excel('B 题附件.xlsx', header=None,sheet_name='二元 2')
#da = pd.read_excel('B 题附件.xlsx', header=None, sheet_name='多元 2')
data = da.to_numpy()

feature = data.shape[1]
minans = 30000
xuhao = 0
long = data.shape[0]
dataA = {}
dataB = {}
for i in range(0, long):
    s = ''.join(str(k) for k in data[i])
    if s not in dataA:
        dataA[s] = 1
    else:
        dataA[s] += 1
    xuhao += 1
print(len(dataA))   # a  初始桶
# print(b)   # b  初始特征对于序号
for iter in range(0, 100):

    a = dataA.copy()
    f = 0
    ans = 0
    record = []
    while f == 0:
        p = []
        for i in a.keys():
            if a[i] != 0:
                p.append(i)
        o = len(p)
        # print(o,end=" ")
        if 1 not in a.values():
            break
        h = np.eye(o) * 100
```

```
h1 = np.eye(o) * 100
h2 = np.eye(o) * 100

# ---------------------------------距离矩阵计算---------------------------------
for i in range(0, o - 1):
    for j in range(i + 1, o):
        jsi = a[p[i]]
        jsj = a[p[j]]
        if jsi == 1 or jsj == 1:
            q = 0   # 要改的为里有多少星号
            q1 = 0   # 桶里要改的位有没有非零
            for k in range(0, feature):
                if p[i][k] != p[j][k]:
                    h[i][j] += 1
                    if p[i][k] == '*':
                        q += 1
                        q1 += 1 * jsi
                    if p[j][k] == '*':
                        q +=1
                        q1+=jsj

            if jsi == 1 and jsj == 1:
                h2[i][j] = 2 * h[i][j]
                h[i][j] = 2 / (h[i][j] * 2)

            elif jsi>2 or jsj>2:
                h2[i][j] = 2 * h[i][j] - q
                h[i][j] = 1 / h2[i][j]
            elif jsi==2 or jsj==2:

                h2[i][j] = h[i][j]*3-q1
                h[i][j] = 1 / (h2[i][j])
                h1[i][j] = 2
        else:
            h[i][j] = -1
# ---------------------------------距离矩阵计算---------------------------------
m1 = 0   # m1 表示待处理人之间最小差距
for i in range(0, o - 1):
    for j in range(i + 1, o):
        h[j][i] = h[i][j]
        if h[i][j] != -1:
            if m1 < h[i][j]:
                m1 = h[i][j]
```

```
        weizhileixing = -1
        haoweizhi = []  # 好位置：两个都没对象的进行处理
        for i in range(0, o - 1):
            for j in range(i + 1, o):
                if h[i][j] == m1:
                    if a[p[i]] == 1 and a[p[j]] == 1:
                        haoweizhi.append([i, j, 1])  # 1:内部消化
                    elif a[p[i]] == 1 or a[p[j]] == 1:
                        if (a[p[i]] == 1 and a[p[j]] > 2) or (a[p[i]] > 2 and a[p[j]] == 1):
                            haoweizhi.append([i, j, 3])  # ：桶多于两个，拉一个出
来
                        else:
                            haoweizhi.append([i, j, 4])  # 桶不多于两个，加入
        # print(m1,haoweizhi)
        # print(h)
        # print(h2)
        seed = random.randint(0, len(haoweizhi) - 1)
        x = p[haoweizhi[seed][0]]
        y = p[haoweizhi[seed][1]]
        weizhileixing = haoweizhi[seed][2]
        # print(weizhileixing)
        if weizhileixing == 1:
            a[x] -= 1
            a[y] -= 1
            cunchujieguo = [x, y]
        elif weizhileixing == 3:
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            a[x] -= 1
            a[y] -= 1
        else:
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            w4 = a[x] + a[y]
            a[x] = 0
            a[y] = 0
        ss = ''
        for i in range(0, feature):
            if x[i] != y[i]:
                ss = ss + '*'
```

