arXiv:2601.00447v1 [cs.GT] 1 Jan 2026

统一质心聚类与非质心聚类中的比例公平

Benjamin Cookson、Nisarg Shah、Ziqi Yu

{bcookson,nisarg,ziqiyu}@cs.toronto.edu

摘要

受民主理念中的比例代表制启发的比例公平准则，近年来在聚类文献中日益受到关注。既有研究在两种不同范式下探讨了这些准则。Chen 等人 [2019] 研究了质心聚类，在这种聚类中，每个数据点的损失由其到所在簇中所选代表点（质心）的距离决定。Caragiannis 等人 [2024] 研究了非质心聚类，在这种聚类中，每个数据点的损失由其到所在簇中任意其他数据点的最大距离决定。

我们将这两种范式推广，提出半质心聚类：在这种聚类中，每个数据点的损失由其质心损失与非质心损失共同构成。我们研究两种比例公平准则——核及其宽松形式完全正当代表性（FJR）。我们的主要结果是一种新颖算法，它能在多项式时间内实现对核的常数近似，即使用于衡量质心损失和非质心损失的距离度量各不相同也能如此。对于限制更严格的损失函数以及较弱的 FJR 准则，我们还推导出改进的结果，并分别证明了相应的下界。

1 引言

聚类是人工智能中的一项核心任务，其目标是将一组n 个数据点划分为k 个簇$C_1,\dots,C_k$，或返回k 个质心$x_1,\dots,x_k$，也可以同时返回簇及其对应的质心。在许多情况下，数据点的划分与质心的选择被视为可以互换。例如，在传统的聚类设定中，常用的k-均值损失函数（待最小化）可以写成两种等价形式：

$$\min_{C_1,\dots,C_k}\sum_{t=1}^{k}\frac{1}{|C_t|}\sum_{i,j\in C_t}d(i,j)^2\equiv\min_{C_1,\dots,C_k,x_1,\dots,x_k}\sum_{t=1}^{k}\sum_{i\in C_t}d(i,x_t)^2.$$

这是因为，对于给定的划分$(C_1,\dots,C_k)$，每个簇C_t对应的质心x_t的最优选择是该簇中各点的平均值（这也可以推广到非欧几里得度量），将这些最优选择代入后即可得到左侧的形式。

经济学与人工智能交叉领域中一个新兴的研究方向，关注每个数据点都代表一个代理的应用，并研究如何应对自利代理人的策略操纵等问题 [Perote and Perote-Pe˜na，2004，Dekel 等人，2010，Meir 等人，2012，Chen 等人，2018，Hossainand Shah，2021] 或确保代理人的公平性 [Balcan etal.，2019年，Krishnaswamy 等人，2021年，Chaudhury 等人，2024]。在此，仅有一个待最小化的全局损失函数并不足够；还需要了解各个代理的损失函数（即偏好）。即使不同的代理层级损失函数形式在总体上等价，也会导致不同的结果。

我们的关注点是公平性——具体而言，是比例公平。简单来说，它为每一个可能的代理群组提供公平性保证，而不仅仅是由有限的敏感属性（如种族和性别，或它们的交集）所界定的群组；保证的力度与

群组规模及其成员偏好的凝聚程度成正比。比例公平准则已应用于人工智能领域中越来越多的问题 [Chen 等人，2019年，Hossain 等人，2021年，Krishnaswamy 等人，2021年，Fan 等人，2023年，Zhang 等人，2023年，Chaudhury 等人，2024]。

对于聚类，Chen 等人 [2019]率先研究了质心聚类中的比例公平，其中代理的损失是其到所在簇质心的距离。他们以设施选址为动机：目标是建造k 个设施；每个代理自然都希望靠近分配给自己的设施。Caragiannis 等人 [2024] 研究了非质心聚类中的比例公平，其中代理的损失是其与所在簇中其他代理之间距离的总体（例如，平均值或最大值）。他们的研究动机源于聚类联邦学习：数据源（代理）被划分为若干簇，每个簇中的代理协作训练一个模型；因此，每个代理都希望（就其数据分布而言）与所在簇中的其他代理相近，从而使协作训练出的模型对该代理自身也足够准确。

但在这两种应用以及许多其他应用中，代理可能自然会同时关注自己到所在簇质心的距离和自己到所在簇中其他代理的距离。例如，在设施选址中，个人可能希望分配给自己的设施离家近，同时也希望使用同一设施的其他人来自自己的社区，这样就更有可能遇到朋友和熟人。在聚类联邦学习中，质心可能代表不同代理簇协作训练出的模型。代理可能既关心模型是否适用于自己的数据分布，也关心所在簇中其他代理的数据分布是否与自己的相似，因为这种凝聚力有助于在簇内建立信任与合作。

代理的偏好甚至可能由质心损失和非质心损失共同构成，而这两类损失分别由完全不同的度量空间诱导。当教师将学生分组完成课程项目时，学生对于自己被分入哪个群组的偏好，取决于该群组被分配的项目主题（质心）以及群组中的其他学生（非质心）。两名学生可能是关系非常密切的朋友，但对项目主题却持有完全不同的看法。

这促使我们开始研究一种对质心聚类和非质心聚类的自然推广所具有的比例公平性，我们将其称为半质心聚类；在这种聚类中，每个代理i 的损失取决于代理C_t 所属的簇（即$i \in C_t$）以及分配给该簇的质心x_t。

通过借鉴 Chen 等人 [2019] 和 Caragiannis 等人 [2024],研究中常用的方法，并结合我们所提出的新颖而精巧的技术，我们得以为半质心聚类实现理想的比例公平保证。

1.1 我们的结果

	双度量损失		**加权单度量损失**	
	Core	FJR	Core	FJR
存在性上界	3	1	$\min\{2/\lambda, 3\}$	1
高效上界	$3+2\sqrt{3}$	4	$\min\{2/\lambda, f_\lambda\}$	$\min\{4, 2/\lambda, f_\lambda\}$
下界	2	1	$\max\{g_\lambda, \frac{2(1-\lambda)}{-2\lambda+1}\}$	1

表 1：半质心聚类在双度量损失和加权单度量损失下的核心与 FJR 近似。这里，$f_\lambda = \frac{\sqrt{2\lambda-11\lambda^2+13}+3-\lambda}{2-2\lambda}$和 $g_\lambda = \frac{\sqrt{\lambda^2-2\lambda+5}-\lambda+1}{2}$。

半质心聚类算法以一组n 数据点（代理）作为输入，并返回数据点的一个划分$(C_1,\ldots,C_k)$ 及相应的质心$(x_1,\ldots,x_k)$。每个代理$i \in C_t$ 的损失由损失函数 $\ell_i(C_t,x_t)$衡量，该函数同时取决于它所属的簇C_t 及其质心x_t。尽管我们的一些结果适用于任意损失函数，但大多数结果关注两类结构化

损失函数。双度量损失衡量代理i到其质心x_t的距离与其簇中任意代理$j \in C_t$之间最大距离之和；前者依据度量d^c计算，后者依据另一种度量d^m计算。加权单度量损失是一种特殊情形，其中两个度量均为某个共同度量的缩放版本，即$d^c = (1-\lambda) \cdot d$且$d^m = \lambda \cdot d$，其中$d$为某个度量，且$\lambda \in [0,1]$。

我们研究以往关于质心聚类和非质心聚类的工作中探讨的两种比例公平准则（及其乘法近似）：核及其松弛形式——充分正当代表（FJR）。直观而言，在给定k个簇和n个数据点时，核要求不存在一个由至少n/k个代理组成的集合S，使得这些代理按比例理应能够组成一个簇，并且存在一个可行的质心y，令S中的每个成员都更偏好(S,y)，而非当前分配给自己的簇。FJR对成功的偏离提出了更严格的要求：对于$i \in S$中的任何代理，其在(S,y)下的最大损失，应低于$i \in S$中任何代理在该算法的聚类方案下的最小损失。

首先，我们证明，以往针对质心聚类或非质心聚类开发的算法，在我们更一般的半质心聚类范式下均不适用，即便是在加权单度量损失这一受限类别中也是如此。随后，对于这两类损失，我们通过设计新颖算法，获得核和FJR近似保证；这些保证既可通过存在性论证获得，也可在多项式时间内实现，并且我们还证明了（存在性）下界。主要结果总结于表1。

最后，我们还在第6节中评估了我们的算法在真实世界数据集上的表现。我们发现，我们的算法始终能够实现对所考虑公平性概念的近乎完美近似，优于我们给出的理论最坏情况界，也优于诸如k-均值等经典聚类算法++的表现。我们还表明，在实现这些保证的同时，它们对经典聚类目标的损害很小。这将先前关于质心和非质心聚类的类似观察推广了[Chen 等人，2019年，Caragiannis 等人，2024]。

1.2 相关工作

如上所述，已有多篇论文研究聚类中的比例公平。Chen 等人[2019]率先为质心聚类引入核的概念，并使用贪心捕获算法，实现对核的$(1+\sqrt{2})$-近似。Micha 和 Shah[2020]将欧几里得L^2度量下的近似因子提升至2，适用于质心可以放置在$\mathbb{R}^t$中的任意位置这一“无约束设定”。Aziz 等人[2024]指出，在这种无约束设定下，无需考虑无限多个可能的质心位置，只需关注n个代理的位置，就能在多项式时间内实现对核的常数近似。Li 等人[2021]研究了比核更强的判据。Caragiannis 等人[2024]研究了非质心聚类中的比例公平，并采用贪心捕获算法的一种变体，针对最大距离损失实现2-核心。他们还研究了作为核的一种弱化形式的FJR。Ebadian and Micha[2025]利用质心聚类框架研究排序问题。在这一设定中，他们将可能的质心集合设为代理人集合，并寻找一种针对质心聚类的抽签，使每个代理人被选为所选聚类中质心的概率相同。他们通过一种受贪心捕获算法启发的算法，实现了具有事后比例公平保证的抽签。

Li 等人[2023]研究了一个类似的基于图的问题，即将朋友划分为群组，这等价于非质心聚类；不过，在该问题中，每个人对于被分入自己所在簇的每位朋友所获得的效用均为1（而非损失）。Arkin 等人[2009]研究了多维几何稳定室友问题，该问题等价于[Caragiannis 等人的非质心聚类模型，2024],但重点关注诸如$k = n/3$这类特殊情形。

最后，聚类中也研究了其他公平性概念（参见 Chhabra 等人[2021]的综述）。其中最著名的一种假设每个数据点都属于多个类别之一，并要求某些受保护类别在各个簇中的代表比例相等[Chierichetti 等人、2017年、Bera 等人、2019]。

2 预备知识

2.1 聚类模型

一个半质心 聚类实例是一个元组$(\mathcal{N}, \mathcal{M}, \{\ell_i\}_{i\in\mathcal{N}}, k)$，其中 $\mathcal{N}$是一个由n 个代理组成的集合； $\mathcal{M}$ 是一个由可能的聚类中心组成的集合； $\ell_i\colon \{C \subseteq \mathcal{N}\colon i \in C\} \times \mathcal{M} \to \mathbb{R}_{\geqslant 0}$ 是代理$i \in \mathcal{N}$的损失函数，其中 $\ell_i(C, x)$表示其属于簇C 并被分配到聚类中心x时产生的损失；而$k \in \mathbb{N}$是算法要返回的簇的数量。

聚类算法接收这样的一个实例，并返回一个聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$，其中 $(C_1, \ldots, C_k)$ 是代理人集合 $\mathcal{N}$ 的一个不相交划分（因此，对于所有互不相同的 t 、 $t' \in [k]$，$C \cup_{t\in[k]}$ 与 $C_t = \mathcal{N} C_t \cap C_{t'} = \varnothing$ ）。我们将每个元组 $(C_t, x_t) \in \mathcal{X}$ 称为一个簇，其中 C_t 是该簇的成员集合（ C_t 中的代理是该簇的成员），而 x_t 是分配给该簇的中心。

给定一个聚类 $\mathcal{X}$ 和一个代理 $i \in \mathcal{N}$，令 $\mathcal{X}(i)$ 表示包含成员 i 的簇的（唯一）索引（即 $i \in C_{\mathcal{X}(i)}$）。在聚类 $\mathcal{X}$ 下，每个代理 $i \in \mathcal{N}$ 承受的损失为 $\ell_i(C_{\mathcal{X}(i)}, x_{\mathcal{X}(i)})$。为简洁起见，稍微滥用一下记号，我们也将其写作 $\ell_i(\mathcal{X})$ 。

2.2 损失函数

我们的一些结果适用于上述定义的任意损失函数。不过，更有意思的结果适用于两类结构化的损失函数，它们自然地结合了先前研究的质心聚类和非质心聚类损失函数。

双度量损失：给定两个距离度量，[1] 一个质心度量 $\mathrm{d}^c\colon \mathcal{N} \times \mathcal{M} \to \mathbb{R}_{\geqslant 0}$ 和一个非质心度量 $\mathrm{d}^m\colon \mathcal{N} \times \mathcal{N} \to \mathbb{R}_{\geqslant 0}$。给定一种聚类 $\mathcal{X}$和一个代理 $\mathrm{i} \in \mathcal{N}$，质心度量定义了其质心损失 $\ell_i^c(\mathcal{X}) \triangleq \mathrm{d}^c(\mathrm{i}, \mathrm{x}_{\mathcal{X}(\mathrm{i})})$，用于衡量代理 i 到其所分配中心的距离；非质心度量则定义了其非质心损失 $\ell_i^m(\mathcal{X}) = \max^{j\in C_{\mathcal{X}(i)}} \mathrm{d}^m(\mathrm{i}, \mathrm{j})$，用于衡量代理 i 到其簇中任意其他代理的最大距离。前者是质心聚类中使用的典型损失函数 [Chen 等，2019 年，Micha 和 Shah，2020年，Aziz 等人，2024年，Kellerhals 和 Peters， 2024] 而后者则是非质心聚类的一种损失函数，针对该函数已推导出颇具吸引力的公平性保证 [Caragiannis 等人， 2024]。

代理的双度量损失是质心部分和非质心部分之和：

$$\ell_i(\mathcal{X}) = \ell_i^m(\mathcal{X}) + \ell_i^c(\mathcal{X}) = \max_{j\in C_{\mathcal{X}(i)}} d^m(i,j) + d^c(i, x_{\mathcal{X}(i)}).$$

值得注意的是，添加缩放因子，使得 $\ell_i(\mathcal{X}) = \alpha \cdot \ell_i^m(\mathcal{X}) + \beta \cdot \ell_i^c(\mathcal{X})$ 对于某些 $\alpha, \beta \geqslant 0$ 成立，并不会使该族更加一般；相反，可以直接将 $d^m = \alpha \cdot d^m$ 和 $d^c = \beta \cdot d^c$ 分别作为新的非质心度量和质心度量。

加权单一度量损失：双度量损失的一个特殊情况是，质心损失和非质心损失都使用同一个度量 d 的（缩放版本）：$(\mathcal{N} \cup \mathcal{M}) \times (\mathcal{N} \cup \mathcal{M}) \to \mathbb{R}_{\geqslant 0}$。也就是说，对于某个 $\lambda \in [0,1]$，我们有 $\mathrm{d}^m(\mathrm{i}, \mathrm{j}) = \lambda \cdot \mathrm{d}(\mathrm{i}, \mathrm{j})$，对所有 $\mathrm{i}, \mathrm{j} \in \mathcal{N}$均成立；并且 $\mathrm{d}^c(\mathrm{i}, \mathrm{x}) = (1-\lambda) \cdot \mathrm{d}(\mathrm{i}, \mathrm{x})$，对所有 $\mathrm{i} \in \mathcal{N}$和 $\mathrm{x} \in \mathcal{M}$均成立，因此总体损失为

$$\ell_i(\mathcal{X}) = \lambda \cdot \max_{j\in C_{\mathcal{X}(i)}} d(i,j) + (1-\lambda) \cdot d(i, x_{\mathcal{X}(i)}).$$

再次注意，使用任意非负缩放因子 α 和 β 代替 λ 和 $1-\lambda$ 并不能带来进一步的推广，因为等效地可以使用 $\lambda = \frac{\alpha}{\alpha+\beta}$。

很容易看出，非质心聚类和质心聚类分别是加权单度量损失（而加权单度量损失又是双度量损失的特例）在 $\lambda = 1$ 和 $\lambda = 0$ 下的特例。因此，我们为双度量损失和加权单度量损失设计的半质心聚类算法可用于质心聚类和非质心聚类。

[1] 更准确地说，这些是伪度量。伪度量 $\mathrm{d}\colon \mathrm{X} \times \mathrm{X} \to \mathbb{R}_{\geqslant 0}$ 满足：对于所有 $\mathrm{x} \in \mathrm{X}$，$\mathrm{d}(\mathrm{x}, \mathrm{x}) = 0$；对于所有 $\mathrm{x}, \mathrm{y} \in \mathrm{X}$，$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \mathrm{d}(\mathrm{y}, \mathrm{x})$；并且对于所有 $\mathrm{x}, \mathrm{y}, \mathrm{z} \in \mathrm{X}$，$\mathrm{d}(\mathrm{x}, \mathrm{y}) \leqslant \mathrm{d}(\mathrm{x}, \mathrm{z}) + \mathrm{d}(\mathrm{z}, \mathrm{y})$。

