FINEX：一种用于精确且灵活的基于密度的聚类的快速索引（含证明的扩展版）*

康斯坦丁·埃米尔·蒂尔，奥地利萨尔茨堡大学丹尼尔·科赫，奥地利萨尔茨堡大学尼古拉斯·奥格斯滕，奥地利萨尔茨堡大学托马斯·许特，奥地利萨尔茨堡大学维利·曼，德国 Celonis SE丹尼尔·乌尔里希·施密特，奥地利萨尔茨堡大学

基于密度的聚类旨在给定数据集中发现相似对象（即簇）的群组。其应用包括流程挖掘和异常检测等。该方法包含两个用户参数（ε，$MinPts$），它们决定聚类结果，但通常事先未知。因此，用户需要交互式地测试各种设置，直到找到令人满意的聚类结果。然而，现有解决方案存在以下局限性：(a) 无效*ff*的剪枝，导致昂贵的邻域计算效率低下。*(b)*近似聚类，会将对象错误地标记为噪声。(c) 受限的参数调优仅限于 ε，而 $MinPts$ 保持不变，从而减少了可探索的聚类结果。(d)缺乏*fl*灵活性，无法适用于多种数据类型和距离函数。

我们提出了 FINEX，这是一种线性空间索引，可克服这些局限。我们的索引提供精确聚类，并且可以使用两个参数中的任意一个进行查询。FINEX 会尽可能避免邻域计算，并利用基于密度的簇的基本性质，降低其余计算的复杂度。因此，我们的解决方案在数据类型和距离函数方面既高ffi效又 fl灵活。此外，FINEX 遵循基于密度的聚类原本直观简洁的理念。在针对来自各个领域的 12 个大型真实世界数据集进行的实验中，FINEX 在精确聚类方面的性能经常远超最先进的技术，且优势达到数量级。

CCS 概念：• **信息系统 → 聚类**; **聚类与分类fi**.

附加关键词和短语：基于密度的聚类、索引、OPTICS、DBSCAN

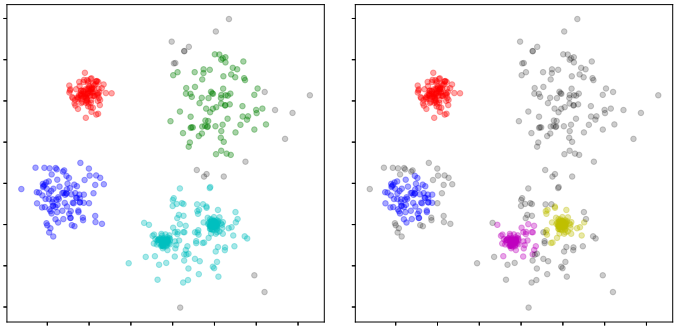
1 引言

聚类旨在识别数据集中的相似对象群组（即簇）。这是一项至关重要的任务，因为它有助于探索和了解数据集的特征 [11]。在基于密度的聚类中，这些簇是由低密度区域分隔开的高密度对象区域。广受欢迎的 DBSCAN [8] 算法可计算此类聚类，并已成功应用于天文学 [19] 和异常检测 [23]等多个领域 fi。尤其是，我们的研究受到流程挖掘领域的启发；在该领域中，DBSCAN 簇用于分析大量以集合形式建模的业务流程 [15]。DBSCAN 有两个用户参数（ε，$MinPts$），用于定义簇的最小密度：在半径为 ε 的范围内至少有 $MinPts$ 个邻居的对象会被归类为核心对象，并从其所在位置生成或扩展簇。如果某个邻居数量较少的对象与核心对象的距离超过 ε，则会被视为噪声。

DBSCAN 的局限性。由于对聚类结果影响重大，选择 (ε，$MinPts$) 并不容易。有时，不存在能够捕捉所有有意义簇的唯一参数设置：如图 1 所示，图中展示了在不同参数下得到的两种基于密度的聚类。图 1a 中右上方的簇（绿色）所需的密度低于图 1b 中右下方的簇（黄色和紫色）。如果没有先验知识，用户就必须在不同设置下反复运行 DBSCAN，直到 fi找到合适的聚类。在大数据时代，这给 DBSCAN 带来了

*这是即将在 ACM SIGMOD 2023 上发表的论文的扩展版本。请引用原始的 SIGMOD 版本。

严重问题，因为它效率低下。若没有邻域索引，该算法需要进行 $O(n^2)$ 次距离计算，这使得交互式聚类无法应用于大型数据集。例如，即使使用最先进的邻域索引，在我们的实验中，对 CELONIS-1 数据集（参见第 6 节）进行聚类也需要长达 9 小时；对于每天要处理数千次聚类查询的流程挖掘公司而言，这个耗时长得令人难以接受。



(a) $\varepsilon = 2$.　　(b) $\varepsilon = 1$.