```
                else:
                    ss = ss + x[i]
            ans += h2[haoweizhi[seed][0]][haoweizhi[seed][1]]
            cunchujieguo.append(ss)
            cunchujieguo.append(weizhileixing)
            record.append(cunchujieguo)
            if weizhileixing == 1:
                a[ss] = 2
            elif weizhileixing == 3:
                a[ss] =+2
            else:
                a[ss] = w4

            # if ans > 358:
            #     print('大于 358')
            #     ans = 3000
            #     break
        print(ans)
        if minans > ans:
            minans = ans
            minrecord = record
print(minans)
print(minrecord)
print(len(minrecord))
lx = []
result = []
for i in data:
    pp = []
    for j in i:
        pp.append(j)
    result.append(pp)
for l in minrecord:
    p1 = l[0]
    p2 = l[1]
    lx.append(l[3])
    t = 0

    if l[3] == 1:
        for i in range(0, long):
            s = ''.join(str(k) for k in result[i])
            if s == p1 or s == p2:
                for j in range(0, feature):
                    result[i][j] = l[2][j]
    elif l[3] == 3:
```

```
            if p1[0] == 'w':
                p1 = p1[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
            else:
                p2 = p2[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
        elif l[3] == 4:
            if p1[0] == 'w':
                p1 = p1[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
            else:
                p2 = p2[1:]
```

```
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
print(lx)
data_df = pd.DataFrame(result)
writer = pd.ExcelWriter('new q.xlsx')
data_df.to_excel(writer, header=None, index=False)
writer.save()
re=0
for i in result:
    for j in i:
        if j=='*':
            re+=1
print(re)
```

附录 2：第二问代码（python）

```
import pandas as pd
import numpy as np
import random

# da = pd.read_excel('test.xlsx', header=None)
#da = pd.read_excel('B 题附件.xlsx', header=None,sheet_name='二元 2')
da = pd.read_excel('1.xlsx', header=None, sheet_name='多元 1')
data = da.to_numpy()

feature = data.shape[1]
minans = 30000
xuhao = 0
long = data.shape[0]
dataA = {}
dataB = {}
for i in range(0, long):
    s = ''.join(str(k) for k in data[i])
    if s not in dataA:
        dataA[s] = 1
    else:
```

```
            dataA[s] += 1
        xuhao += 1
print(len(dataA))   # a 初始桶
# print(b)   # b 初始特征对于序号
for iter in range(0, 10):

        a = dataA.copy()
        f = 0
        ans = 0
        record = []
        while f == 0:
            p = []
            for i in a.keys():
                if a[i] != 0:
                    p.append(i)
            # print(sum(a.values()))
            o = len(p)
            # print(o,end=" ")
            if 1 not in a.values():
                break
            h = np.eye(o) * 100
            h1 = np.eye(o) * 100
            h2 = np.eye(o) * 100
            # ---------------------------------距离矩阵计算---------------------------------
            for i in range(0, o - 1):
                for j in range(i + 1, o):
                    jsi = a[p[i]]
                    jsj = a[p[j]]
                    if jsi == 1 or jsj == 1:
                        q = 0   # 要改的位里有多少星号（单体和桶中的一个）
                        q1 = 0   # 要改的位里有多少星号（单体和桶中的所有）
                        q2 = 0   # 桶内不同的位有多少星号
                        for k in range(0, feature):
                            if p[i][k] != p[j][k]:
                                h[i][j] += 1

                            if p[i][k] == '*':
                                q += 1
                                q1 += 1 * jsi
                            if p[j][k] == '*':
                                q +=1
                                q1+=jsj
                            if (jsi >= 2 and p[i][k] == '*') or (jsj >= 2 and p[j][k] ==
'*'):
```