2.3 比例公平性保证

我们重点研究先前工作着重探讨的两种比例公平准则。

定义 1（核）。对于 $\alpha \geqslant 1$，若聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$ 属于 α-核，则不存在代理群组 $S \subseteq \mathcal{N}$，满足 $|S| \geqslant n/k$，且存在可行中心 $y \in \mathcal{M}$，使得对于所有 $i \in S$，$\alpha \cdot \ell_i(S, y) < \ell_i(\mathcal{X})$（若出现这种情况，则称联盟 S 以中心 y 偏离）。

换言之，任何人数至少为 S n/style id='3'>$/k$ 的代理群组，如果按比例应当获得一个代表它的聚类中心，就不能通过单方面成为一个拥有可行中心 y 的独立簇来改善自身状况。

第二项标准是一种放宽形式，它对偏离联盟提出了更高的门槛。

定义 2（完全正当代表性（FJR））。对于 $\alpha \geqslant 1$，若聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$ 是α-FJR，则不存在代理群组 $S \subseteq \mathcal{N}$，满足 $|S| \geqslant n/k$，且存在可行中心 $y \in \mathcal{M}$，使得对于所有 $i \in S$，$\alpha \cdot \ell_i(S,y) < \min_{j \in S} \ell_j(\mathcal{X})$。

也就是说，对于偏离联盟中的每个成员，乘以 α 后的新损失不仅应小于其偏离前自身的损失，还应小于任何偏离成员偏离前的损失。因此显然，对于所有 $\alpha \geqslant 1$，α-FJR 是 α-核心的一个放宽。

命题 1. 对于每个 $\alpha \geqslant 1$，α-核心中的每种聚类也都是 α-*FJR*。

证明。反设结论不成立。对于某个实例，存在一种聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$，以及一个 $\alpha \geqslant 1$，使得 $\mathcal{X}$ 属于 α-核心，但不满足 α-FJR。根据 FJR 的定义，这意味着存在一个代理群组 $S \subseteq N$，满足 $|S| \geqslant n/k$，以及一个中心 $y \in \mathcal{M}$，使得对于所有 $i \in S$，$\alpha \cdot \ell_i(S, y) < \min_{j \in S} \ell_j(\mathcal{X})$。

容易看出，对于每个 $i \in S$，$\alpha \cdot \ell_i(S, y) < \min_{j \in S} \ell_j(\mathcal{X})$ 意味着 $\alpha \cdot \ell_i(S, y) < \ell_i(\mathcal{X})$，也就是说，$(S, y)$ 同样会导致一次 α-核心违反，从而产生矛盾。 □

3 贪心捕获算法

算法 1： 质心贪心捕获
[Chen 等人， 2019]

输入： $\mathcal{N}$， $\mathcal{M}$， d， k；
输出： $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$。
初始化： $\mathcal{N}' \leftarrow \mathcal{N}, \mathcal{X} \leftarrow \emptyset, \delta \leftarrow 0$
// 仍有未捕获代理
当 $\mathcal{N}' \neq \emptyset$ **执行时**
  平滑地增加 δ
  // 球在某个中心处打开
  如果 $\exists x \in \mathcal{M}$， $C \subseteq \mathcal{N}'$ 使得 $|C| \geqslant n/k$ 且 $d(i, x) \leqslant \delta$ 对所有 $i \in C$ **则**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{(C, x)\}$
    $\mathcal{N}' \leftarrow \mathcal{N}' \setminus C$
  end
  // 开球不断扩大
  若 $\exists (C, x) \in \mathcal{X}, i \in \mathcal{N}'$ 使得 $d(i, x) \leqslant \delta$ **then**
    $(C, x) \leftarrow (C \cup \{i\}, x)$
    $\mathcal{N}' \leftarrow \mathcal{N}' \setminus \{i\}$
  end
end
// 代理切换簇
$P \leftarrow \{x \in \mathcal{M} : (C, x) \in \mathcal{X}$ 对于某个 $C\}$
$\forall i \in \mathcal{N}, x_i \in \arg\min_{x \in P} d(i, x)$
$\forall x \in P, \hat{C}_x \leftarrow \{i \in \mathcal{N} : x_i = x\}$
$\mathcal{X} \leftarrow \{(\hat{C}_x, x) : x \in P\}$
返回 $\mathcal{X}$

算法 2： 非质心贪心捕获
[Caragiannis 等人， 2024]

输入： $\mathcal{N}$， d， k；
输出： $\mathcal{X} = \{C_1, \ldots, C_k\}$。
初始化： $\mathcal{N}' \leftarrow \mathcal{N}$、 $\mathcal{X} \leftarrow \emptyset$、 $\delta \leftarrow 0$
// 未捕获代理仍然存在
当 $\mathcal{N}' \neq \emptyset$ **时执行**
  平滑地增加 δ
  // 球在代理处打开
  如果 $\exists i \in \mathcal{N}'$， $C \subseteq \mathcal{N}'$ 使得 $|C| \geqslant$ 最小化$(|\mathcal{N}'|, n/k)$ 并且 $d(i, i') \leqslant \delta$ 对所有 $i' \in C$ **则**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{C\}$
    $\mathcal{N}' \leftarrow \mathcal{N}' \setminus C$
  end
  // 开球不会增长
end
// 代理不会切换簇
返回 $\mathcal{X}$

在比例公平聚类的既有研究中，贪心捕获算法一直是实现质心世界和非质心世界中比例公平准则常数近似的首选方法。Chen 等人 [2019] 将其用于质心聚类，而Caragiannis 等人 [2024] 则将其调整为非质心聚类。质心变体和非质心变体分别以算法 1和2的形式正式给出，二者有许多相似之处。在这两种变体中，每个代理最初都处于“未捕获”状态，同时若干个球以相同的速率开始扩张。一旦某个球覆盖了至少n/k 个未捕获代理，它就会捕获这些代理并变为“开放”（在质心变体中，开放球的中心还会被加入算法选取的至多k 个聚类中心之中）。但这两种变体在三个关键方面有所不同。

1. 球在哪里扩张：在质心版本中，每个可行簇中心周围都会有一个球扩张。在没有中心的非质心版本中，每个代理周围都会有一个球扩张。2. 开放球是否继续扩张：在质心变体中，球开放后仍会继续扩张，并在覆盖任何未捕获代理时立即将其捕获。在非质心变体中，球一旦开放便停止扩张。这很直观，因为在质心范式中，开放球扩张并捕获远处的代理，不会使先前已捕获代理的情况变差；但在非质心范式中则会如此。3. 代理最终是否会更换簇：在质心变体中，算法只返回一组（至多k 个）聚类中心，每个代理的损失就是它到最近聚类中心的距离。请注意，这个中心可能不是捕获该代理的球的中心，因为算法之后可能又添加了一个更近的中心。因此，某个球开放时捕获的代理集合未必构成单个簇。

在非质心变体中，最后不存在“切换”，而且一个球捕获的代理集合共同构成一个簇。这同样是因为，允许某个代理切换到她更偏好的另一个簇，可能会显著恶化该簇中现有代理的情况。

Chen 等人 [2019] 证明，在质心范式中，质心贪心捕获产生的聚类属于 $(1+\sqrt{2})$-核心；而 Caragiannis 等人 [2024] 证明，在非质心范式中，非质心贪心捕获产生的聚类属于 2-核心。这自然引出了一个问题：在半质心聚类范式中，某种贪心捕获变体能否对核心实现常数近似？

遗憾的是，我们可以证明，解决半质心范式并没有那么简单。要理解其中原因，只需考虑加权单度量损失这一受限情形；在该情形中，损失函数由单一度量d 和参数 $\lambda \in [0,1]$刻画。注意，贪心捕获算法的输入仅为d ，而不是 λ。因此，若（某种贪心捕获变体）要针对加权单度量损失实现 α-核心（对于任意 $\lambda \in [0,1]$），它必须同时针对非质心损失（$\lambda = 1$）和质心损失（$\lambda = 0$）实现 α-核心。

对于通过改变上述三个二元维度而产生的八种贪心捕获变体，可以逐一论证它们都无法同时为质心损失和非质心损失实现对核心的任何有限近似（我们会在附录A节讨论其中一些变体）。我们的第一个结果表明，事实上，没有任何算法能够实现这样的同时保证。

定理 1. 对于每个 $\alpha \geqslant 1$ 和 $\beta \geqslant 1$，都存在一个半质心聚类实例，其中不存在任何一种聚类，能够同时就质心损失而言属于 α-核，并就非质心损失而言属于 β-核。

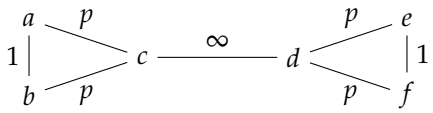

图 1：一个半质心聚类实例，其中任何聚类都无法同时就质心损失和非质心损失而言，为核心提供有限近似。

定理 1. 对于每个 $\alpha \geqslant 1$ 和 $\beta \geqslant 1$，都存在一个半质心聚类实例，其中不存在任何一种聚类，能够同时就质心损失而言属于 α-核，并就非质心损失而言属于 β-核。

证明。 任取 $\alpha \geqslant 1$， $\beta \geqslant 1$，以及 $p > \max\{\alpha, \beta\}$。考虑 图 1中的实例。这里 $\mathcal{N} = \{a、b、c、d、e、f\}$、$\mathcal{M} = \mathcal{N}$ $= 3$ $^{m} =$ $^{c} =$ [2] 、k，并令 ddd，其中 d 由图中所示的距离定义，未标出的距离均假设为三角不等式允许的最大值。由于 $n/k = 2$，偏离联盟必须包含至少 2 个代理。

反设该实例中存在一种聚类，同时属于质心范式下的α-核心以及非质心范式下的 β-核心。

非质心 β-核心：我们断言 $\{a, b\}$ 必须是一个簇。否则，a 和 b 要么分别与不同代理处于同一个簇中，要么至少有一个是单元素簇。在前一种情况下， $\{a, b\}$ 可以偏离，使每个成员的非质心损失都改善 $p > \beta$ 倍，这与假设矛盾。在后一种情况下，由于 $k = 3$，根据鸽巢原理，必定存在一个簇 C，使得 $|C| \geqslant 3$ （这意味着 $C \cap \{d, e, f\} \neq \emptyset$）。如果 $C \cap \{a, b, c\} \neq \emptyset$，那么 $C \cap \{a, b, c\}$ 和 $C \cap \{d, e, f\}$ 中较大的那个簇必定至少包含两个代理；该簇可以偏离，且其每个成员都能获得无限倍的改善，这再次导致矛盾。唯一剩下的可能是 $C = \{d, e, f\}$，在这种情况下， $\{e, f\}$ 可以偏离，使每个成员改善 $p > \beta$ 倍，这与假设矛盾。

因此， $\{a, b\}$ 必定是其中一个簇。根据对称性， $\{e, f\}$ 也必定是一个簇，因此剩下的簇必定是 $\{c, d\}$。由此，在非质心范式下，唯一可能属于 β-核心的聚类是$\{(\{a, b\}, x_1), (\{c, d\}, x_2), (\{e, f\}, x_3)\}$， 其中某个 $x_1, x_2, x_3 \in \mathcal{N}$。

[2]无穷距离 可以用非常大的距离代替，而不影响证明；不过，为简单起见，我们仍将它们设为无穷大 .

质心 α-核心：如果 $x_1 \notin \{a, b\}$，那么 $\{a, b\}$ 可以选择 a 或 b 作为聚类中心进行偏离，并使各自的质心损失至少改善因子 $p > \alpha$，这就产生了矛盾。不失一般性，假设 $x_1 = a$。同理，不失一般性地假设 $x_3 = e$。现在，要么 $x_2 \in \{a, b, c\}$，要么 $x_2 \in \{d, e, f\}$。在前一种情况下，$\{d, f\}$ 可以偏离并将 f 作为中心；在后一种情况下，$\{b, c\}$ 可以偏离并将 b 作为中心。两种情况下，每个成员的损失都至少改善因子 $p > \alpha$，因此产生矛盾。

证明到此结束。 □

4 核

根据 定理 1，我们将注意力转向不只是将单一度量 d 作为输入的规则；也就是说，对于加权单度量损失，它们还将 λ 作为输入；对于更一般的双度量损失，它们则将质心度量和非质心度量 d^c 和 d^m作为输入。

在第 4.1 节中，我们介绍双度量损失的结果。随后，在第 4.2 节中，我们介绍加权单度量损失这一受限情形的改进。

4.1 双度量损失

以下两项是我们工作中最重要、也最复杂的结果。

定理 2. 对于双度量损失，始终存在一个属于3-核心的聚类。

定理 3. 对于双度量损失，可以在多项式时间内计算出一个属于$(3+2\sqrt{3})$-核心的聚类。

为了理解算法 3，我们首先需要介绍最紧密簇问题。该问题由Caragiannis 等人 [2024]针对非质心聚类提出。将其调整为我们的半质心聚类场景后，该问题只需找到簇(C,x)，其中 $|C| \geqslant n/k$，使簇C中任一代理所承受的最大损失最小。对于任意损失函数，其定义和近似算法如下。

定义 3（α-MCC）。对于 $\alpha \geqslant 1$，给定代理人集合 $\mathcal{N}$、中心集合 $\mathcal{M}$、损失函数 ℓ 和阈值 θ（通常为 n/k），若簇 $(C \subseteq \mathcal{N}, x \in \mathcal{M})$ 满足 $|C| \geqslant \min(|\mathcal{N}|, \theta)$，则称其为一个 α- 近似最紧密簇（简称 α-MCC 簇），当且仅当对于每个满足 $|S| \geqslant \min(|\mathcal{N}|, \theta)$ 的簇 $(S \subseteq \mathcal{N}, y \in \mathcal{M})$，都有 $\max_{i \in C} \ell_i(C, x) \leqslant \alpha \cdot \max_{i \in S} \ell_i(S, y)$。当 $\alpha = 1$ 时，它简称为最紧密簇。

Caragiannis 等人 [2024] 使用此方法替代贪心捕获子程序；该子程序用于找到半径最小的簇 (C,x)，其半径为 $\max_{i \in C} d(i,x)$，以获得更好的公平性保证。 定理 2 和 3均通过使用 算法 3得到，该算法调用一个用于寻找 α-MCC 簇的子程序；定理 2 通过一个低效的子程序得到，该子程序用于寻找 MCC 簇；而 定理 3 则通过嵌入我们设计的高效子程序来计算 4-MCC 簇。

算法描述： 算法 3 分为两个阶段。阶段 1 通过迭代寻找关于双度量损失的 α-MCC 簇，并将其添加到暂定解中。我们将阶段 1 找到的聚类记为 $\hat{\mathcal{X}} = \{(\hat{C}_1, x_1), \ldots, (\hat{C}_k, x_k)\}$。对于每个簇 $(\hat{C}_t, x_t) \in \hat{\mathcal{X}}$，算法将该簇中任一代理的最大损失存储为 $r_t = \max_{i \in \hat{C}_t} \ell_i(\hat{C}_t, x_t)$。

阶段 2 以聚类 $\hat{\mathcal{X}}$ 为起点，并允许每个代理$i \in \mathcal{N}$ 在满足两项审慎约束的前提下，从其第t个簇切换到任意第t'个簇：

- 限制新簇中代理的损失：我们知道，每个$j \in C_{t'}$ 的原始损失至多为$r_{t'}$。如果代理i切换到该簇，它可能会使j的损失增加到$d_c(j, x_{t'}) + d_m(j, i)$。我们要求该值至多为$c \cdot r_{t'}$，其中常数$c$ 作为输入提供给算法。

算法 3： 双度量算法

输入： 代理人集合 $\mathcal{N}$，可行簇中心集合 $\mathcal{M}$，非质心度量 d^m，质心度量 d^c，中心数量 k，增长因子 $c \geqslant 1$，MCC近似因子 α

输出： 聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$

初始化：$\mathcal{N}' \leftarrow \mathcal{N}$，$\widehat{\mathcal{X}} \leftarrow \emptyset$，$t \leftarrow 1$；

// 阶段 1：构建一个暂定聚类 $\widehat{\mathcal{X}}$

当 $\mathcal{N}' \neq \emptyset$ **时** // 未捕获代理仍然存在

  // 找到一个 α-MCC 簇 $(\widehat{C}_t, x_t)$ 并将其最大损失保存为 r_t

  $(\widehat{C}_t, x_t) \leftarrow$ 一个来自 $\mathcal{N}', \mathcal{M}$ 的 α-最紧密簇（阈值为 n/k）；

  $r_t \leftarrow$ 最大值 $_{i \in \widehat{C}_t} \ell_i(\widehat{C}_t, x_t)$；

  // 将该簇添加到 $\widehat{\mathcal{X}}$ 并将其代理标记为已捕获

  $\widehat{\mathcal{X}} \leftarrow \widehat{\mathcal{X}} \cup \{(\widehat{C}_t, x_t)\}$；