图 1. 二维空间中采用两种不同ff值进行的基于密度的聚类。两种聚类结果都可能有意义。

最先进的索引。 OPTICS 技术 [2] 试图缓解预先选择距离阈值的问题：它计算出一种簇排序，可用作索引，在给定某个阈值 $\varepsilon^* \leq \varepsilon$进行查询时返回聚类结果。然而，OPTICS 只能提供近似的 聚类结果，因为它会错误地将一些簇对象标记为噪声。在需要精确结果以正确区分噪声与簇的场景中（例如异常检测 [23]），这是一个关键问题。此外，OPTICS 不允许调整 $MinPts$，这限制了它的适用性。

FINEX。 我们提出 FINEX，这是一种**快**速、线性空间**索**引，适用于**精**确且 fl灵活的基于密度的聚类方法，可适用于任何数据类型和距离函数。它克服了 DBSCAN 的复杂度问题和 OPTICS 的准确性问题，同时不引入额外限制或开销。这使 FINEX 有别于其他方法：这些方法要么偏离基于密度的聚类原始定义 [6],仅适用于特fi定场景 [9]，要么需要额外的用户参数 [16]。

贡献. 我们的贡献旨在实现对大型数据集进行交互式密度　　y-基聚类　　g大　　g型数据集：

(1) 精确性与参数调优： FINEX 支持两种新的查询类型： ε^*-查询和 $MinPts^*$-查询，它们返回精确的聚类。针对一个生成（$\varepsilon, MinPts$）对构建 FINEX 后，用户可以查询关于（$\varepsilon^*, MinPts$）且满足 $\varepsilon^* \leq \varepsilon$ 的所有聚类，也可以查询关于（$\varepsilon, MinPts^*$）且满足 $MinPts^*$ $MinPts$ 的聚类。与 OPTICS 不同，FINEX 能够正确识别所有簇对象。

(2) 效*ffi*率与灵活性： 与 DBSCAN 相比，FINEX 能够快速回答所有查询，且无需从头重新计算。其效ffi率基于这样一个事实：稠密簇是稀疏簇的子集，这一结论适用于任何数据类型和距离函数。FINEX 利用这一事实减少邻域计算的次数和成本，并且仅依赖基于密度的聚类的标准要求（即，对象对之间的对称距离函数）。

(3)快速线性空间索引： FINEX旨在成为一种内存中的数据结构，其空间复杂度与对象数量呈线性关系。具体而言，我们的索引是数据集的一个排列，其中每个对象都配备了两个

特殊距离度量、密度计数以及对其最密邻居的引用。FINEX 利用这些属性识别可能被错误标记为噪声的簇对象，并在稀疏聚类分解为多个稠密簇时查找连通分量。

(4) 线性时间聚类： 与 OPTICS 一样，FINEX 能够针对 $\varepsilon^* \leq \varepsilon$进行线性时间近似聚类。因此，FINEX 返回的聚类至少与 OPTICS 的聚类一样准确。此外，与 OPTICS 不同，如果查询参数与生成 FINEX 时所用的 (ε, $MinPts$) 参数对完全相同，我们可以在线性时间内提供精确聚类。

(5) 实验评估： 我们通过在 12 个大型真实世界数据集上进行实验评估，补充了理论结果。这些数据集来自不同领域，包含集合型对象或多维向量对象，并使用 Jaccard 距离和欧几里得距离。我们将 FINEX 的聚类运行时间与两种从头计算精确聚类的竞争算法进行比较：经过高度优化的 DBSCAN，以及最先进的算法 AnyDBC [16]。此外，我们还比较了 FINEX 和 OPTICS 的索引构建时间与准确率。结果表明，FINEX 在几乎所有场景中都优于竞争算法，优势往往达到数量级。

论文结构. 本文其余部分组织如下：第 2 节概述相关文献，第 3 节介绍最重要的定义，并讨论现有解决方案的局限性。第 4 节中，我们对 OPTICS 进行严格的形式化定义，并研究簇排序的基本性质，为提出我们的解决方案奠定基础。随后，第 5 节介绍 FINEX 及其理论保证。为支持我们的理论发现，我们在第 6 节通过实验将 FINEX 与最先进的解决方案进行比较。最后，我们在第 7 节总结全文。此外，我们在附录 A 中给出了所有理论结果的证明。

2 相关工作

据我们所知，FINEX 首次提出了 *first* 线性空间数据结构，可高效提供精确 密度型聚类。尽管 DBSCAN [8] 也能生成精确聚类，但每当输入参数发生变化时，都必须从头重新计算。定义 3.5 中引入了精确性的形式化定义，遵循密度型聚类的原始概念。