```
                                    q2 += 1
                        if jsi == 1 and jsj == 1:
                            h2[i][j] = 2 * h[i][j]
                            h[i][j] = 2 / (h[i][j] * 2)
                        elif q2==h[i][j]:
                            h2[i][j]=h[i][j]
                            h[i][j] = 1 / (h[i][j])
                            h1[i][j]=-1 #  进桶  桶不变
                        elif jsi>2 or jsj>2:
                            h2[i][j] = 2 * h[i][j] - q
                            h[i][j] = 1 / h2[i][j]
                        elif jsi==2 or jsj==2:
                            h2[i][j] = h[i][j]*3-q1
                            h[i][j] = 1 / (h2[i][j])
                            h1[i][j] = 2
                    else:
                        h[i][j] = -1
        # --------------------------------距离矩阵计算--------------------------------
        m1 = 0   # m1 表示待处理人之间最小差距
        for i in range(0, o - 1):
            for j in range(i + 1, o):
                h[j][i] = h[i][j]
                if h[i][j] != -1:
                    if m1 < h[i][j]:
                        m1 = h[i][j]

        weizhileixing = -1
        haoweizhi = []   #  好位置：两个都没对象的进行处理
        for i in range(0, o - 1):
            for j in range(i + 1, o):
                if h[i][j] == m1:
                    if a[p[i]] == 1 and a[p[j]] == 1:
                        haoweizhi.append([i, j, 1])   # 1:内部消化

                    elif a[p[i]] == 1 or a[p[j]] == 1:
                        if h1[i][j] == -1:
                            haoweizhi.append([i, j, 2])   #  不管桶几个，进桶  桶不变
                        elif (a[p[i]] == 1 and a[p[j]] > 2) or (a[p[i]] > 2 and a[p[j]] ==
1):
                            haoweizhi.append([i, j, 3])   #  ：桶多于两个，拉一个出
来
                        else:
                            haoweizhi.append([i, j, 4])   #  桶不多于两个，加入
        # print(m1,haoweizhi)
```

```
        # print(h)
        # print(h2)
        seed = random.randint(0, len(haoweizhi) - 1)
        x = p[haoweizhi[seed][0]]
        y = p[haoweizhi[seed][1]]
        weizhileixing = haoweizhi[seed][2]
        # print(x,y,weizhileixing)
        if weizhileixing == 1:
            a[x] -= 1
            a[y] -= 1
            cunchujieguo = [x, y]
        elif weizhileixing == 2:
            w2= a[x]+a[y]
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            a[x] -= 1
            a[y] -= 1
        elif weizhileixing == 3:
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            a[x] -= 1
            a[y] -= 1
        else:
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            w4 = a[x] + a[y]
            a[x] = 0
            a[y] = 0
        ss = ''
        for i in range(0, feature):
            if x[i] != y[i]:
                ss = ss + '*'
            else:
                ss = ss + x[i]
        ans += h2[haoweizhi[seed][0]][haoweizhi[seed][1]]
        cunchujieguo.append(ss)
        cunchujieguo.append(weizhileixing)
        record.append(cunchujieguo)
```

```
            if weizhileixing == 1:
                a[ss] = 2
            elif weizhileixing ==2:
                a[ss] = w2
            elif weizhileixing == 3:
                a[ss] = 2
            else:
                a[ss] = w4

            # if ans > 358:
            #     print('大于 358')
            #     ans = 3000
            #     break
        print('-----')
        print(ans)
        if minans > ans:
            minans = ans
            minrecord = record
            mina=a.copy()
print(minans)
print(minrecord)
print(len(minrecord))
print(mina.values())
lx = []
result = []
for i in data:
    pp = []
    for j in i:
        pp.append(j)
    result.append(pp)
for l in minrecord:
    p1 = l[0]
    p2 = l[1]
    lx.append(l[3])
    t = 0

    if l[3] == 1:
        for i in range(0, long):
            s = ''.join(str(k) for k in result[i])
            if s == p1 or s == p2:
                for j in range(0, feature):
                    result[i][j] = l[2][j]
    elif l[3] == 2:
        if p1[0] == 'w':
```