  $\mathcal{N}' \leftarrow \mathcal{N}' \setminus \widehat{C}_t$；

  $t \leftarrow t + 1$；

结束

// 阶段 2：有选择地允许代理切换到更优的簇

$\forall t \in [k], C_t \leftarrow \widehat{C}_t$；

对于 $i \in \mathcal{N}$ **执行**

  $t \leftarrow \widehat{\mathcal{X}}(i)$；

  // 簇：代理 i 可以切换到这些簇，而不会对其成员造成太大影响

  $V_i \leftarrow \{t' \in [k]: \quad \forall j \in \widehat{C}_{t'}, \ d^c(j, x_{t'}) + d^m(j, i) \leqslant c \cdot r_{t'}\}$；

  // 其中，代理 i 选择一个使其损失上界最小的簇

  $t^* \leftarrow \arg\min_{t' \in V_i} \left(d^c(i, x_{t'}) + c \cdot r_{t'} + \min_{j \in \widehat{C}_{t'}} (d^m(i, j) - d^c(j, x_{t'})) \right)$；

  // 如果这样能够改善其损失的上界，则切换到该方案

  如果 $d^c(i, x_{t^*}) + c \cdot r_{t^*} + \min_{j \in \widehat{C}_{t^*}} (d^m(i, j) - d^c(j, x_{t^*})) < c \cdot r_t$ **则**

    $C_t \leftarrow C_t \setminus \{i\}$；

    $C_{t^*} \leftarrow C_{t^*} \cup \{i\}$；

  end

结束

返回 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$

• 有界自身损失：我们证明，算法 3 中的表达式——dc(i, xt′) + c · rt′ + minj∈ˆCt′(dm(i, j) − dc(j, xt′))——作为代理 i 切换到第 t′ 个簇时损失的上界，无论在阶段 2 期间还有哪些其他代理切换到第 t′ 个簇。这确保了代理 i 是否切换的决定独立于阶段 2 中其他代理所作的决定，并且原本位于 Ct 中的代理 i 的损失在整个阶段 2 期间始终被 c · rt 限制，无论阶段 2 作出何种决定。只有当新的损失上界低于 c · rt 时，代理 i 才会进行切换。

建立近似核保证的关键在于阶段 2 中实施的这种选择性切换。直观地说，这可以看作是在质心贪心捕获与非质心贪心捕获之间取得平衡：质心贪心捕获允许代理切换到任意其他簇，因为这不会增加该簇中代理的质心损失；而非质心贪心捕获则完全禁止此类切换，以免损害被切入簇中的代理。增长因子 c 谨慎地控制着这种平衡，对该因子进行最优设置后，我们可以得到以下上界。

引理 1。 给定一个用于计算 α-近似最紧密簇的子程序，算法 3在增长因子设为 $c = \frac{3\alpha+\sqrt{\alpha(\alpha+8)}}{4\alpha}$ 时，可找到一个关于双度量损失的 $12(\alpha + \sqrt{\alpha(\alpha+8)} + 2)$-核心中的聚类。

我们首先证明 算法 3的若干有用性质。这些性质界定了代理在算法 3的阶段 2 期间切换簇（或选择不切换簇）时所产生的损失，无论切换的是代理自身还是其他代理。

回想一下，阶段 1 会生成一个暂定聚类 $\widehat{\mathcal{X}}$，其中 $r_t = \max_{i\in\widehat{C}_t} \ell_i(\widehat{C}_t, x_t)$ 是第 t 个暂定簇 $(\widehat{C}_t, x_t)$中的最大损失。接着，在阶段 2 中，每个代理 $i \in \widehat{C}_t$ 都有机会切换到每个 $t' \in V_i$对应的第 t' 个簇。对于每个这样的簇，该代理都会计算以下表达式

$$\Phi(i,t') \triangleq d^c(i, x_{t'}) + c \cdot r_{t'} + \min_{j\in\widehat{C}_{t'}}(d^m(i,j) - d^c(j, x_{t'})),$$

选取其最小值点 $t^* \in \arg\min_{t'\in V_i} \Phi(i,t')$，并在 $\Phi(i,t') < c \cdot r_t$ 时切换到该点。以下论断有助于阐明这一步背后的推理。

第一个引理说明，如果某个代理在阶段 1 和阶段 2 结束时都属于第 t 个簇（即它在阶段 2 期间没有切换簇），那么其损失在阶段 1 结束时至多为 r_t，而在阶段 2 结束时仍有界于 $c \cdot r_t$。损失的增加可能是因为其他代理在阶段 2 期间切换到了第 t 个簇。

引理 2. 对于每个 $i \in \mathcal{N}$，若 $\widehat{\mathcal{X}}(i) = \mathcal{X}(i) = t$，则 $\ell_i(\mathcal{X}) = \ell_i(C_t, x_t) \leqslant c \cdot r_t$。

证明。 我们的目标是证明，对于所有 $j \in C_t$，都有 $d^c(i,x_t) + d^m(i,j) \leqslant c \cdot r_t$。任取一个 $j \in C_t$。

情况 1：根据 r_t 的定义，我们必须有$d^c(i, x_t) + d^m(i, j) \leqslant r_t \leqslant c \cdot r_t$，其中最后一个不等式成立是因为$c \geqslant 1$。换言之，这样的代理甚至不会使代理$i$ 的损失增加一点点。

情况 2：由于代理 j 被允许切换到第t个簇，我们必须有$t \in V_j$。然后，算法 3 中的算法 3 确保$d^c(i, x_t) + d^m(i, j) \leqslant c \cdot r_t$；这正是禁止代理加入某个簇的条件，如果加入该簇会显著损害该簇的“核心成员”。

证明到此结束。 □

下一个引理表明，如果代理i 在阶段 2*期间切换到第t^*个簇，则其最终损失 $\ell_i(\mathcal{X})$将由 $\Phi(i, t^*)$界定。

引理 3。 对于每个 $i \in \mathcal{N}$，如果 $\mathcal{X}(i) = t^* \neq \widehat{\mathcal{X}}(i)$，那么

$$\ell_i(\mathcal{X}) = \ell_i(C_{t^*}, x_{t^*}) \leqslant \Phi(i,t^*) = d^c(i, x_{t^*}) + c \cdot r_{t^*} + \min_{j\in\widehat{C}_{t^*}}(d^m(i,j) - d^c(j, x_{t^*})).$$

证明。令 $i' = \arg\max_{j \in C_{t^*}} d^m(i, j)$ 为 C_{t^*} 中距离 i 最远的代理。于是，我们有 $\ell_i(C_{t^*}, x_{t^*}) = d^c(i, x_{t^*}) + d^m(i, i')$。我们希望证明该值至多为 $\Phi(i, t^*)$。令 $j^* = \arg\min_{j \in \hat{C}_{t^*}} (d^m(i, j) - d^c(j, x_{t^*}))$，则 $\Phi(i, t^*) = d^c(i, x_{t^*}) + c \cdot r_{t^*} + d^m(i, j^*) - d^c(j^*, x_{t^*})$。让我们简化目标：

$$\begin{aligned} & d^c(i, x_{t^*}) + d^m(i, i') \leqslant d^c(i, x_{t^*}) + c \cdot r_{t^*} + d^m(i, j^*) - d^c(j^*, x_{t^*}) \\ \Leftrightarrow\ & d^m(i, i') \leqslant c \cdot r_{t^*} + d^m(i, j^*) - d^c(j^*, x_{t^*}). \end{aligned} \tag{1}$$

情况1：$i' \in \hat{C}_{t^*}$。然后，根据 $j^* \in \hat{C}_{t^*}$ 也成立这一事实，我们必有 $d^c(j^*, x_{t^*}) + d^m(j^*, i') \leqslant \ell_{j^*}(\hat{C}_{t^*}, x_{t^*}) \leqslant r_{t^*}$，从而得到 $d^m(j^*, i') \leqslant r_{t^*} - d^c(j^*, x_{t^*})$。

情况2：$i' \notin \hat{C}_{t^*}$。然后，由于代理 i' 在阶段 2 期间切换到了第 t^* 个簇，我们必有 $t^* \in V_{i'}$。算法 3 的算法 3 以及 $j^* \in \hat{C}_{t^*}$ 这一事实确保 $d^c(j^*, x_{t^*}) + d^m(j^*, i') \leqslant c \cdot r_{t^*}$，即 $d^m(j^*, i') \leqslant c \cdot r_{t^*} - d^c(j^*, x_{t^*})$。

由于 $c \geqslant 1$，我们在两种情况下都得到 $d^m(j^*, i') \leqslant c \cdot r_{t^*} - d^c(j^*, x_{t^*})$。利用三角不等式，我们得到

$$d^m(i, i') \leqslant d^m(i, j^*) + d^m(j^*, i') \leqslant d^m(i, j^*) + c \cdot r_{t^*} - d^c(j^*, x_{t^*}),$$

这就是所需的 公式 (1)，证明完毕。 □

现在我们知道了以下结论。在阶段 2 期间，如果代理 i 保持在其原始簇 $\hat{\mathcal{X}}(i) = t$ 中，其损失仍以上界 $c \cdot r_t$ (引理 2) 为界；如果它切换到簇 $t' \in V_i$，其损失仍以上界 $\Phi(i, t')$ 为界。这解释了为什么代理 i 只有在 $\Phi(i, t^*) < c \cdot r_t$ 时，才会切换到簇 $t^* \in \arg\min_{t' \in V_i} \Phi(i, t')$：代理只是在选择能够为其最终损失提供最小上界的簇。下一个引理对此进行了说明。

引理 4. 对于每个 $i \in \mathcal{N}$ 和 $t' \in V_i$，我们有 $\ell_i(\mathcal{X}) \leqslant \Phi(i, t') = d^c(i, x_{t'}) + c \cdot r_{t'} + \min_{j \in \hat{C}_{t'}} (d^m(i, j) - d^c(j, x_{t'}))$。

证明。令 $t = \hat{\mathcal{X}}(i)$，并令 $t^* \in \arg\min_{t' \in V_i} \Phi(i, t')$。如果代理 i 在算法 3 的阶段 2 期间切换到第 t^* 个簇，则根据以下事实，结论成立：对于所有 $t' \in V_i$，$\Phi(i, t^*) \leqslant \Phi(i, t')$。如果代理 i 在阶段 2 期间没有进行切换并仍留在第 t 个簇中，则根据算法 3 的算法 3，这必然是因为 $c \cdot r_t \leqslant \Phi(i, t^*)$。对于所有 $t' \in V_i$，结论同样由 $\Phi(i, t^*) \leqslant \Phi(i, t')$ 得出。

□

与引理 4的证明类似，我们可以看到，如果代理i 在t阶段 1 结束时位于第$c \cdot r_t$ 仍然是其最终损失的上界，无论代理是留在第t个簇中，还是在阶段 2 期间切换到第t^*个簇。这是因为代理只有在该簇能够为损失提供更小的上界时，才会切换到第t^*个簇。

引理 5。 对于每个 $i \in \mathcal{N}$，如果 $\hat{\mathcal{X}}(i) = t$，那么 $\ell_i(\mathcal{X}) \leqslant c \cdot r_t$。

证明。如果 $\mathcal{X}(i) = t$（即代理在阶段 2 期间留在第 t 个簇中），那么该结论可由引理 2 得出。如果 $\mathcal{X}(i) = t^* \neq t$，那么根据引理 3，有 $\ell_i(\mathcal{X}) \leqslant \Phi(i, t^*)$。然而，根据算法 3，只有当 $\Phi(i, t^*) < c \cdot r_t$ 时，代理 i 才会切换到第 t^* 个簇，这意味着 $\ell_i(\mathcal{X}) \leqslant c \cdot r_t$，正如所需。 □

综上，引理 4和5 为我们提供了两种不同的方式来给代理 i的损失给出上界，分别根据代理 t所在的第 i 个簇（该簇是它在阶段 1 结束时所属的簇），以及根据阶段 2 期间代理 t'在 V_i 中的第 i 个簇（代理有机会切换到这些簇）。有了这些界，我们就可以证明中心引理了。

证明。 引理 1 的证明

固定一个 $\alpha \geqslant 1$，并令 $\beta = \frac{1}{2}(\alpha + \sqrt{\alpha(\alpha + 8)} + 2)$。

为导出矛盾，假设算法返回的某个聚类 $\mathcal{X} = \{(C_1, x_1), \ldots\ldots, (C_k, x_k)\}$ 中存在某个偏离簇 (S, y)，使得对于所有 $i \in S$，都有 $\ell_i(S, y) < \frac{1}{\beta}\ell_i(\mathcal{X})$。令 $i = \arg\min_{i \in S} \hat{\mathcal{X}}(i)$ 为该偏离簇中在阶段 1 的算法 3 期间最先被捕获的代理。令 $\hat{\mathcal{X}}(i) = t$。

令 $j = \arg\max_{j\in S} \ell_j(S,y)$ 为 S 中在偏离簇的所有代理中损失最高的代理。

首先注意，必然有 $\ell_j(S,y) \geqslant \frac{1}{\alpha} r_t$。否则，这将与以下事实矛盾：在算法的该步骤中，$(\widehat{C}_t, x_t)$ 是最紧密簇的 α-近似。

接下来，我们可以证明 $\widehat{\mathcal{X}}(j) \neq t$，并且在算法的第二阶段中，$j$ 没有机会转入第 t 个簇。

证明 $\widehat{\mathcal{X}}(j) \neq t$ 很容易。若 $\widehat{\mathcal{X}}(j) = t$ 成立，那么根据引理 5，必有 $\ell_j(\mathcal{X}) \leqslant c \cdot r_t$。由于 j 想要转而选择 (S, y)，因此也必有 $\ell_j(S,y) < \frac{1}{\beta}\ell_j(\mathcal{X}) \leqslant c\beta r_t$。代入 β 和 c 的正确取值后，可以看出只要 $\alpha \geqslant 1$，就有 $c_\beta < \frac{1}{\alpha}$，这与 $\ell_j(S,y) \geqslant \frac{1}{\alpha} r_t$ 矛盾。

再稍作推导，我们还可以证明，在算法的第二阶段，j 没有机会切换到第 t 个簇。如果 j 确实有这个机会，那么根据引理 4，我们必有 $\ell_j(\mathcal{X}) \leqslant d^c(j, x_t) + c \cdot r_t + \min_{i'\in \widehat{C}_t}\{d^m(i', j) - d^c(i', x_t)\}$。由于 $i \in \widehat{C}_t$，这还意味着 $\ell_j(\mathcal{X}) \leqslant d^c(j, x_t) + c \cdot r_t + d^m(i, j) - d^c(i, x_t)$。

根据三角不等式，我们有 $d^c(j, x_t) \leqslant d^c(j, y) + d^c(i,y) + d^c(i, x_t)$。将其代入对 j 的损失所作的界中，可得：

$$\ell_j(\mathcal{X}) \leqslant c \cdot r_t + d^m(i,j) + d^c(j,y) + d^c(i,y)$$

注意，由于 i 和 j 都属于 S，$d^m(i, j) + d^c(j, y) \leqslant \ell_j(S, y)$。此外，我们必有 $d^c(i, y) \leqslant \ell_i(S, y) < \frac{1}{\beta}\ell_i(\mathcal{X}) \leqslant c\beta r_t$，其中最后一个不等式由引理 5 得出。将这些代入不等式，可得：

$$\ell_j(\mathcal{X}) < \ell_j(S,y) + \left(c + \frac{c}{\beta}\right) \cdot r_t.$$

这意味着，当j 偏离到(S,y)时，它最多只能将自身损失改善以下因子：

$$\frac{\ell_j(S,y) + \left(c + \frac{c}{\beta}\right) \cdot r_t}{\ell_j(S,y)}.$$

$\ell_j(S,y) \geqslant \frac{1}{\alpha} r_t$ 这一事实使我们能够将其上界限定为

$$\frac{\frac{1}{\alpha} \cdot r_t + \left(c + \frac{c}{\beta}\right) \cdot r_t}{\frac{1}{\alpha} \cdot r_t} = 1 + c \cdot \alpha \cdot \left(1 + \frac{1}{\beta}\right).$$

将c 和 β 用 α表示的值代入后，结果为 $12(\alpha + \sqrt{\alpha(\alpha+8)} + 2)$，这就产生了矛盾。

因此，j 必定没有机会在算法的第二阶段切换到第 t 个簇。所以，必定存在某个 $i' \in \widehat{C}_t$，使得 $d^c(i', x_t) + d^m(i', j) > c \cdot r_t$，重新排列后可得 $d^m(i', j) > c \cdot r_t - d^c(i', x_t)$。作为证明的最后一部分，我们可以证明这同样会导致矛盾。

根据三角不等式，我们有 $d^m(i', j) \leqslant d^m(i', i) + d^m(i, j)$。重新整理可得 $d^m(i, j) \geqslant d^m(i', j) - d^m(i', i)$。将 $d^m(i', j) > c \cdot r_t - d^c(i', x_t)$ 代入该不等式，得到 $d^m(i, j) > c \cdot r_t - (d^c(i', x_t) + d^m(i', i))$。由于 i 和 i′ 都属于 $\widehat{C}_t$，因此 $d^c(i', x_t) + d^m(i', i) \leqslant \ell_{i'}(\widehat{C}_t, x_t) \leqslant r_t$，由此可得 $d^m(i, j) > (c-1)r_t$。由于 i 和 j 都属于 S，这说明 $\ell_i(S, y) \geqslant d^m(i, j) > (c-1)r_t$。