索引结构。 OPTICS 算法 [2] 采用线性簇排序 来提供稠密簇。事实上，FINEX 和 OPTICS 利用的是同一项基本性质：稠密簇是稀疏簇的子集，可以通过排序进行编码。然而，OPTICS 并未充分利用这一性质，因此无法报告精确聚类。FINEX 提供了这一功能，并进一步支持调整 $MinPts$ 参数，从而扩展可探索解空间。此外，FINEX 的线性时间近似聚类比 OPTICS 得出更准确的结果。

开发 FINEX 需要对簇排序进行全面的形式化。Achtert 等人 [1] 率先在其工作中对 OPTICS 进行形式化，以实现对预计算的 OPTICS 排序的更新。虽然他们的形式化无法直接用于我们的目的，但其基本思路启发了我们的形式化方法。

Gan 和 Tao [9] 基于不同的形式化方法，开发了一种高效的 OPTICS 近似版本，但该版本仅适用于向量数据和 Lp范数。相比之下，FINEX 能够生成精确结果，并且在数据类型和距离函数方面具有灵活性。

高***ffi***效的邻域计算。实现交互式基于密度的聚类的自然步骤，是加速ε-邻域计算，因为它们占据了DBSCAN运行时间的主要部分。现有用于高效邻域计算的数据结构，例如用于向量数据的 R 树 [10] 或kd 树 [4]，以及用于度量空间的 M 树 [7]。Mann 等人 [18] 比较

多种用于集合数据的先进技术。由于 FINEX 支持与 DBSCAN 相同的数据类型和距离函数，因此可以与其中任何一种技术结合使用。

邻域计算剪枝。 FINEX 的效率源于，相较于 DBSCAN，它减少了邻域计算的次数；DBSCAN 会为数据集中的所有对象计算邻域。AnyDBC [16, 17]算法也只对部分对象执行邻域计算。这些对象会生成部分簇，并依次连接，直到得到精确聚类。与 FINEX 相比，AnyDBC 需要额外的用户参数，且仅适用于度量距离函数（这些函数必须满足三角不等式）。

过滤方法。 Brecheisen 等人 [5] 发现，使用真实距离函数计算出的簇是使用下界距离函数计算出的簇的子集。因此，聚类本身可以用作另一种聚类的过滤器。然而，Brecheisen 等人利用这一结果计算一种特定的聚类，而 FINEX 则可检索任意数量的聚类。

其他方法。 Jang 和 Jiang [13] 建议通过抽取对象的随机样本，并仅为这些对象计算邻域，来加速基于密度的聚类。与 FINEX 相比，这种方法得到的是精确聚类的近似结果，并且需要对统计数据分布作出额外假设。

Campello 等人 [6] 提出了一种替代性的基于密度的聚类概念，其中簇仅由核心对象组成。虽然该概念避免了OPTICS的近似问题，因为 OPTICS相对于核心对象是精确的（参见第4 节），但其方法相对于原始的基于密度的聚类概念仍然是一种近似。FINEX 在这两种情形下都提供了精确的解决方案。

3 预备知识

本节回顾基于密度的聚类及其最重要的定义。此外，我们介绍 DBSCAN 所计算的精确聚类的新定义。随后，我们解释 OPTICS 算法，以及对其计算的索引结构至关重要的两个量：核心距离和可达距离。最后，我们通过示例说明 OPTICS 的局限性，并介绍本文如何克服这些局限。

3.1 基于密度的聚类

基于密度的聚类在支持的数据类型和距离函数方面具有灵活性。它只需要一个数据集 D和一个对称距离函数 d： $D \times D$！ $\mathbb{R}_{\geq 0}$。密度由两个参数定义：(1) 一个距离阈值$\varepsilon \in \mathbb{R}_{\geq 0}$，用于定义数据集 D中对象 p 的 ε-邻域 $N_\varepsilon($ $p)$：

$$N_\varepsilon(p) = \{q \in D \ : \ d(p,q) \leq \ \varepsilon\}$$

(2)一个规模阈值，称为 $MinPts \in \mathbb{N}$，用于定义对象的核心对象 状态：在其 ε-邻域中至少有 $MinPts$ 个对象的对象称为核心对象。邻域小于 $MinPts$ 且位于另一个核心对象的ε-邻域中的对象，被称为一个簇的边界对象 。注意， $p \in N_\varepsilon(p)$始终成立。簇、核心对象和边界对象之间的关系在定义 3.1–3.4 中有所阐述。

Definit 义*3.1*（直接密度可达 *[8]*）。令 $p, q \in D$。 p 是 直接密度可达 于 q 关于（ε， $MinPts$）。 if

(1) $|N_\varepsilon(q)|$ $MinPts$
(2) $p \in N_\varepsilon(q)$

对象只有在作为核心对象时，才能直接密度可达，这意味着这种关系并不对称。以下定义是对这一概念的自然延伸。

定义*fi 3.2*（密度可达 *[8]*）。令 $p, q \in D$。 p是 密度可达 自 q关于（ε， $MinPts$）当且仅当 $\exists p_1, \ldots,$ $p_n \in D$使得