```
                p1 = p1[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
            else:
                p2 = p2[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
        elif l[3] == 3:
            if p1[0] == 'w':
                p1 = p1[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
            else:
                p2 = p2[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
```

```
    elif l[3] == 4:
        if p1[0] == 'w':
            p1 = p1[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
        else:
            p2 = p2[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
print(lx)
data_df = pd.DataFrame(result)
writer = pd.ExcelWriter('new duoyuan.xlsx')
data_df.to_excel(writer, header=None, index=False)
writer.save()
re=0
for i in result:
    for j in i:
        if j=='*':
            re+=1
print(re)
```

附录 3：第三问代码（python）

```
import pandas as pd
import numpy as np
import random
```

```
# da = pd.read_excel('test.xlsx', header=None)
#da = pd.read_excel('B 题附件.xlsx', header=None,sheet_name='二元 2')
da = pd.read_excel('B 题附件.xlsx', header=None, sheet_name='多元 2')
data = da.to_numpy()

feature = data.shape[1]
minans = 30000
xuhao = 0
long = data.shape[0]
dataA = {}
dataB = {}
for i in range(0, long):
    s = ''.join(str(k) for k in data[i])
    if s not in dataA:
        dataA[s] = 1
    else:
        dataA[s] += 1
    xuhao += 1
print(dataA)   # a  初始桶
# print(b)   # b  初始特征对于序号
kk=8

for iter in range(0, 3):

    a = dataA.copy()
    f = 0
    ans = 0
    record = []
    while f == 0:
        p = []
        zuixiao=1000
        for i in a.keys():
            if a[i] != 0:
                p.append(i)
                if a[i]<zuixiao:
                    zuixiao=a[i]
        #print(a)
        o = len(p)
        print(o,end=" ")
        if zuixiao>=kk :
            break
        h = np.eye(o) * 100
        h1 = np.eye(o) * 100
        h2 = np.eye(o) * 100
```

```
        # ---------------------------------距离矩阵计算---------------------------------
        for i in range(0, o - 1):
            for j in range(i + 1, o):

                jsi = a[p[i]]
                jsj = a[p[j]]
                if jsi<kk or jsj<kk:
                    q = 0   # 要改的位里有多少星号（单体和桶中的一个）
                    q1 = 0   # 要改的位里有多少星号（单体和桶中的所有）
                    for k in range(0, feature):
                        if p[i][k] != p[j][k]:
                            h[i][j] += 1
                            if p[i][k] == '*':
                                q += 1
                                q1 += jsi
                            if p[j][k] == '*':
                                q += 1
                                q1 += jsj
                    # 第三问新指标的分子：就的人的个数 若原始群都不满足 分子为两群之和
                    if jsi == 1 or jsj == 1:
                        if jsi == 1 and jsj == 1:# 内部消化
                            h2[i][j] = 2 * h[i][j]
                            h[i][j] = 2 / (h[i][j] * 2)
                        elif jsi >= kk or jsj >=kk :# 1 群+大群合成
                            h2[i][j] = (jsi+jsj) * h[i][j] - q1
                            h[i][j] = 1 / h2[i][j]
                        else:# 个数 1 的群+个数 2 的群=3 群
                            h2[i][j] = h[i][j] * (jsi+jsj) - q1
                            h[i][j] = (jsi+jsj) / (h2[i][j])
                            h1[i][j] = 2
                    elif jsi<kk and jsj<kk:
                        h2[i][j] = (jsi + jsj) * h[i][j] - q1
                        h[i][j] = (jsi + jsj)  / h2[i][j]
                    else:
                        h2[i][j] = (jsi + jsj) * h[i][j] - q1
                        h[i][j] = 1 / h2[i][j]
                else:#两个已满足的大群
                    h[i][j]=-1
        # ---------------------------------距离矩阵计算---------------------------------
        m1 = 0   # m1 表示待处理人之间最小差距
        for i in range(0, o - 1):
            for j in range(i + 1, o):
```