最后，由于$i \in \widehat{C}_t$，根据引理 5，我们知道 $\ell_i(\mathcal{X}) \leqslant c \cdot r_t$。这意味着，当$i$ 偏离到$(S、y)$时，它们的损失改善幅度至多为$\frac{c \cdot r_t}{(c-1)r_t} = \frac{1}{2}(\alpha + \sqrt{\alpha(\alpha+8)} + 2)$，这又导出了一个矛盾，从而完成了证明。

□

根据引理 1，定理 2 的证明便可轻松推出。

定理 2的证明。 令 $\alpha = 1$ （即令算法 3 在阶段 1 中迭代计算最紧密簇），代入引理 1，将增长因子设为$c = 3/2$，并得到核心的 3-近似。 □

虽然定理 2 保证了 3-核中存在一种聚类，但其计算依赖于迭代寻找最紧密簇，而在非质心范式下，这一问题已是 NP 困难 [Caragiannis 等人， 2024]。幸运的是，引理 1 表明，即使在算法 3 的阶段 1中，我们只能找到近似最紧密簇，也仍能获得公平性保证。

接下来，我们将展示如何在多项式时间内计算一个 4-近似最凝聚簇。算法 4 的工作方式如下。对于每个可行簇中心 $y \in \mathcal{M}$，它使用精心构造的距离度量 d^y，运行非质心贪心捕获；该距离度量对所有 $i, j \in \mathcal{N}$ 定义为 $d^y(i, j) = d^m(i, j) + d^c(i, y) + d^c(j, y)$，直至得到第一个簇 C^y。随后，它返回最大损失最低的 (C^{y^*}, y^*)，即 $y^* \in \arg\min_{y \in \mathcal{M}} \max_{i \in C_y} \ell_i(C^y, y)$。

算法 4： 4-近似最凝聚聚类算法

输入： 代理集合 $\mathcal{N}$，可行中心集合 $\mathcal{M}$，非质心度量 d^m，质心度量 d^c，簇数 k

输出： 簇 (C, x)

对于 $y \in \mathcal{M}$ **执行**
- $\forall i, j \in \mathcal{N}: d^y(i, j) \leftarrow d^m(i, j) + d^c(i, y) + d^c(j, y)$; // 构造度量
- $C^y \leftarrow$ 由算法 2 在输入 $(\mathcal{N}, d^y, k)$下生成的第一个簇; // 运行贪心捕获
- $r^y \leftarrow$ 最大值$_{i \in C_y} \ell_i(C^y, y)$; // ℓ_i = 双度量损失

结束

$y^* \leftarrow \arg\min_{y \in \mathcal{M}} r^y$; // 最佳中心

返回 (C^{y^*}, y^*)

引理 6. 对于 *du al* 度量损失，算法 *4* 可在多项式时间内计算出一个 4-近似的最紧密簇 .

证明。对于任意实例，固定某个 $y \in \mathcal{M}$，并令 d^y 为 $\mathcal{N}$ 上的一个度量距离函数，其中对于任意不同的 $i、j \in \mathcal{N}$，$d^y(i, j) = d^m(i, j) + d^c(i, y) + d^c(j, y)$（若 $i = j$，则令 $d^y(i, j) = 0$）。

我们首先可以证明，d^y 确实在 $\mathcal{N}$ 上诱导出一个度量空间。为此，只需证明三角不等式成立，因为显然 d^y 满足另外两个度量公理。

对于任意 $i, i', j \in \mathcal{N}$，应有 $d^y(i, j) \leqslant d^y(i, i') + d^y(i', j)$。根据 d^y 的定义，这等价于说 $d^m(i, j) + d^c(i, y) + d^c(j, y) \leqslant d^m(i, i') + d^c(i, y) + d^c(i', y) + d^m(i', j) + d^c(i', y) + d^c(j, y)$。由于 d^m 是一个度量，我们有 $d^m(i, j) \leqslant d^m(i, i') + d^m(i', j)$。因此，只需证明 $d^c(i, y) + d^c(j, y) \leqslant d^c(i, y) + d^c(i', y) + d^c(i', y) + d^c(j, y)$。显然这也成立，因为 $d^c(i, y) + d^c(j, y)$ 出现在不等式的右侧。

算法 4 对代理运行非质心贪心捕获算法，使用距离度量d^y，对每个 $y \in \mathcal{M}$，然后在原始的对偶度量设定中，将该簇视为以 y 为中心，并从中选出找到的最紧密簇。

为导出矛盾，假设算法 4 无法找到最紧密簇的 4-近似。设(S, y) 是向算法 4输入的某个实例中最紧密的簇。定义$r = \max_{i \in S} \ell_i(S, y)$为$S$中任一代理所承受的最大损失。假设算法 4 返回簇$(C^*, x^*)$，其中$r^* = \max_{i \in C^*} \ell_i(C^*, x^*) > 4r$。

取距离度量d^y，并令C^y 为算法 4 使用d^y 作为其距离函数时找到的非质心簇。首先，注意到 $\max_{i \in C_y} \ell_i(C^y, y) \geqslant r^*$。否则，算法 4 就会返回$(C^y, y)$，而不是$(C^*, x)$。

接下来，考虑成员集合 S 中任意代理相对于 d^y 的最大非质心损失。我们有 $\forall i \in S$，$\max^{j \in S} d^y(i, j) = \max_{j \in S} \{d^m(i, j) + d^c(i, y) + d^c(j, y)\} \leqslant \max^{j \in S} d^m(i, j) + d^c(i, y) + \max^{j \in S} d^c(j, y) \leqslant r + \max^{j \in S} d^c(j, y)$。对于所有 $j \in S$，我们必须有 $d^c(j, y) \leqslant r$，因此得到 $\max^{j \in S} d^y(i, j) \leqslant 2r$。

Caragiannis 等人 [2024]证明，在任何非质心聚类实例中，算法 2 都能以非质心损失为准则，对该实例中最紧密簇实现 2-近似。由于 C_y 是通过在 $\mathcal{N}$ 上运行算法 2，并使用距离函数 d_y得到的，因此 C_y 必定是该实例中最紧密非质心簇的 2-近似。具体而言，这意味着 C_y 中任意代理感受到的相对于 d_y 损失的最大非质心损失，不超过成员集合 $C' \subseteq \mathcal{N}$ （其中 $|C'| \geqslant n/k$）中任意代理所感受到的最大此类损失的两倍。对于 S，这意味着 $\max_{i,j\in C_y} d_y(i,j) \leqslant 2\max_{i,j\in S} d_y(i,j) \leqslant 2(2r) = 4r$。

最后请注意，对于任意代理成员集合 C，该成员集合中任一代理 $i \in C$ 在度量 d_y 下所感受到的最大非质心损失，都将是代理 i 在簇 (C, y) 下所感受到的对偶度量损失的一个上界。为此，只需注意，i 在 C 下相对于 d_y 的非质心损失等于 $\max_{j\in C} d_y(i,j) = \max_{j\in C}(d^m(i,j) + d^c(i,y) + d^c(j,y)) \geqslant \max_{j\in C} d^m(i,j) + d^c(i,y) = \ell_i(C,y)$。在 C_y的情况下，这告诉我们 $r^* \leqslant \max_{i\in C_y} \ell_i(C_y,y) \leqslant \max_{i\in C_y}\max_{j\in C_y} d_y(i,j) \leqslant 4r$，由此得出矛盾，证明至此完成。 □

插件　将其转化为引理 1使我们能够在多项式时间内得到一个常数近似的核心。

定理 3。 对于双度量损失，可以在多项式时间内计算出一个属于$(3+2\sqrt{3})$-核心的聚类。

定理 *3* 的证明定理 *3*。 根据 引理 6， 算法 4 计算出一个 4-MCC 簇。令 算法 3 在阶段 1 中反复使用该算法，并按照 $\alpha = 4$ 在 引理 1中的设定。然后，将增长因子设为 $c = 3+\sqrt{3}$，由此得到一个 $12(4+\sqrt{48+2}) = 3+2\sqrt{3}$ 核心近似。每次调用 算法 4 都在多项式时间内运行，而显然 算法 3 的其余部分也在多项式时间内运行。 □

下界： 人们可能会想知道这些近似效果有多好。如前所述，半质心聚类比质心聚类和非质心聚类都更具挑战性。具体而言，只需令 $d^m = 0$ 或 $d^c = 0$ （距离度量为零意味着所有成对距离均为零），便可直接从任一范式引入下界；这样，双度量损失就分别与质心损失或非质心损失一致。对于非质心损失，Caragiannis 等人 [2024] 没有给出任何下界；也就是说，关于（最大距离）非质心损失的精确核心聚类是否存在，仍是一个开放问题。然而，Chen 等人 [2019] 证明了质心损失下界为 2-核心，我们可以据此引入该下界。有些出人意料的是，尽管我们做出了合理的努力，却无法推出优于 2 的下界。

4.2 加权单度量损失

本节重点改进加权单度量损失这一受限情形下的核心近似保证。回顾一下，该情形由单一度量d 和参数 $\lambda \in [0,1]$确定，使得对于任意$i \in \mathcal{N}$及簇(C,x)，都有 $\ell_i(C,x) = \lambda \cdot \max_{j\in C} d(i,j) + (1-\lambda)\cdot d(i,x)$。我们设计了两种能够改进这一保证的算法。

第一个算法采用非质心贪心捕获（算法 2），该算法围绕代理生长球，并对其进行调整以适用于半中心世界：将每个簇分配给距离捕获该簇的球所属代理最近的聚类中心。由此得到的高效算法见算法 5，其保证如下。

引理 7。 对于参数为 $\lambda \in (0,1]$, 的加权单度量损失，*算法 5* 可在$\frac{2}{\lambda}$-核心中于多项式时间内找到一个聚类。

证明。 为导出矛盾，假设该结论不成立，并且存在某个实例，使得贪心聚类解 $\mathcal{X} = \{(C_1,x_1),\ldots,(C_k,x_k)\}$ 不属于 $\frac{2}{\lambda}$-核心。这意味着存在某个偏离簇(S,y)，使得对于所有 $i \in S$， $\frac{2}{\lambda}\cdot\ell_i(S,y) < \ell_i(\mathcal{X})$。

令 $i = \operatorname{argmin}_{i'\in S}\mathcal{X}(i')$ 为由算法 5捕获的S 中的第一个代理。令 $\mathcal{X}(i) = t$。令 $j \in C_t$ 为“捕获”了簇 C_t 的代理（也就是说，在算法捕获C_t 时，

算法 5：非质心贪心捕获与贪心质心选择 输入：代理人集合 $\mathcal{N}$、可行中心集合 $\mathcal{M}$、度量 d、簇数 k 输出：聚类 $\mathcal{X} = \{(C_1, x_1)、\ldots、(C_k, x_k)\}$1 初始化： $\mathcal{N}' \leftarrow \mathcal{N}$, $\mathcal{X} \leftarrow \emptyset$,$\delta \leftarrow 0$; // 未捕获代理仍然存在 2 当 $\mathcal{N}' \neq \emptyset$ 时执行 3 平滑地增加 δ; // 找到非质心球 4 如果 $\exists i \in \mathcal{N}'$, $C \subseteq \mathcal{N}'$ 使得 $|C| \geqslant \min(|\mathcal{N}'|, n/k)$ 且对所有 $i' \in C$ 都有 $d(i, i') \leqslant \delta$，则 5 $x \leftarrow \arg\min_{x \in \mathcal{M}} d(i, x)$; // 距球中心最近的质心6 $\mathcal{X} \leftarrow \mathcal{X} \cup \{(C, x)\}$;7 $\mathcal{N}' \leftarrow \mathcal{N}' \setminus C$; 8 结束 9 结束 10 返回 $\mathcal{X}$

$d(i',j) \leqslant \delta$ 对所有 $i' \in C_t$ 均成立），并令 $r_t = \max^{i \in C_t} d(i,j)$ 为 δ 在 C_t 被捕获时的取值。

根据三角不等式，对于任意 $i' \in C_t$，我们有 $d(i, i') \leqslant d(i, j) + d(j, i') \leqslant 2r_t$。由此，我们有 $\ell_i(\mathcal{X}) \leqslant (1-\lambda)d(i, x_t) + \lambda 2 r_t$。

注意，必然存在某个 $i' \in S$ 使得 $d(i,i') \geqslant r_t$。若非如此，则 算法 5 本会捕获 S，且早于捕获 C_t，且 i 是 S 的“捕获”代理。因此，我们还有$\ell_i(S,y) \geqslant (1-\lambda)d(i, y) + \lambda r_t$。

由于算法 5 以贪心方式为每个簇选择中心，必然有 $d(j, y) \geqslant d(j, x_t)$，否则 y 会被选为 C_t 的中心，而非 x_t。由三角不等式，我们有 $d(i, x_t) \leqslant d(i, j) + d(j, x_t) \leqslant r + d(j, y) \leqslant r + d(i, j) + d(i, y) \leqslant 2r + d(i, y)$。由此可得 $d(i, x_t) \leqslant 2r + d(i, y)$

上述不等式使我们能够界定 $\ell_i(\mathcal{X}) \leqslant (1-\lambda)d(i, x_t) + \lambda 2r \leqslant (1-\lambda)(2r + d(i, y)) + \lambda 2r = 2r + (1-\lambda)d(i, y)$。这意味着，i 通过偏离，最多可以将其损失改善以下因子：

$$\frac{2r_t + (1-\lambda)d(i,y)}{(1-\lambda)d(i,y) + \lambda r_t} = \frac{(2-\lambda)r_t}{(1-\lambda)d(i,y) + \lambda r_t} + 1 \leqslant \frac{(2-\lambda)r_t}{\lambda r_t} + 1 = \frac{2}{\lambda}$$

这就得到了所需的矛盾。 □

第二种算法的正式描述如下算法 6，它与算法 3类似：在阶段 1 中，将代理划分为初始聚类；在阶段 2 中，允许代理切换到另一个能够降低其损失、且不会显著增加该簇中其他代理损失的簇。不过，两者有两个关键区别。首先，算法 6 使用质心贪心捕获子程序，而不是算法 3 在阶段 1 中迭代寻找簇时所用的（近似）最紧密簇子程序。其次，由于只有单一度量，切换条件要简单得多，并且对于 λ的某些取值，可以得到更好的界。经过仔细分析，可得以下保证。

引理 8. 对于参数为 $\lambda \in [0, 1)$ 的加权单度量损失，算法 6 可在多项式时间内找到一个聚类，使其位于 $\frac{\sqrt{2\lambda - 11\lambda^2 + 13} + 3 - \lambda}{2 - 2\lambda}$-核心中。

证明。 运行算法 6时，我们将增长因子设为$c = q\lambda$，其中 $q = \frac{\sqrt{2\lambda - 11\lambda^2 + 13} + 5\lambda - 1}{6}$。

设 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$ 为算法 6 返回的聚类。进一步，设 $\widehat{\mathcal{X}} = \{(\widehat{C}_1, x_1), \ldots, (\widehat{C}_k, x_k)\}$为算法 6 完成阶段 1 后找到的聚类。

算法 6： 半球增长算法

输入： 代理人集合 $\mathcal{N}$、可行中心集合 $\mathcal{M}$、距离度量 d、参数 $\lambda \in [0,1]$、簇数簇 k、增长因子 $c \geqslant 1$

输出： 聚类 $\mathcal{X} = \{(C_1, x_1)、\ldots、(C_k、x_k)\}$

初始化：$\mathcal{N}' \leftarrow \mathcal{N}$，$\widehat{\mathcal{X}} \leftarrow \emptyset$、$\delta \leftarrow 0$、$t \leftarrow 0$;

// 阶段 1：　通过质心贪心捕获构建暂定聚类

当 $\mathcal{N}' \neq \emptyset$ **时执行**　// 未捕获代理仍然存在

　平滑地增加 δ;

　if $\exists x_t \in \mathcal{M}$, $\widehat{C}_t \subseteq \mathcal{N}'$ 满足条件：$|\widehat{C}_t| \geqslant$ 最小化$(|\mathcal{N}'|, n/k)$ 且 $d(i, x_t) \leqslant \delta$ 对所有 $i \in \widehat{C}_t$ **那么**

　　$\widehat{\mathcal{X}} \leftarrow \widehat{\mathcal{X}} \cup \{(\widehat{C}_t, x_t)\}$;

　　$r_t \leftarrow \delta$;

　　$\mathcal{N}' \leftarrow \mathcal{N}' \setminus \widehat{C}_t$;

　　$t \leftarrow t + 1$

　end

结束

// 阶段 2：　有选择地允许代理切换簇

$\forall t \in [k], C_t \leftarrow \widehat{C}_t$;

对于 $i \in \mathcal{N}$ **执行**

　// 允许的簇中损失上界最小的簇

　$t' \leftarrow \arg\min_{t \in [k]: d(i,x_t) \leqslant c \cdot r_t} \{(1-\lambda) \cdot d(i, x_t) + c \cdot \lambda \cdot 2r_t\}$;

　$t^* \leftarrow \widehat{\mathcal{X}}(i)$;

　// 如果能改善损失上界，则进行切换

　如果 $(1-\lambda) \cdot d(i, x_{t'}) + c \cdot \lambda \cdot 2r_{t'} < d(i, x_{t^*}) + c \cdot \lambda \cdot r_{t^*}$ **那么**

　　$C_{t^*} \leftarrow C_{t^*} \setminus \{i\}$;

　　$C_{t'} \leftarrow C_{t'} \cup \{i\}$;

　end

结束

返回 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$

首先，我们将证明，对于任意代理$i \in \mathcal{N}$，$\ell_i(\mathcal{X}) = (1-\lambda)\, d(i, x_t) + \lambda\, d(i, x_t) + \lambda\, q\, r_t = d(i, x_t) + q\, r_t$，如果 $\widehat{\mathcal{X}}(i) = t$，那么$\ell_i(\mathcal{X})$将不超过$(1-\lambda)\, d(i, x_t) + \lambda\, d(i, x_t) + \lambda\, q\, r_t$。通过分析算法第二阶段的运行方式即可看出这一点。

在算法 6 的阶段 2 中，每个代理 $i \in \mathcal{N}$ 都可以选择从算法最初为其分配的簇切换到一个新簇。然而，他们只能切换到簇 $(C_{t'}, x_{t'})$，前提是 $d(i, x_{t'}) \leqslant c r_{t'}$。在该算法中，c 被称为“增长因子”，因为第二阶段可以理解为：将每个簇球的半径增大 c 倍，然后每个代理选择切换到一个其位于半径范围内的新簇。还需注意的是，每个代理会根据切换到 $(C_{t'}, x_{t'})$ 后可能遭受的最坏情况损失，判断某个新簇 $(C_{t'}, x_{t'})$ 是否会比其当前簇带来更小的损失。这一点反映在算法 6 的算法 6 中的 $c \cdot \lambda \cdot 2r_t$ 项上。由于每个代理损失函数中的非质心部分会根据其他代理在算法第二阶段选择切换到的簇而发生变化，因此，只有当无论其他代理如何切换，其在该新簇下的损失都仍能保证小于当前损失时，每个代理才会切换到新簇。

对于任意满足 $\widehat{\mathcal{X}}(i) = t$ 的 $i \in \mathcal{N}$，如果 i 在算法的第二阶段没有切换到新的簇，那么我们可以证明它的最终损失将小于 $d(i, x_t) + q \cdot r_t$。为此，假设 i 在阶段 2 没有切换簇，因此 $\mathcal{X}(i) = t$。由于我们知道 i 没有切换，因此它到其最终簇质心的距离必定为 $d(i, x_t) \leqslant r_t$。此外，根据第二阶段允许代理切换簇的方式，我们知道对于所有满足条件的 $j \in C_t$，必有 $d(j, x_t) \leqslant c \cdot r_t$。根据三角不等式，这意味着对于 $j = \max_{j \in C_t} d(j, i)$，也就是 C_t 中距离 i 最远的代理，必有 $d(i, j) \leqslant d(i, x_t) + d(j, x_t) \leqslant d(i, x_t) + c \cdot r_t$。

综合以上结果并代入$c = q\lambda$，我们得到：

$$\ell_i(\mathcal{X}) = (1-\lambda) \cdot d(i, x_t) + \lambda \cdot d(i, j) \leqslant (1-\lambda) \cdot d(i, x_t) + \lambda \cdot (d(i, x_t) + c \cdot r_t) = d(i, x_t) + q \cdot r_t$$

类似地，如果i在阶段 2确实选择切换簇，我们也可以给出其损失的上界。现在假设i 确实进行了切换，因此算法的第二阶段结束后，$\mathcal{X}(i) = t'$。

按照与上文类似的推理，我们可以给出i的损失上界：

$$\ell_i(\mathcal{X}) = (1-\lambda) d(i, x_{t'}) + \lambda \ell_i^m(\mathcal{X}) \leqslant (1-\lambda) d(i, x_{t'}) + \lambda 2 c r_{t'} = (1-\lambda) d(i, x_{t'}) + 2 q r_{t'}$$

这里，第二个不等关系来自这样一个事实：$C_{t'}$ 中的每个代理到$x_{t'}$的距离至多为$c r_{t'}$。结合与上文类似的三角不等式论证，我们可以得到 $\ell_i^m(C_{t'}, x_{t'}) \leqslant 2 c r_{t'}$。请注意，这比我们在<style id='i'>i不切换时为其非质心损失所得到的上界更差。这是因为i 不切换时，我们知道$d(i,x_t) \leqslant r_t$；而i 切换时，我们只知道较弱的上界 $d(i, x_{t'}) \leqslant c \cdot r_t$。

从算法 6的第 13 行可以看出，代理i 只有在新簇的损失上界小于原第一阶段簇的损失上界时，才会从其第一阶段簇切换到新簇。因此，无论i 在第二阶段做什么，我们都可以保证其损失不会超过$d(i,x_t) + q \cdot r_t$。

确定了每个代理的损失上界后，我们可以证明所需的核保证。反设结论不成立，且算法 6 生成了聚类 $\mathcal{X}$，使得存在某个偏离簇$(S、y)$，其中$S \subseteq \mathcal{N}, |S| \geqslant \lceil \frac{n}{k} \rceil$，且$y \in \mathcal{M}$，并且对于所有$i \in S$，都有 $\frac{\sqrt{2\lambda - 11\lambda^2 + 13 + 3 - \lambda}}{2 - 2\lambda} \ell_i(S、y) < \ell_i(\mathcal{X})$。

令 $i = \min_{i \in S} \widehat{\mathcal{X}}(i)$ 表示 S 中在算法阶段 1 首先被捕获的代理，并令$\widehat{\mathcal{X}}(i) = t$。我们知道 $d(i, x_t) \leqslant r_t$。令 $j = \arg\max_{j \in S} d(j, y)$ 表示 S 中距离 y 最远的代理。我们知道 $d(j, y) \geqslant r_t$ 必定成立，否则，算法第一阶段会在捕获 $(\widehat{C}_t, x_t)$ 之前捕获 (S, y)。

首先，我们考虑 $d(j,x_t) \leqslant c \cdot r_t$ 的情况。这意味着在算法的阶段 2 中，j将有机会切换到第 t 个簇。对于以 x_t 为中心的簇，j 可能感受到的最大损失为 $(1-\lambda) d(j,x_t) + 2 q r_t$；由于 j 将切换到上界最低的簇

损失，这意味着 j 偏离至 (S,y) 最多可以将其损失改善一个 $\frac{(1-\lambda)d(j,x_t)+2qr_t}{(1-\lambda)d(j,y)}$ 的因子。注意，这可以上界为

$$\begin{aligned}\frac{(1-\lambda)d(j,x_t)+2qr_t}{(1-\lambda)d(j,y)} &\leqslant \frac{(1-\lambda)(d(j,y)+d(i,y)+d(i,x_t))+2qr_t}{(1-\lambda)d(j,y)}\\ &\leqslant \frac{(1-\lambda)d(j,y)+(1-\lambda)(d(i,y)+r_t)+2qr_t}{(1-\lambda)d(j,y)}\\ &\leqslant \frac{(1-\lambda)r_t+(1-\lambda)(d(i,y)+r_t)+2qr_t}{(1-\lambda)r_t}\\ &= \frac{(2(1-\lambda)+2q)r_t+(1-\lambda)d(i,y)}{(1-\lambda)r_t}\\ &\leqslant \frac{2(1-\lambda)+2q}{(1-\lambda)}+\frac{d(i,y)}{r_t}. \end{aligned} \tag{2}$$

对于i，必须有 $\ell_i^m(S,y) \geqslant (1-\lambda)d(i,y)$，且 $\ell_i^m(\mathcal{X})$ 不能超过 i 在 $(\widehat{C}_t, x_t)$ 下的上界损失；这意味着i偏离所能改善的损失不超过一个因子

$$\frac{d(i,x_t)+q\cdot r_t}{(1-\lambda)d(i,y)} \leqslant \frac{(1+q)r_t}{(1-\lambda)d(i,y)}. \tag{3}$$

结合公式 (2)和(3)，我们得到以下界：

$$\begin{aligned}\min\left\{\frac{1+q}{1-\lambda}\cdot\frac{r_t}{d(i,y)}, \frac{2(1-\lambda)+2q}{(1-\lambda)}+\frac{d(i,y)}{r_t}\right\} &\leqslant \max_{z\geqslant 0}\min\left\{\frac{1+q}{1-\lambda}z, \frac{2(1-\lambda)+2q}{(1-\lambda)}+\frac{1}{z}\right\}\\ &= \frac{\sqrt{\lambda^2-3\lambda(q+1)+q^2+3q+2}-\lambda+q+1}{1-\lambda},\end{aligned}$$

其中，最后一步可通过令z 中的两个表达式相等，并求解所得的z二次方程得出。代入所选的q 值并化简后，我们得到以下界：

$$\frac{\sqrt{2\lambda-11\lambda^2+13}+3-\lambda}{2-2\lambda}.$$

这就产生了矛盾。这表明必然有$d(j,x_t) > c\cdot r_t$，也就是说，j 在算法 6第二阶段期间不能切换到第t个簇。

结合这一点和三角不等式，可得 $q_\lambda r_t < d(j,x_t) \leqslant d(j,i)+d(i,x_t) \leqslant d(j,i)+r_t$。重新整理可得 $d(i,j) > (q_\lambda - 1)r_t = (\frac{q-\lambda}{\lambda})r_t$。

由此，我们可以说 $\ell_i(S,y) \geqslant \lambda d(i,j) > \lambda\frac{q-\lambda}{\lambda}r_t = (q-\lambda)r_t$。

结合 $\ell_i(\mathcal{X}) \leqslant d(i,x_t)+q\cdot r_t \leqslant (q+1)r_t$这一事实，我们可以得出，在这种情况下，$i$ 偏离到(S,y) 无法使其损失改善超过$q+1\,q-\lambda$这一因子。

代入我们选定的q 值并再次化简，得到：

$$\frac{\sqrt{2\lambda-11\lambda^2+13}+3-\lambda}{2-2\lambda}$$

这就产生了矛盾，并完成了证明。 □

结合 引理 7 和 8中的界，以及存在性 3-核心（定理 2）和多项式时间 $(3+2\sqrt{3})$-核心（定理 3）所提供的保证，我们可以从双度量损失中引入以下存在性界和可高效实现的界。

定理 4。 对于参数为 $\lambda \in [0,1]$ 的加权单度量损失，$\min\{\frac{2}{\lambda}, 3\}$-核心中的聚类始终存在，而$\min\{\frac{2}{\lambda}, \frac{\sqrt{2\lambda-11\lambda^2+13+3-\lambda}}{2-2\lambda}\}$-核心中的聚类可以在多项式时间内找到。

下界： 类似地，我们也推导出了加权单度量损失下核心近似的两个下界。第一个下界在 $(\sqrt{5}+1)/2$（当 $\lambda = 0$（质心）时）与 1（当 $\lambda = 1$（非质心）时）之间插值。由于 Chen 等人 [2019]证明了 $\lambda = 0$（质心）的下界为 2，这优于$(\sqrt{5}+1)/2$，因此将他们的反例改编到半质心聚类后，当λ 接近 0 时，可以得到更好的下界。

引理 9。 对于参数为 $\lambda \in [0,1]$ 的加权单度量损失，没有任何聚类算法能够实现优于 $\frac{\sqrt{\lambda^2-2\lambda+5-\lambda+1}}{2}$ 的核心近似。

证明。 我们将分析 图 1中的实例，其中 $p = \frac{\sqrt{\lambda^2-2\lambda+5+\lambda-1}}{2\lambda}$。我们将分析范围限制在该实例中包含 $\{a、b、c\}$的部分；由于 $k = 3$，不失一般性，我们可以假设其中放置了一个聚类中心。

如果最初选择的簇是({a, b, c}, b)，那么 a 的损失为$(1-\lambda)$d(a, b) $+\lambda$d(a, c) $=(1-\lambda) + \lambda$p，而 b 的损失为$(1-\lambda)0 + \lambda$p $= \lambda$p。如果 a 和 b 偏离到簇({a, b}, b)，那么他们的损失将分别改善因子$(1-\lambda) + \lambda$ p 和 p。这种偏离将导致核心性以因子$(1-\lambda) + \lambda$p $= \frac{\sqrt{\lambda^2+2\lambda+5-\lambda+1}}{2}$遭到违反。同样的论证可以表明，({ a, b, c}, a)也不能作为一个簇；显然，({a, b, c}, c)也不行，因为 a 和 b 的损失只会比之前更糟。

如果最初选择的簇是({a, b}, b)，那么请注意，a 的损失为$(1-\lambda)$d(a, b) $+\lambda$d(a, b) $=$1。相比之下，在聚类({a, c}, a)中，a 的损失为$(1-\lambda)0 + \lambda$d(a, c) $= \lambda$p。这意味着，a 通过偏离可以将损失降低因子$\frac{1}{\lambda}$ p $= \frac{2}{\sqrt{\lambda^2-2\lambda+5+\lambda-1}} = \frac{\sqrt{\lambda^2+2\lambda+5-\lambda+1}}{2}$。由于我们假设只从 {a, b, c} 中选择一个点作为中心，因此这种偏离也必定会使 c 的损失改善无限大的因子。请注意，如果初始簇中选作质心的是 a 而不是 b，同样的分析也成立。显然，中心 c 也不适用于这个簇，因为它只会让两个代理都得不到任何改善。

接下来，注意到簇 $(\{a,c\}, a)$ 不能被选中。a 的损失将为$(1-\lambda)0 + \lambda p = \lambda p$。但在聚类$(\{a,b\}, a)$ 下，a的损失将为$(1-\lambda)0 + \lambda = \lambda$，因此偏离可使$a$的损失改善$p \geqslant \frac{\sqrt{\lambda^2+2\lambda+5-\lambda+1}}{2}$倍。$b$ 的损失也会改善无限倍。显然，尝试选择以任何其他中心构成的这个簇都无济于事，因为这只会使a的初始损失更高。此论证也可用于证明，成员集合为 $\{b,c\}$ 的簇不可行。

这排除了由这3个代理组成的任何大小大于1的簇的可能性。最后，这些代理中不能有超过2个参与单元素簇，因为由于$k = 3$，另一侧的所有代理都必须属于同一个簇，因此我们可以用与本证明第二段类似的论证，在那里找到一个偏离集合。 □

引理 10. 对于参数为 $\lambda \in [0,1]$ 的加权单度量损失，没有任何聚类算法能够实现优于 $\frac{2(1-\lambda)}{2\lambda+1}$ 的核心近似。

证明。取自 Chen 等人 [2019] 的断言 1 中的实例。 $\mathcal{N} = \{a_1, \ldots, a_6\}, \mathcal{M} = \{y_1, \ldots, y_6\}, k = 3$。代理之间的距离由 $d(a_1, a_2) = d(a_1, a_3) = d(a_2, a_3) = 3$ 和 $d(a_4, a_5) = d(a_4, a_6) = d(a_5, a_6) = 3$ 给出，其他所有点对之间的距离均为 ∞。代理与中心之间的距离如下表所示（也可参见 Chen 等人 [2019] 的断言 1）。

Chen 等人 [2019]证明在这个实例中，不可能实现优于 f
2 的质心核心近似。

修复某个聚类 $\mathcal{X} = \{(C_1, x_1), (C_2, x_2), (C_3, x_3)\}$。根据 Chen 等人 [2019], 的研究，在质心世界中，必定存在一对代理希望偏离，以实现至少 2 的核心改进。更具体地说，由于有 3 个簇，且该实例由两个彼此相距 的代理/中心“群组”组成，其中

	y_1	y_2	y_3	y_4	y_5	y_6
a_1	4	1	2	∞	∞	∞
a_2	2	4	1	∞	∞	∞
a_3	1	2	4	∞	∞	∞
a_4	∞	∞	∞	4	1	2
a_5	∞	∞	∞	2	4	1
a_6	∞	∞	∞	1	2	4

一个群组中被选为 $\mathcal{X}$ 中心的中心数量最多为 1 个。

不失一般性，假设群组 $\{a_1,a_2,a_3\}$ 中只有一个中心被 $\mathcal{X}$从该群组中选中，并假设 y_1 就是该中心（由于实例的对称性，我们可以选择任一群组及该群组中的任一中心，论证都完全相同）。在此情形下，来自 $\mathcal{X}$ 的另外两个中心分别与 a_1 和 a_2 相距 ∞。a_1 和 a_2 要么属于包含 y_1，的簇，要么属于一个簇，其中心距它们比 y_1 更远。因此，我们必须有$\ell_{a_1}(\mathcal{X}) \geqslant (1-\lambda)d(a_1,y_1) = 4(1-\lambda)$，且 $\ell_{a_2}(\mathcal{X}) \geqslant (1-\lambda)d(a_2,y_1) = 2(1-\lambda)$。

现在考虑偏离簇 $(S = \{a_1,a_2\},y_3)$。a_1 和 a_2 在该簇下的损失如下：

$$\ell_{a_1}(S,y_3) = \lambda d(a_1,a_2) + (1-\lambda)d(a_1,y_3) = 3\lambda + 2(1-\lambda)$$
$$\ell_{a_2}(S,y_3) = \lambda d(a_1,a_2) + (1-\lambda)d(a_2,y_3) = 3\lambda + 1(1-\lambda)$$