```
                h[j][i] = h[i][j]
                if h[i][j] != -1:
                    if m1 < h[i][j]:
                        m1 = h[i][j]
weizhileixing = -1
haoweizhi = []   # 好位置：两个都没对象的进行处理
for i in range(0, o - 1):
    for j in range(i + 1, o):
        if h[i][j] == m1:
            if a[p[i]] == 1 and a[p[j]] == 1:
                haoweizhi.append([i, j, 1])   # 1:内部消化
            elif a[p[i]] == 1 or a[p[j]] == 1:
                if (a[p[i]] == 1 and a[p[j]] > 2) or (a[p[i]] > 2 and a[p[j]] == 1):
                    haoweizhi.append([i, j, 3])   # ：桶多于两个，加入
                else:
                    haoweizhi.append([i, j, 4])   # 桶不多于两个，加入
            elif (a[p[i]]<kk) or (a[p[j]]<kk):# 大群+大群
                haoweizhi.append([i,j,5])
# print(m1,haoweizhi)
# print(h)
# print(h2)
seed = random.randint(0, len(haoweizhi) - 1)
x = p[haoweizhi[seed][0]]
y = p[haoweizhi[seed][1]]
weizhileixing = haoweizhi[seed][2]
#print(weizhileixing,a[x],a[y])
if weizhileixing == 1:
    a[x] -= 1
    a[y] -= 1
    cunchujieguo = [x, y]
elif weizhileixing == 3:
    if a[x] == 1:
        cunchujieguo = ['w' + x, y]
    else:
        cunchujieguo = [x, 'w' + y]
    w3= a[x] +a[y]
    a[x] = 0
    a[y] = 0
elif weizhileixing == 4:
    if a[x] == 1:
        cunchujieguo = ['w' + x, y]
    else:
        cunchujieguo = [x, 'w' + y]
    w4 = a[x] + a[y]
```

```
                a[x] = 0
                a[y] = 0
            else:
                w5 = a[x] + a[y]
                cunchujieguo=[x,y]
                a[x] = 0
                a[y] = 0
            ss = ''
            for i in range(0, feature):
                if x[i] != y[i]:
                    ss = ss + '*'
                else:
                    ss = ss + x[i]
            ans += h2[haoweizhi[seed][0]][haoweizhi[seed][1]]
            cunchujieguo.append(ss)
            cunchujieguo.append(weizhileixing)
            record.append(cunchujieguo)
            if weizhileixing == 1:
                a[ss] = 2
            elif weizhileixing == 3:
                a[ss] = w3
            elif weizhileixing ==4:
                a[ss] = w4
            else:
                a[ss] = w5

            # if ans > 358:
            #         print('大于 358')
            #         ans = 3000
            #         break
            co=0
            for i in a.keys():
                co+=a[i]*(i.count('*'))
            # print(co,ans,weizhileixing)
            # if co!=ans:
            #         print(a)
        print(ans)
        if minans > ans:
            minans = ans
            minrecord = record
            mina=a.copy()
print('start')
print(minans)
print(minrecord)
```

```
print(len(minrecord))
for i in mina.keys():
    if mina[i]!=0:
        print(i,mina[i])
lx = []
result = []
for i in data:
    pp = []
    for j in i:
        pp.append(j)
    result.append(pp)
for l in minrecord:
    p1 = l[0]
    p2 = l[1]
    lx.append(l[3])
    t = 0

    if l[3] == 1:
        for i in range(0, long):
            s = ''.join(str(k) for k in result[i])
            if s == p1 or s == p2:
                for j in range(0, feature):
                    result[i][j] = l[2][j]
    elif l[3] == 3:
        if p1[0] == 'w':
            p1 = p1[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
        else:
            p2 = p2[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
```