这些结果，加上我们关于 $\mathcal{X}$ 下代理损失的下界，使我们得以得出结论：通过偏离，a_1 至少可以获得 $4(1-\lambda)\ 3\lambda+2(1-\lambda)$ 的改进，而 a_2 则至少可以获得 $2(1-\lambda)\ \ 3\lambda+1(1-\lambda)$的改进。

容易验证，对于所有 $\lambda \in [0,1]$，都有 $4(1-\lambda)\ 3\lambda+2(1-\lambda) \geqslant {}^{2}(1-\lambda)\ 3\lambda+1(1-\lambda)$，这意味着将导致 $2(1-\lambda)\ 3\lambda+1(1-\lambda) = 2(1-\lambda)\ 2\lambda+1$ 的核心违反，从而完成证明。 □

根据上述两个下界，我们可以作出如下概括性陈述。

定理 5. 对于参数为 $\lambda \in [0,1]$ 的加权单度量损失，任何聚类算法都无法达到优于 $\max\{{}^{\sqrt{\lambda^2-2\lambda+5}-\lambda+1}_{2}, {}^{2}(1-\lambda)\ 2\lambda+1\}$ 的核心近似。

证明。 这可直接由 引理 9 和引理 10得出。 □

5 完全正当代表性

接下来，我们关注较弱的完全正当代表性（FJR）判据。该判据是核心的一种放宽形式，最初由 Peters 等人 [2021] 在基于认可的委员会选举这一背景下提出。即使采用双度量损失，我们也已经能够实现对核心的（较小）常数近似（定理 2 和 3），但我们将证明，这种放宽形式具有若干优势。

Caragiannis 等人 [2024] 研究了非质心聚类中的 FJR，并证明，迭代地寻找一个 α-近似最紧密簇会产生一个 α-FJR 聚类，即使在 任意损失函数下也是如此。相同的结果及几乎完全相同的证明，也适用于我们更一般的半质心聚类设置。为完整起见，我们将算法正式表述为 算法 7 并给出完整证明。这说明，在任意损失函数下都存在一个 1-FJR 聚类；而在双度量损失下，结合我们用于计算 4-MCC 簇的多项式时间算法（引理 6），可以在多项式时间内计算一个 4-FJR 聚类。

算法 7： 迭代式 α-MCC 聚类

输入： 代理人集合 $\mathcal{N}$、可行中心集合 $\mathcal{M}$、代理的损失函数 $(\ell_i)_{i\in\mathcal{N}}$、中心数量 k、近似因子 $\alpha \geqslant 1$；

输出： 聚类 $\mathcal{X} = \{(C_1, x_1)、\ldots\ldots、(C_k,\ x_k)\}$

初始化：$\mathcal{N}' \leftarrow \mathcal{N}$、$\mathcal{X} \leftarrow \emptyset$；

// 未捕获代理仍然存在

当 $\mathcal{N}' \neq \emptyset$ **时执行**

  $(C, x) \leftarrow$ 一个来自 $\mathcal{N}', \mathcal{M}$ 的 α-MCC 簇（阈值为 n/k）；　// 查找一个 α-MCC 簇

  $\mathcal{X} \leftarrow \mathcal{X} \cup \{(C、x)\}$；　// 将其添加到聚类中

  $\mathcal{N}' \leftarrow \mathcal{N}' \setminus C$；　// 将其中的代理标记为已捕获

结束

返回 $\mathcal{X}$

定理 6。 对于任意损失函数和 $\alpha \geqslant 1$，算法 7迭代地寻找一个 α-近似最紧密簇，并返回一个 α-*FJR* 聚类。因此，在任意损失函数下都存在一个1-*FJR* 聚类；此外，根据 引理 6，在双度量损失下，可以在多项式时间内计算一个4-*FJR* 聚类。

证明。 反设算法 7 并非总能产生 α-FJR 聚类。那么，存在一个实例，其中算法 7 返回的聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$ 存在一个偏离簇(S,y)，使得对所有$i \in S$，

$$\alpha \cdot \ell_i(S, y) < \min_{j\in S} \ell_j(\mathcal{X}). \tag{4}$$

令 $i^* \in \operatorname{argmin}_{i\in S} \mathcal{X}(i)$ 为S 中第一个被算法“捕获”（即从 $\mathcal{N}'$中移除）的代理（若有并列，则任意打破）。令 $t = \mathcal{X}(i^*)$。那么，根据 公式 (4)，对于所有 $i \in S$，

$$\alpha \cdot \ell_i(S, y) < \min_{j\in S} \ell_j(\mathcal{X}) \leqslant \ell_{i^*}(\mathcal{X}) \leqslant \max_{i\in C_t} \ell_i(\mathcal{X}) = \max_{i\in C_t} \ell_i(C_t, x_t).$$

因此，$\alpha \cdot \max_{i\in S} \ell_i(S, y) < \max_{i\in C_t} \ell_i(C_t, x_t)$。由于在将 (C_t, x_t) 加入 $\mathcal{X}$ 之前，S 中的所有代理都尚未被捕获，这与当时 (C_t, x_t) 是一个 α-MCC 簇相矛盾。由此证明算法 7 总能产生 α-FJR 聚类。

由于根据定义 MCC 簇必然存在，代入 $\alpha = 1$ 即可得到 FJR 聚类的存在性。对于双度量损失，由于可以通过 算法 4 (引理 6) 高效地计算 4-MCC 簇，因此我们得到可以在多项式时间内计算 4-FJR 聚类。 □

在 第 3 节中，我们注意到，对于一个聚类，同时相对于质心损失和非质心损失属于 α-核心（这是显然的）是其相对于加权单一度量损失属于 α-核心（对所有 $\lambda \in [0, 1]$）的必要条件。

对于 FJR，情况显然仍然如此；遗憾的是，我们可以证明，当允许两个损失函数使用不同的度量时，同时相对于由两个不同度量诱导的质心损失和非质心损失实现 FJR 近似仍然是不可行的。

定理 7。 对于任意 $\alpha \geqslant 1$ 和 $\beta \geqslant 1$，存在一个实例，使得当允许两个损失函数使用不同的度量时，不存在一个聚类能够同时相对于质心损失满足 α-*FJR*，并相对于非质心损失满足 β-*FJR*。

证明。 考虑一个实例，其中包含一个由 6 个代理组成的集合 $\mathcal{N} = \{a、b、c、d、e、f\}$，一个包含两个可行簇中心的集合$\mathcal{M} = \{x_1, x_2\}$，以及 $k = 3$。

非质心度量： 令 $d^m(a、b) = d^m(c、d) = d^m(e、f) = 0$，并令其他每一对代理之间的非质心距离均为 ∞。[3]

质心度量：令 $d^c(a, x_1) = d^c(c, x_1) = d^c(e, x_1) = 0$，令 $d^c(b, x_2) = d^c(d, x_2) = d^c(f, x_2) = 0$，并令其他每个智能体-中心对之间的质心距离为 ∞。

[3]这里，为便于说明，我们使用 ∞，但也可以将其替换为一个足够大的数。

有限非质心 *FJR* 近似：首先，请注意，唯一能针对非质心损失产生有限 FJR 近似的聚类，是将代理人集合划分为 $\mathcal{X}=\{\{a、b\}、\{c、d\}、\{e、f\}\}$ 的聚类。

为此，考虑任意一个$C \in \mathcal{X}$。如果C 被“拆分”（即它本身不是一个簇），那么C 中至少有一个成员必须构成单元素簇；唯一的另一种可能是C 是真子集，包含于某个簇中。在这种情况下，C 可以偏离，使其两个成员的损失都从无限降为零，从而得到无限的 FJR 近似。

因此，$\mathcal{X}$ 中的每个未拆分的配对都构成一个大小为二的簇，而 $\mathcal{X}$ 中的每个被拆分的配对至少有一个成员构成单元素簇。由于 $\mathcal{X}$中有三个配对，这已经占用了$k=3$个簇，导致 $\mathcal{X}$ 中被拆分配对的其他成员未被纳入划分，这是不可能的。因此，不可能存在被拆分的配对，这说明划分必定为 $\mathcal{X}=\{\{a、b\}、\{c、d\}、\{e、f\}\}$。

有限质心 *FJR* 近似：现在，让我们尝试为 $\mathcal{X}$ 中的每个簇分配一个中心，使得相对于质心损失的近似值有限。由于只有两个可行中心x_1和x_2，，它们中至少有一个必须至多分配给 $\mathcal{X}$中的一个簇。不失一般性，假设这个中心是x_1。有三个代理a、c和e，在质心度量下，它们与x_1 的距离为零。然而，它们分别属于不同的簇。因此，它们中至少有两个必须被分配到x_2，，并可以与x_1 一起偏离，使其质心损失都从非零降为零，从而得到相对于质心损失的无限 FJR 近似。如果我们假设的是x_2，，则可以对代理b、d和f作出同样的论证。

这表明，不存在一种聚类能同时针对质心损失和非质心损失实现有限的 FJR 近似，从而完成证明。□

当两种损失均由共同度量 d诱导时，一种出人意料地简单的算法能够针对质心损失和非质心损失同时实现 FJR 的常数近似。这与核的情况形成对比：对于核，即使只有一种度量，也无法实现同时有限近似（定理 1）。

该算法是 算法 5：它先使用度量 d 运行非质心贪心捕获，然后执行贪心质心选择步骤。我们在第 4.2 节中介绍了该算法，并证明它对所有 $2/\lambda$均能产生$\lambda \in [0,1]$-核心。需要注意的是，针对非质心损失，该算法实现了 2-核心（因此也实现了 2-FJR）（$\lambda=1$）。然而，随着 λ 趋近于 0（即质心损失），它对核的近似比会变得无界；鉴于定理 1，这是不可避免的。不过，我们证明了，对于 FJR 这一较弱判据，该算法的近似比仍不超过 5。

引理 11。 *算法 5在给定度量 d 的情况下，生成的聚类是5-FJR，且该性质相对于由 d 诱导的质心损失成立。*

证明。 反设结论不成立。那么，存在一个实例，其中算法 5返回了聚类 $\mathcal{X}=\{(C_1,x_1),\ldots,(C_k,x_k)\}$，且存在一个偏离簇$(S,y)$ 使得

$$\forall q \in S : 5 \cdot d(q,y) < \min_{p \in S} \ell_p^c(\mathcal{X}). \tag{5}$$

令 i $\in$ S 为 S 中最先被算法 5 捕获的代理。为简便起见，将包含代理 i 的簇记为 (C, x) $\triangleq$ (C$\mathcal{x}$(i), x$\mathcal{x}$(i))。由于算法 5 通过非质心贪心捕获来创建代理人集合的划分，C 必定是由一个以某个代理 j $\in$ C 为中心的球捕获的。注意，算法 5 为 C 选定的中心 x 必须是距离代理 j 最近的可行中心。令 r $=\max_{i' \in C} d(j,i')$ 为以 j 为中心的球捕获 C 时的半径。

首先，我们证明 $d(i,x) \leqslant 2r + d(i,y)$。为此，请注意

$$d(i,x) \leqslant d(i,j) + d(j,x) \leqslant d(i,j) + d(j,y) \leqslant d(i,j) + d(i,j) + d(i,y) \leqslant 2r + d(i,y),$$

其中，第一个和第三个不等式使用了三角不等式，第二个不等式利用了以下事实：x 是代理j 最近的中心（因此，$d(j,x) \leqslant d(j,y)$），最后一个不等式则由r的定义得出。

现在，根据方程（5），我们有

$$\forall q \in S : 5 \cdot d(q,y) < \min_{p \in S} \ell_p^c(\mathcal{X}) \leqslant \ell_i^c(\mathcal{X}) = d(i,x) \leqslant 2r + d(i,y).$$

特别地，代入 q = i，可得 5 · d(i, y) < 2r + d(i, y)，从而得到 d(i, y) < r/2。然后，对于其他每个代理 q ∈ S，都有 5 · d(q, y) < 2r + d(i, y) < 5r/2，，由此得到 d(q, y) < r/2。

根据三角不等式，对于所有 p、q ∈ S，都有 d(p, q) ⩽ d(p, y) + d(q, y) < r/2 + r/2 = r。但这产生了矛盾，因为这意味着在算法 5 的执行过程中，以 S 中任意代理为中心的球会先于以代理 j 为中心的球捕获 S，然后后者才会捕获 C。

□

结合引理 11，我们可以陈述如下结论：

定理 8。 *算法 5给定度量 d，可以在多项式时间内找到一个聚类，使其相对于质心损失满足 5-FJR，并且相对于非质心损失满足2-FJR，二者均使用 d 定义。*

证明。 相对于质心损失的 5-FJR ℓ^c 已在引理 11中得到证明。

Caragiannis 等人 [2024] 证明了，非质心贪心捕获相对于非质心损失 ℓ^m可以实现 2-FJR。容易看出，由于算法 5 执行非质心贪心捕获，并使用其生成的代理人集合 $\mathcal{N}$的划分$(C_1, \ldots, C_k)$，因此它保留了相对于非质心损失 ℓ^m的这一 2-FJR 保证。

□

6 实验

本节旨在通过实证比较本文提出的比例公平算法与经典聚类方法，比较内容涵盖比例公平性指标以及经典聚类方法所优化的传统指标。

由于经典方法仅将单一距离度量 d 作为输入，因此我们的实证分析仅限于加权单度量损失，该损失取决于单一度量 d 以及参数 $\lambda \in [0,1]$。

算法。 我们评估四种算法。

- *GC*： 算法 5，先执行非质心贪心捕获，再进行贪心质心选择，并达到 $\frac{2}{\lambda}$-核心（引理 7）。我们将其记为“GC”。
- *SemiBall*： 算法6，其模仿质心贪心捕获，但限制簇切换，并达到 $\frac{\sqrt{2\lambda-1}1\lambda^2+13+3-\lambda}{2-2\lambda}$-核心（引理 8）。我们将其记为~~“SemiBall”~~。
- *k*-均值++：这是一种经典算法，可近似优化*k*-均值目标函数，$\sum_{i \in \mathcal{N}} d(i, \mathcal{X}(i))^2$。
- *k-medoids*： 这是一种经典算法，可近似优化*k*-medoids 目标函数，$\sum_{i \in \mathcal{N}} d(i, \mathcal{X}(i))$。

k-均值++ 和*k*-medoids 的实现基于 Python 的 Scikit-learn 软件包。[4]

值得注意的是，四种算法中的三种——GC、*k*-均值++和*k*-medoids——仅将距离度量d 作为输入，而不使用参数 λ。因此，这些方法返回的聚类与 λ无关，不过它们对比例公平性指标的近似结果会受到 λ的影响。

数据集。 我们使用了 UCI机器学习知识库中的三个数据集 [Kelly等人， 2025]：鸢尾花，皮马印第安人糖尿病和*Adult*。这三个数据集也被Caragiannis 等人 [2024]用于非质心聚类实验。

- 鸢尾花数据集包含 150 个数据点，记录了三种鸢尾花的萼片和花瓣尺寸。

[4]Scikit-learn： https://scikit-learn.org

- （皮马印第安人）糖尿病数据集包含胰岛素剂量、葡萄糖水平和妊娠次数等健康相关指标。

- *Adult* 数据集包含 48,842 个样本，具有类别特征（如种族和教育程度）以及数值特征（如年龄和每周工作时长）。该数据集通常用于分类任务，即预测个人收入是否超过指定阈值；但在此处，它用于对个人进行聚类。我们对所有类别特征应用独热编码。[5]

指标。 我们使用五项指标评估上述四种算法。

前两项指标是核心违规和 FJR 违规（即使聚类处于 α-核心或 α-FJR 状态所需的最小 α）；这两项都是比例公平性指标。我们通过求解整数线性规划，精确计算核心违规和 FJR 违规。给定一个聚类后，该规划会找到一个偏离簇 (C,x)，记录相应指标可能出现的最大违规程度。

接下来的三项指标是 k-均值目标（如上所述）、k-中心点目标（如上所述），以及平均簇内距离，$\sum_{t\in[k]} \frac{1}{|C_t|} \sum_{i,j\in C_t} d(i,j)$ [Ahmadi 等人， 2022], 后者是另一种常用目标；这三项可以视为效率指标。给定聚类后，这些指标很容易计算。

实验设置。 遵循 Chen 等人 [2019],Caragiannis 等人 [2024], 的做法，我们假设 $\mathcal{N} = \mathcal{M}$，并使用欧氏 L_2 距离度量。我们改变两个参数：簇数 $k \in \{5,6,\ldots, 25\}$ 以及加权损失参数 $\lambda \in \{0.1,0.2,\ldots,0.9\}$具体而言，在改变 $\lambda \in \{0.1,0.2,\ldots,0.9\}$ 并固定 $k = 15$时，以及改变 $k \in \{5,6,\ldots, 25\}$ 并固定 $\lambda = 0.5$ 时，我们报告实验结果。

尽管我们的算法（以及传统的基线方法）具有良好的可扩展性，但求解整数线性规划以计算其返回的聚类的精确核心和 FJR 违规情况是计算密集型的。因此，对于两个较大的数据集，即 *Adult* 和*Diabetes*，我们在 40 次独立试验中的每次试验里随机抽取 100 个数据点。对于鸢尾花 数据集，由于k-means++使用随机初始化步骤，我们运行它 20 次。我们的图表展示了上述五项指标的平均值以及 95% 置信区间。

结果。 关于 *Adult*、*Diabetes*和 鸢尾花 数据集的结果展示于 图 2 至 4。这三个数据集的定性结论相似，因此我们以 *Adult* 数据集（图 2）为例进行说明。

首先，我们的两种算法 GC 和 SemiBall 实现了近乎精确的核心和 FJR，其表现显著优于引理 7 和8给出的最坏情况近似保证。经典算法 k-means++和k-medoids 存在明显的核心和 FJR 违规，且其公平性会随着k 增大而恶化；这可能是因为k 仍小于$n/2$，因此k 增大时，能够偏离的小联盟数量也会增加（联盟规模接近n/k）。尤其是，k-means++极不公平，特别是在k 较大且 λ 较小的情况下（即质心损失占主导时）。有趣的是，k-medoids 的公平性尚可；
Caragiannis 等人 [2024] 指出，在 $\lambda = 1$ (非质心损失)的设置下，它并不公平；但我们的结果表明，随着 λ 减小，其公平性会提高；当 $\lambda \to 0$ (质心损失)时，它的公平性几乎与 GC 和 SemiBall 一样高，并能实现近乎精确的核心和 FJR 聚类。

与k-均值++ 和k-中位数相比，GC 和 SemiBall 公平性的提升是以效率指标下降为代价的。不过，效率损失很小，尤其是在k 较大或 λ 较小时。在我们的两种算法中，SemiBall 在效率指标方面似乎始终表现更好（公平性指标则相同）。

因此，我们的主要结论是，SemiBall 和k-medoids 是半质心聚类中很有吸引力的算法选择；二者之间的选择可能取决于k 和 λ 的取值，以及委托方对公平性与效率权衡的期望。

[5]若不进行独热编码，将类别特征编码为数值（例如 1、2、3、.....）会产生误导性的距离。

7 讨论

我们的工作留下了一些亟待解决的技术问题，可从表 1中上下界之间的差距看出。未来工作还可以探讨一些推广方向，例如质心损失与非质心损失的非加性组合，以及若干令人振奋的应用。

我们基本未深入探讨的这一模型的一个尤为重要的特殊情形是 $\mathcal{M} = \mathcal{N}$。如前所述，将这一限制应用于质心聚类，正是EbadianandMicha [2025]用来对抽选问题进行建模的方法（随机选择一个能够代表总体的代理人委员会），也可以用于对许多问题进行建模：希望将一组 n 个数据点划分为 k 个簇，并返回一个点作为该簇的代表点。这可以有许多应用，例如核心集选择 [Chai 等人、 2023], 可扩展的大语言模型微调 [Yang 等人、2024], 特征选择 [Liu 等人、 2011], 以及降维 [Napoleon and Pavalakodi）。 2011]然而，我们的半质心算法无法保证其为某个簇选择的质心是该簇中的点，而在所有这些应用中，这一点可能都是理想的。探索带有这一额外约束的算法将是一个有趣的未来研究方向。

当 $\mathcal{M} \neq \mathcal{N}$ 时，例如将会议录用的论文划分为 k 场次并为每场安排一个主题，或将学生划分为 k 个团队并为每个团队分配一个课程项目时，可以使用我们针对双度量损失的算法，因为这类情况可能需要使用两种不同的度量。

其中一些应用的规模可能很大，我们的多项式时间算法或许无法满足需求，因此可能需要设计近线性时间算法。通过此类超快速算法能够实现怎样的比例公平保证，仍是一个尚未探索的领域。

参考文献

Saba Ahmadi、Pranjal Awasthi、Samir Khuller、Matth¨aus Kleindessner、Jamie Morgenstern、Pattara Sukprasert 和 Ali Vakilian。《聚类中的个体偏好稳定性》，2022年。网址：https://arxiv.org/abs/2207.03600。

Esther M. Arkin、SangWonBae、Alon Efrat、Kazuya Okamoto、JosephSBMitchell 和 ValentinPolishchuk。《几何稳定室友》。《信息处理快报》，109(4):219–224，2009年。

Haris Aziz、Barton E Lee、Sean Morota Chu 和 Jeremy Vollen。比例代表性聚类。载于第*20*届网络与互联网经济会议（*WINE*）论文集，2024年。即将出版。

Maria-Florina Balcan、Travis Dick、Ritesh Noothigattu 和 Ariel D Procaccia。无嫉妒分类。载于第*33*届神经信息处理系统年度会议（*NeurIPS*）论文集，第1238–1248页，2019年。

Suman Bera、Deeparnab Chakrabarty、Nicolas Flores 和 Maryam Negahbani。用于聚类的公平算法。载于第*32*届神经信息处理系统年度会议（*NeurIPS*）论文集，第4955–4966页，2019年。

Ioannis Caragiannis、Evi Micha 和 Nisarg Shah。非质心聚类中的比例公平。载于第*38*届神经信息处理系统年度会议（*NeurIPS*）论文集，第19139–19166页，2024年。

Chengliang Chai、Jiayi Wang、Nan Tang、Ye Yuan、Jiabin Liu、Yuhao Deng 和 GuorenWang。使用基于聚类的方法高效选择核心集。载于第*29*届国际知识发现与数据挖掘会议（*KDD*）论文集，第167–178页，2023年。

Bhaskar Ray Chaudhury、Aniket Murhekar、Zhuowen Yuan、Bo Li、Ruta Mehta 和 ArielD Procaccia。通过比例否决核心实现公平联邦学习。载于第*41*届国际机器学习会议（*ICML*）论文集，第42245–42257页，2024年。

Xingyu Chen、Brandon Fain、Liang Lyu 和 Kamesh Munagala。比例公平聚类。载于第*36*届国际机器学习会议（*ICML*）论文集，第1032–1041页，2019年。Yiling Chen、Chara Podimata、Ariel D Procaccia 和 Nisarg Shah。高维策略无关线性回归。载于第*19*届*ACM*经济与计算会议（*EC*）论文集，第9–26页，2018年。Anshuman Chhabra、Karina Masalkovait˙e 和 Prasant Mohapatra。聚类公平性综述。*IEEEAccess*，9:130698–130720，2021年。Flavio Chierichetti、Ravi Kumar、Silvio Lattanzi 和 Sergei Vassilvitskii。通过公平子集实现公平聚类。载于第*30*届神经信息处理系统年会（*NeurIPS*）论文集，第5029–5037页，2017年。Ofer Dekel、Felix Fischer 和 Ariel D. Procaccia。激励相容回归学习。计算机与系统科学杂志，76(8):759–777，2010年。Soroush Ebadian 和 Evi Micha。通过比例代表制促进抽选。载于第*24*届自主智能体与多智能体系统国际会议（*AAMAS*）论文集，第667–675页，2025年。Ziming Fan、Nianli Peng、Muhang Tian 和 Brandon Fain。多目标强化学习中的福利与公平性。载于第*22*届自主智能体与多智能体系统国际会议（*AAMAS*）论文集，第1991–1999页，2023年。Safwan Hossain 和 Nisarg Shah。策略噪声对线性回归的影响。自主智能体与多智能体系统，35(2):21，2021年。Safwan Hossain、Evi Micha 和 Nisarg Shah。面向多智能体多臂老虎机的公平算法。载于第*34*届神经信息处理系统年会（*NeurIPS*）论文集，第24005–24017页，2021年。Leon Kellerhals 和 Jannik Peters。聚类中的比例公平：社会选择视角。载于第*37*届神经信息处理系统年会（*NeurIPS*）论文集，第111299–111317页，2024年。Markelle Kelly、Rachel Longjohn 和 Kolby Nottingham。UCI机器学习知识库。https://archive.ics.uci.edu，2025年。访问日期：2025-05-15。Anilesh Krishnaswamy、Zhihao Jiang、Kangning Wang、Yu Cheng 和 Kamesh Munagala。惠及所有人的公平性：分类的尽力而为公平性保证。载于第*24*届人工智能与统计国际会议（*AISTATS*）论文集，第3259–3267页，2021年。Bo Li、Lijun Li、Ankang Sun、Chenhao Wang 和 Yingfan Wang。聚类的近似群体公平性。载于第*38*届国际机器学习会议（*ICML*）论文集，第6381–6391页，2021年。Lily Li、Evi Micha、Aleksandar Nikolov 和 Nisarg Shah。公平地为朋友分组。载于第*37*届*AAAI*人工智能会议（*AAAI*）论文集，第5747–5754页，2023年。Huawen Liu、Xindong Wu 和 Shichao Zhang。使用层次特征聚类进行特征选择。载于第*20*届*ACM*信息与知识管理国际会议（*CIKM*）论文集，第979–984页，2011年。Reshef Meir、Ariel D. Procaccia 和 Jeffrey S. Rosenschein。策略无关分类算法。人工智能，186:123–156，2012年。Evi Micha 和 Nisarg Shah。比例公平聚类再探。载于第*47*届自动机、语言与编程国际研讨会（*ICALP*）论文集，第85:1–85:16页，2020年。

D Napoleon 和 S Pavalakodi。针对高维数据集，使用 k 均值聚类算法进行降维的一种新方法。*International Journal of ComputerApplications*，13(7):41–46，2011年。Javier Perote 和 Juan Perote-Pe˜na。简单回归的策略无关估计量。*Mathematical SocialSciences*，47:153–176，2004。Dominik Peters、Grzegorz Pierczy´nski 和 Piotr Skowron。具有可加效用的比例参与式预算。载于神经信息处理系统第*34*届年会（*NeurIPS*）论文集，第12726–12737页，2021年。Yu Yang、Siddhartha Mishra、Jeffrey Chiang 和 Baharan Mirzasoleiman。Smalltolarge (s2l)：通过总结小模型的训练轨迹，为大语言模型微调实现可扩展的数据选择。载于神经信息处理系统第*37*届年会（*NeurIPS*）论文集，第83465–83496页，2024年。Guojun Zhang、Saber Malekmohammadi、Xi Chen 和 Yaoliang Yu。联邦学习中的比例公平。*Transactions on Machine Learning Research*，109(4):219–224，2023年。

附录

贪心捕获算法

在第 3 节讨论的三个维度中，它们导出了贪心捕获的八种变体，其中质心贪心捕获（算法1）作出如下选择：

（1）球围绕质心生长。（2）球“捕获”一群 n/k 个点并开放后，只要包含任何尚未捕获的点，就会继续生长并捕获这些点。（3）所有代理都被捕获后，每个代理都有机会切换到包含距其最近质心的簇。

我们可以证明，即使禁用特征（2）和（3）中的一个，该算法相对于质心损失的 $(1+\sqrt{2})$-核保证仍然成立。

定理9。考虑贪心捕获的一种变体，其中球围绕代理生长，且开放的球继续生长，和/或允许代理在算法结束时切换簇（至少满足其中一项）。那么，该算法始终返回一个相对于质心损失属于 $(1+\sqrt{2})$-核的聚类。

证明。反设结论不成立，令 $\mathcal{X}=\{(C_1,x_1),\ldots\ldots,(C_k,x_k)\}$ 为此类算法得到的聚类，并假设存在一个偏离簇（S，y），其中 $|S|\geqslant\lceil\frac{n}{k}\rceil$ 且对于每个 $i\in S$，都有 $(1+\sqrt{2})\cdot d(i,y)<d(i,x_{\mathcal{X}(i)})$。

令 $i=\arg\min_{i'\in S}\mathcal{X}(i')$ 为 S 中第一个被该算法捕获的代理，并令 $t=\mathcal{X}(i)$。我们知道 $x_t\neq y$，否则 i 显然不会属于偏离簇 S。令 $j=\arg\max_{j'\in S}d(j',y)$ 为 S 中距离 y 最远的代理，并令 $\mathcal{X}(j)=t'$。

首先，注意到必有 $d(j,y)\geqslant d(i,x_t)$。如果并非如此，且 $d(j,y)<d(i,x_t)$，那么根据贪心捕获的运作方式，簇 (S,y) 会在 i 被 x_t 捕获之前被捕获，这将导致矛盾。

利用上述不等式，我们可以证明 $t\neq t'$。如果情况并非如此，且 $t=t'$，那么我们可以得到如下结论：

$$\begin{aligned}(1+\sqrt{2})\cdot d(j,y)&<d(j,x_{t'})=d(j,x_t)\\&\leqslant d(i,x_t)+d(i,y)+d(j,y)\ \ //\text{Triangle inequality}\\&<d(i,x_t)+\frac{1}{1+\sqrt{2}}\cdot d(i,x_t)+d(j,y)\\&\leqslant\left(\frac{1}{1+\sqrt{2}}+2\right)\cdot d(j,y).\end{aligned}$$

因此，$1+\sqrt{2}<\frac{1}{1+\sqrt{2}}+2=1+\sqrt{2}$，这就产生了矛盾。

我们还可以证明 $d(j,x_t)\geqslant d(j,x_{t'})$。我们可以分两种情况来讨论。

情况 1：开放球持续增长。如果所讨论的贪心捕获变体中开放球持续增长，且 $d(j,x_t)<d(j,x_{t'})$，那么，由于 x_t 在 $x_{t'}$ 捕获 j 之前捕获了 i，因此在算法中 $x_{t'}$ 捕获 j 时，x_t 已经捕获了 i，而且在 δ 的某个先前取值下，j 已经位于 x_t 的球半径范围内。这意味着 x_t 一开始就必定捕获了 j。但这与 $\mathcal{X}(j)=t'$ 矛盾，因为无论是否允许代理在算法的最后一步切换簇，j 在算法结束时都不可能落入第 t' 个簇。

情况2：代理可以切换簇。如果在所考虑的贪心捕获变体结束时，代理可以切换簇，且 $d(j,x_t)<d(j,x_{t'})$，那么无论开球是否继续增长，j总会想要切换到x_t，而不是$x_{t'}$，这与 $\mathcal{X}(j)=t'$ 矛盾。

根据这个不等式，我们可以得出以下结论：

$$\begin{aligned}(1+\sqrt{2})\cdot d(j,y) &< d(j,x_{t'})\\ &\leqslant d(j,x_t)\\ &\leqslant d(i,x_t)+d(i,y)+d(y,j)\\ &< d(i,x_t)+\frac{1}{1+\sqrt{2}}\cdot d(i,x_t)+d(j,y)\\ &\leqslant \left(\frac{1}{1+\sqrt{2}}+2\right)\cdot d(j,y),\end{aligned}$$

再次得到 $1+\sqrt{2} < \frac{-1}{1+\sqrt{2}} + 2 = 1+\sqrt{2}$，这构成矛盾。 □

B 平衡聚类的不可能性结果

本节针对半中心世界中的平衡聚类给出两个不可能性结果。直观而言，平衡性要求代理人集合的划分中各簇的大小大致相等。在许多应用中，例如为课程项目组建学生群组，或在会议上安排口头报告场次，这可能是一个理想的判据。

形式化地说，可以将一个聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$ 称为平衡的，若 $|C_t| \in \{\lfloor n/k \rfloor, \lceil n/k \rceil\}$对所有 $t \in [k]$ 都成立（等价地，$||C_t| - |C_{t'}|| \leqslant 1$ 对所有 $t, t' \in [k]$都成立）。然而，即使采用以下较弱判据，我们的两个不可能性结果仍然成立。

定义4（平衡聚类）。 我们称聚类 $\mathcal{X} = \{(C_1, x_1), \ldots, (C_k, x_k)\}$ 为平衡的，如果当k整除n时， $|C_t| = n/k$ 对所有$t \in [k]$都成立。

令人欣慰的是，非质心贪心捕获确实能在这一宽松定义下生成平衡聚类。[6] 然而，质心贪心捕获无法提供此类保证。这是因为，尽管它仍会在捕获 $\lceil n/k \rceil$ 个未捕获代理后立即开启球，但（i）开放球会继续扩张并捕获更多代理；（ii）最后，代理会被重新分配给最近的中心。在某种程度上，我们用于双度量损失的算法（算法 3）和用于加权单度量损失的其中一种算法（算法 6）至少保留了第二个特点。这使得最终各簇的大小相当难以预测（而且不一定大致相等）。

话虽如此，算法 5先执行非质心贪心捕获，再进行贪心质心选择，确实会生成平衡聚类，并且在加权单度量损失下达到 $\frac{2}{\lambda}$-核心。然而，当 λ 趋近于 0（质心世界）时，这一结果会变得无界。下一个结果表明，若要求聚类平衡，这种情况无法避免。因此，在 算法 6中，我们计算不平衡聚类，以便在小 λ 区间内实现有限的核心近似（从而在 $\lambda \in [0,1]$的整个范围内获得有限的核心近似），这是必要的。

定理 10. 对于参数为 $\lambda \in [0,1]$ 的加权单度量损失，存在一个实例，使得对于任何 $\alpha < \frac{1}{\sqrt{\lambda}}$，都不存在属于 α-核心的平衡聚类。