```
                        break
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
    elif l[3] == 4:
        if p1[0] == 'w':
            p1 = p1[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
        else:
            p2 = p2[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
    elif l[3] == 5:
        for i in range(0, long):
            s = ''.join(str(k) for k in result[i])
            if s == p1:
                for j in range(0, feature):
                    result[i][j] = l[2][j]
        for i in range(0, long):
            s = ''.join(str(k) for k in result[i])
            if s == p2:
                for j in range(0, feature):
                    result[i][j] = l[2][j]
```

```
print(lx)
data_df = pd.DataFrame(result)
writer = pd.ExcelWriter('q3-.xlsx')
data_df.to_excel(writer, header=None, index=False)
writer.save()
re=0
for i in result:
    for j in i:
        if j=='*':
            re+=1
print(re)
```

附录 4：第四问代码（python）

```
import pandas as pd
import numpy as np
import random

# da = pd.read_excel('test.xlsx', header=None)
#da = pd.read_excel('B 题附件.xlsx', header=None,sheet_name='二元 2')
da = pd.read_excel('1.xlsx', header=None, sheet_name='多元 1')
data = da.to_numpy()

feature = data.shape[1]
minans = 30000
xuhao = 0
long = data.shape[0]
dataA = {}
dataB = {}
for i in range(0, long):
    s = ''.join(str(k) for k in data[i])
    if s not in dataA:
        dataA[s] = 1
    else:
        dataA[s] += 1
    xuhao += 1
print(len(dataA))   # a  初始桶
# print(b)   # b  初始特征对于序号
for iter in range(0, 10):

    a = dataA.copy()
    f = 0
    ans = 0
    record = []
    while f == 0:
        p = []
```

```
            for i in a.keys():
                if a[i] != 0:
                    p.append(i)
            # print(sum(a.values()))
            o = len(p)
            # print(o,end=" ")
            if 1 not in a.values():
                break
            h = np.eye(o) * 100
            h1 = np.eye(o) * 100
            h2 = np.eye(o) * 100
            # ---------------------------------距离矩阵计算---------------------------------
            for i in range(0, o - 1):
                for j in range(i + 1, o):
                    jsi = a[p[i]]
                    jsj = a[p[j]]
                    if jsi == 1 or jsj == 1:
                        q = 0   # 要改的位里有多少星号（单体和桶中的一个）
                        q1 = 0   # 要改的位里有多少星号（单体和桶中的所有）
                        q2 = 0   # 桶内不同的位有多少星号
                        for k in range(0, feature):
                            if p[i][k] != p[j][k]:
                                h[i][j] += 1

                                if p[i][k] == '*':
                                    q += 1
                                    q1 += 1 * jsi
                                if p[j][k] == '*':
                                    q +=1
                                    q1+=jsj
                                if (jsi >= 2 and p[i][k] == '*') or (jsj >= 2 and p[j][k] ==
'*'):

                                    q2 += 1
                        if jsi == 1 and jsj == 1:
                            h2[i][j] = 2 * h[i][j]
                            h[i][j] = 2 / (h[i][j] * 2)
                        elif q2==h[i][j]:
                            h2[i][j]=h[i][j]
                            h[i][j] = 1 / (h[i][j])
                            h1[i][j]=-1 #  进桶  桶不变
                        elif jsi>2 or jsj>2:
                            h2[i][j] = 2 * h[i][j] - q
                            h[i][j] = 1 / h2[i][j]
                        elif jsi==2 or jsj==2:
```

```
                            h2[i][j] = h[i][j]*3-q1
                            h[i][j] = 1 / (h2[i][j])
                            h1[i][j] = 2
                    else:
                        h[i][j] = -1
        # ---------------------------------距离矩阵计算---------------------------------
        m1 = 0   # m1 表示待处理人之间最小差距
        for i in range(0, o - 1):
            for j in range(i + 1, o):
                h[j][i] = h[i][j]
                if h[i][j] != -1:
                    if m1 < h[i][j]:
                        m1 = h[i][j]