证明。 考虑图 1中的实例。注意，$k = 3$ 整除$n = 6$。令$p = \frac{1}{\sqrt{\lambda}}$。为导出矛盾，假设存在一个平衡聚类 $\mathcal{X} = \{(C_1, x_1), (C_2, x_2), (C_3, x_3)\}$，满足 $|C_1| = |C_2| = |C_3| = 2$，且它属于某个 $\alpha < \frac{1}{\sqrt{\lambda}}$ 的 α-核心。

[6] 当一个球覆盖至少 $\lceil n/k \rceil$ 个未捕获代理时，由于它们到中心的距离相等，该球可能实际上覆盖更多代理。不过，在这种情况下，可以修改算法，让球恰好捕获球内 $\lceil n/k \rceil$ 的未捕获代理，而将其余代理留在边界上，以便之后由其他球捕获。这种修改不会影响算法所达到的任何保证。此外，它会按照 定义4 生成平衡聚类。注意，当 k 不能 整除 n 时，最后一个非空簇甚至可能包含少于 $\lfloor n/k \rfloor$ 个代理。

首先，注意 $\{a, b\}$ 必须是其中一个簇。为导出矛盾，假设情况并非如此。那么，代理 a 和 b 中至多一个能与 c 聚在一起（即 $\mathcal{X}(a) = \mathcal{X}(c)$ 和 $\mathcal{X}(b) = \mathcal{X}(c)$ 中至多一个成立）。不失一般性，假设代理 a 与 c 聚在一起（若代理 b 与 c 聚在一起，论证对称）。于是，$\ell_a(\mathcal{X}) \geqslant \lambda \cdot p = \sqrt{\lambda}$；仅非质心损失部分就足以得到这个下界。由于代理 b 与 $\{d, e, f\}$ 中一个代理聚在一起，而它与该代理相距无穷远，因此 $\ell_b(\mathcal{X}) = \infty$。接着，考虑偏离簇 $(\{a, b\}, a)$，它使 $\ell_a(\{a, b\}, a) = \lambda$，且 $\ell_b(\{a, b\}, a) = 1$。与 $\mathcal{X}$ 相比，代理 b 的损失改善了无穷倍，代理 a 的损失改善了 $\frac{\sqrt{\lambda}}{\lambda} = \frac{1}{\sqrt{\lambda}}$ 倍，这与 $\mathcal{X}$ 属于某个 $\alpha < \frac{1}{\sqrt{\lambda}}$ 的 α-核心这一事实相矛盾。

因此，$\{a, b\}$ 必须是其中一个簇，并且代理 a 和 b 中至少有一个不是该簇的质心。不失一般性，假设质心不在代理 b 处。接下来，考虑偏离簇 $(\{b, c\}, b)$。$\ell_b(\{b, c\}, b) = \lambda \cdot p = \sqrt{\lambda}$ 且 $\ell_c(\{b, c\}, b) = p$。相比之下，$\ell_b(\mathcal{X}) \geqslant 1$，因此代理 b 的表现至少提升了 $\frac{1}{\sqrt{\lambda}}$ 倍。此外，由于 $\{a, b\}$ 是一个簇，代理 c 必须与 $\{d, e, f\}$ 中的某个代理聚在一起，而代理 c 与该代理相距无限远，故得到 $\ell_c(\mathcal{X}) = \infty$。这意味着代理 c 的表现提升了无限倍。这与 $\mathcal{X}$ 对某个 $\alpha < \frac{1}{\sqrt{\lambda}}$ 属于 α-核的事实相矛盾。至此证明完成。 □

由于更广泛的双度量损失类别包含所有 $\lambda \in [0, 1]$ 的加权单度量损失，因此可以得出结论：不可能设计出一种均衡聚类算法，使其在双度量损失下实现对核的任何有限近似。

定理 11. 对于双度量损失，任何始终生成平衡聚类的算法，对于任意 $\alpha \geqslant 1$，都无法实现 α-核心。

证明。假设存在矛盾，即存在一种算法，始终能在双度量损失下生成属于 α-核心的平衡聚类。接着，考虑图 1 中的实例，并将非质心度量设为 $d^m = \lambda \cdot d$，将质心度量设为 $d^c = (1 - \lambda) \cdot d$，其中任意 $\lambda < 1/\alpha^2$ 均成立。由此导出的双度量损失为 $\ell_i(C, x) = \lambda \cdot \max_{j \in C} d(i, j) + (1 - \lambda) \cdot d(i, x)$，这恰好是由距离度量 d 和参数 λ 导出的加权单度量损失。根据定理 10，任何平衡聚类所达到的核心近似（包括该算法生成的聚类）都必须至少为 $\frac{1}{\sqrt{\lambda}} > \alpha$，这就得到了所需的矛盾。 □

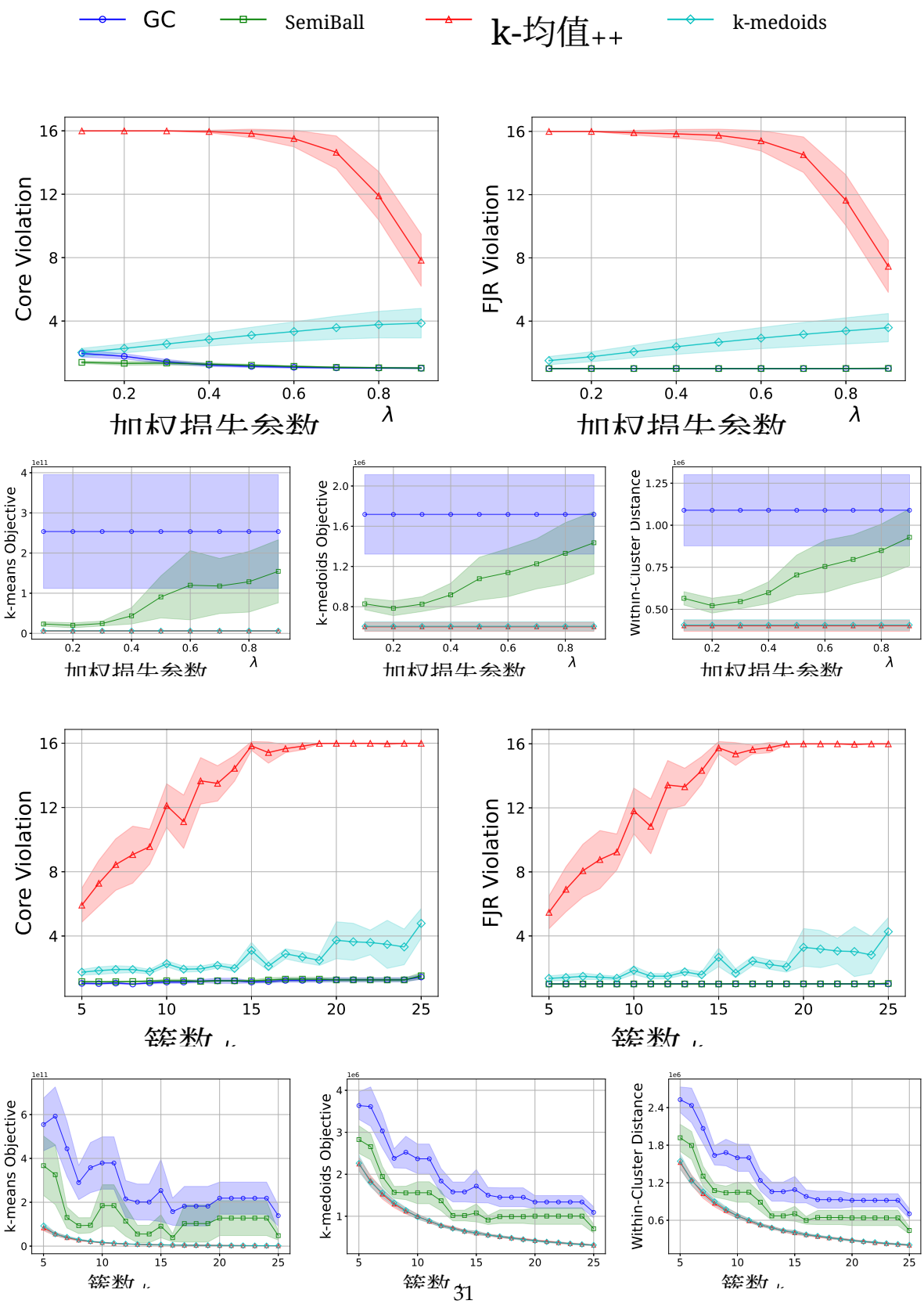

图 2：*Adult*（人口普查收入）数据集的结果。通用图例位于顶部。上方五个图表比较了在改变 $\lambda \in \{0.1、0.2、\ldots、0.9\}$ 并固定 $k = 15$ 时，各算法在我们的五项指标上的表现；下方五个图表对应于固定 $\lambda = 0.5$ 并改变 $k \in \{5, 6, \ldots, 25\}$。

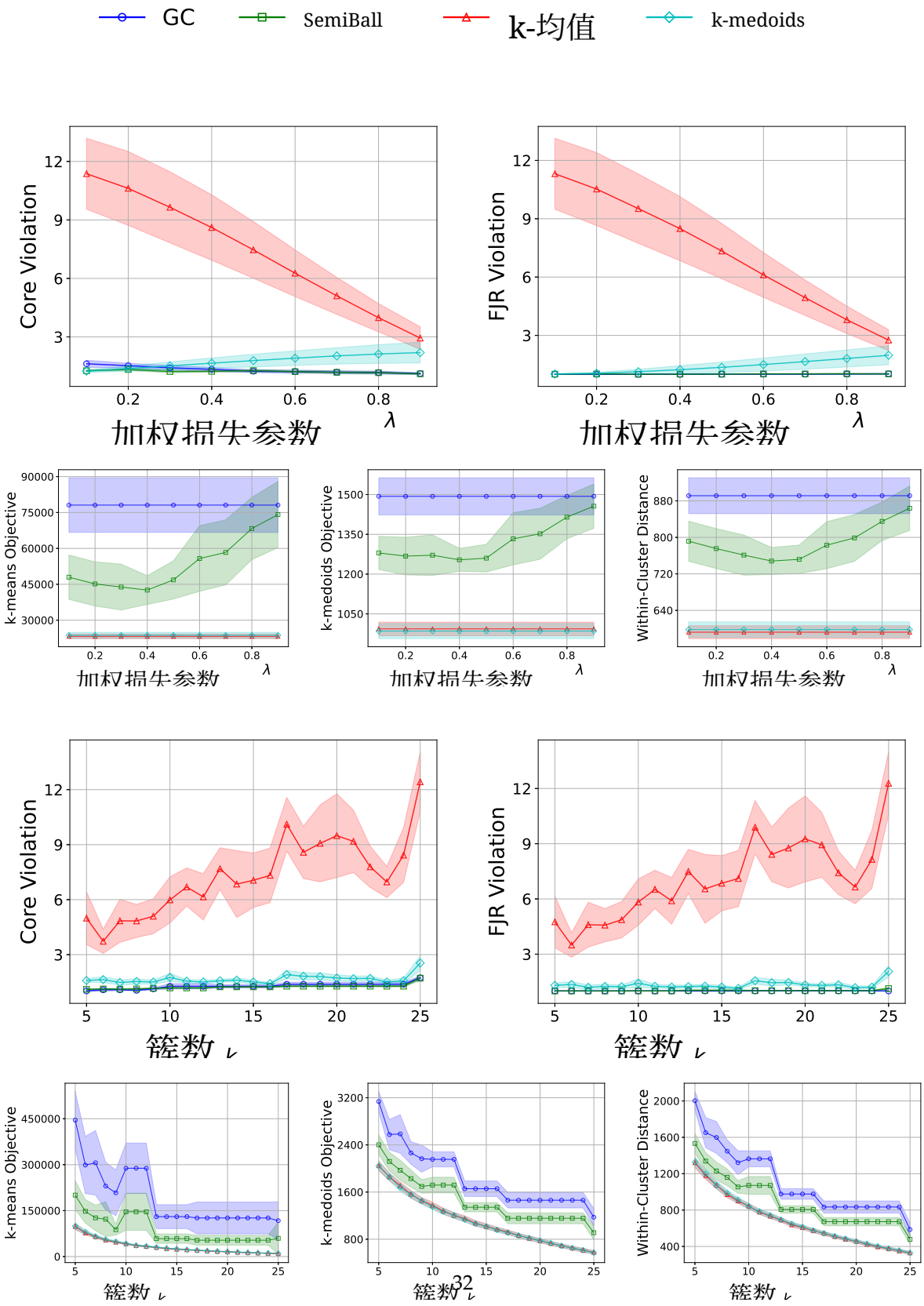

图3：*Diabetes* 数据集的结果。公共图例位于顶部。上方五个图表比较了在改变 $\lambda \in \{01、0.2、\ldots、0.9\}$ 并固定 $k = 15$ 时，各算法在我们的五个指标上的表现；下方五个图表则对应于固定 $\lambda = 0.5$ 并改变 $k \in \{5, 6, \ldots, 25\}$。

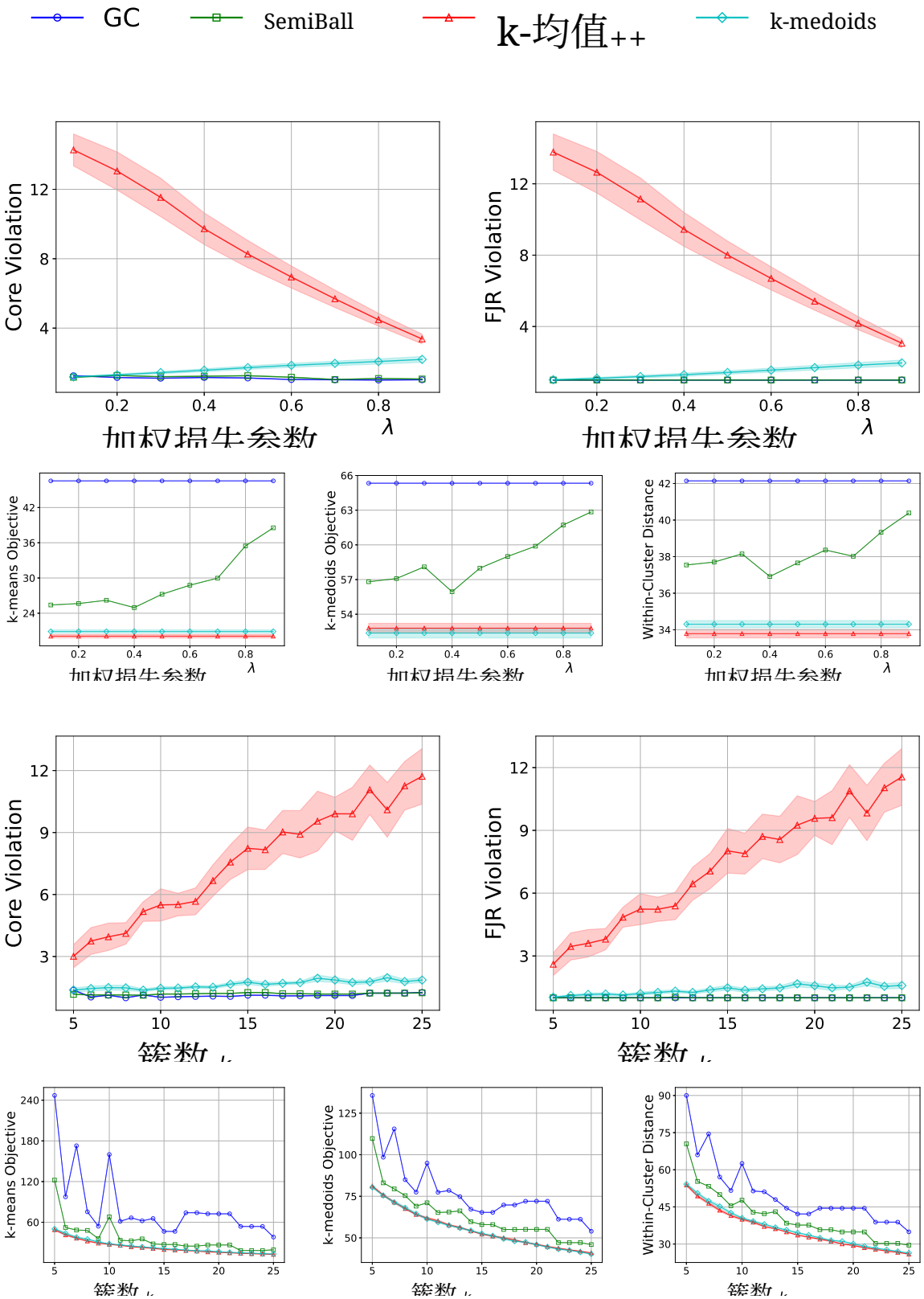

图4：鸢尾花 数据集的结果。公共图例位于顶部。上方五个图表比较了在改变 $\lambda \in \{01、0.2、\ldots、0.9\}$ 并固定 $k = 15$ 时，各算法在我们的五个指标上的表现；下方五个图表则对应于固定 $\lambda = 0.5$ 并改变 $k \in \{5, 6, \ldots, 25\}$。