        weizhileixing = -1
        haoweizhi = []   # 好位置：两个都没对象的进行处理
        for i in range(0, o - 1):
            for j in range(i + 1, o):
                if h[i][j] == m1:
                    if a[p[i]] == 1 and a[p[j]] == 1:
                        haoweizhi.append([i, j, 1])   # 1:内部消化

                    elif a[p[i]] == 1 or a[p[j]] == 1:
                        if h1[i][j] == -1:
                            haoweizhi.append([i, j, 2])   # 不管桶几个，进桶 桶不变
                        elif (a[p[i]] == 1 and a[p[j]] > 2) or (a[p[i]] > 2 and a[p[j]] ==
1):
                            haoweizhi.append([i, j, 3])   # ：桶多于两个，拉一个出
来
                        else:
                            haoweizhi.append([i, j, 4])   # 桶不多于两个，加入
        # print(m1,haoweizhi)
        # print(h)
        # print(h2)
        seed = random.randint(0, len(haoweizhi) - 1)
        x = p[haoweizhi[seed][0]]
        y = p[haoweizhi[seed][1]]
        weizhileixing = haoweizhi[seed][2]
        # print(x,y,weizhileixing)
        if weizhileixing == 1:
            a[x] -= 1
            a[y] -= 1
            cunchujieguo = [x, y]
        elif weizhileixing == 2:
```

```
            w2= a[x]+a[y]
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            a[x] -= 1
            a[y] -= 1
        elif weizhileixing == 3:
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            a[x] -= 1
            a[y] -= 1
        else:
            if a[x] == 1:
                cunchujieguo = ['w' + x, y]
            else:
                cunchujieguo = [x, 'w' + y]
            w4 = a[x] + a[y]
            a[x] = 0
            a[y] = 0
        ss = ''
        for i in range(0, feature):
            if x[i] != y[i]:
                ss = ss + '*'
            else:
                ss = ss + x[i]
        ans += h2[haoweizhi[seed][0]][haoweizhi[seed][1]]
        cunchujieguo.append(ss)
        cunchujieguo.append(weizhileixing)
        record.append(cunchujieguo)
        if weizhileixing == 1:
            a[ss] = 2
        elif weizhileixing ==2:
            a[ss] = w2
        elif weizhileixing == 3:
            a[ss] = 2
        else:
            a[ss] = w4

        # if ans > 358:
        #         print('大于 358')
        #         ans = 3000
```

```
            #         break
        print('-----')
        print(ans)
        if minans > ans:
            minans = ans
            minrecord = record
            mina=a.copy()
print(minans)
print(minrecord)
print(len(minrecord))
print(mina.values())
lx = []
result = []
for i in data:
        pp = []
        for j in i:
            pp.append(j)
        result.append(pp)
for l in minrecord:
        p1 = l[0]
        p2 = l[1]
        lx.append(l[3])
        t = 0

        if l[3] == 1:
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1 or s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
        elif l[3] == 2:
            if p1[0] == 'w':
                p1 = p1[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p1:
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
            else:
                p2 = p2[1:]
                for i in range(0, long):
                    s = ''.join(str(k) for k in result[i])
                    if s == p2:
```

```
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
                        break
    elif l[3] == 3:
        if p1[0] == 'w':
            p1 = p1[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
        else:
            p2 = p2[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
    elif l[3] == 4:
        if p1[0] == 'w':
            p1 = p1[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
```

```
                        for j in range(0, feature):
                            result[i][j] = l[2][j]
        else:
            p2 = p2[1:]
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p2:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
                    break
            for i in range(0, long):
                s = ''.join(str(k) for k in result[i])
                if s == p1:
                    for j in range(0, feature):
                        result[i][j] = l[2][j]
print(lx)
data_df = pd.DataFrame(result)
writer = pd.ExcelWriter('new duoyuan.xlsx')
data_df.to_excel(writer, header=None, index=False)
writer.save()
re=0
for i in result:
    for j in i:
        if j=='*':
            re+=1
print(re)
```