(1) $p_1 = q, p_n = p$(2) p_{i+1} 相对于 $(\varepsilon, MinPts)$ 从 p_i 直接密度可达，对所有 $i = 1, \ldots\ldots, n$ 1

定义*fi 3.3*（密度连接 *[8]*）。令 p、 $q \in D$。 p 关于 $(\varepsilon, MinPts)$与 q 密度连接，当且仅当存在 $\exists o \in D$，使得 p和 q均关于 $(\varepsilon, MinPts)$ 从 o 密度可达。

密度可达是一种传递关系，而密度连接还具有对称性。有了这些概念，我们现在可以给出基于密度的簇的形式化定义。

定*fi*义 *3.4*（簇与噪声 *[8]*）。任何满足以下条件的非空子集$K \surd D$ 均为关于（ε， $MinPts$）的簇：

（1）极大性： $\forall p$， $q \in D$：如果 $p \in K$且 q关于（ε， $MinPts$）从 p密度可达，则 $q \in K$也成立。
（2）连通性： $\forall p$， $q \in K$： p关于（ε， $MinPts$）与 q密度连接。
不包含在任何簇中的每个对象都被视为噪声。

请注意，可能存在一些边界对象，它们分别从不*ff*同簇的核心对象密度可达。根据定义 3.4，我们将此类边界对象称为有歧义的，因为它们属于多个簇。

DBSCAN 的精确性。 DBSCAN 算法通过检测一个未处理的核心对象 $o \in D$，并创建一个新簇，其中包含相对于 $(\varepsilon,MinPts)$从 o 密度可达的所有对象，从而为数据集 D 计算基于密度的聚类。DBSCAN 在所有对象都已处理后终止，因此所有簇都已计算完成。请注意，该算法会将所有密度连接的对象分配到恰好一个簇中。（有歧义的）边界对象会被分配到由最先检测到它们的核心对象定义的簇fi中 fi。因此，DBSCAN 的结果就是我们在新的精确聚类 定义 3.5 中所称的精确聚类。

定*fi*义*3.5*（精确聚类）。 令$K_1, \ldots\ldots, K_n$为根据定义3.4，相对于（ε， $MinPts$）的 D的基于密度的簇。精确聚类 是 $E_1, \ldots\ldots, E_n$对 D的划分，使得

(1) $E_i \surd K_i$(2) $8K$的所有核心对象都在 E_i 中；(3) 每个边界对象都恰好位于一个与其所属簇对应的分区中。

换言之，精确聚类与基于密度的聚类之间的区别在于，精确聚类会将每个有歧义的边界对象分配给且仅分配给一个簇。

3.2 最先进的索引技术

OPTICS 算法 [2] 旨在让基于密度的聚类更具交互性。其思路是，只需运行一次 OPTICS，就能替代多次运行 DBSCAN。与 DBSCAN 一样，用户需要选择参数$\varepsilon \in \mathbb{R}_{\geq 0}$ 和 $MinPts \in \mathbb{N}$。随后，OPTICS 会计算出一个簇排序，其中包含了关于满足 $0 \leq \varepsilon^* \leq \varepsilon$ 的任意 $(\varepsilon^*,MinPts)$ 基于密度的聚类的信息。然而，OPTICS 排序也遗漏了关于这些聚类的大量信息（参见第3.3节）。为了理解这一点，我们 fi首先在定义 3.6–3.8 中介绍该排序的组成部分。

定义 *3.6*（*MinPts* 距离）。

$$M(p) = \min\{\delta \in \mathbb{R}_{\geq 0} : |N_\delta(p)| \geq MinPts\}$$

定义*fi 3.7*（核心距离的 $p \in D$ *[2]*）。

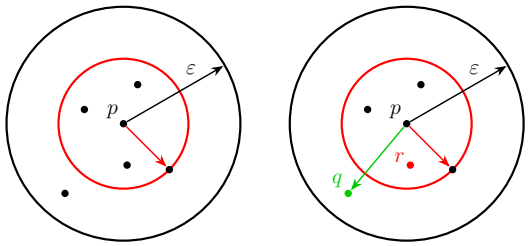
$$C_{\varepsilon,MinPts}(p) = \begin{cases} M(p), & \text{if } |N_\varepsilon(p)| \geq MinPts \\ \infty, & \text{otherwise} \end{cases}$$

考虑任意核心对象 $p \in D$并令 $\delta < C_{\varepsilon,MinPts}(p)$。由此可知， $|N_\delta(p)| < MinPts$。因此， p 的核心距离是使得 p 仍为核心对象的最小距离。噪声和边界对象的核心距离始终为 ∞。

定义 *3.8*（ $p \in D$ *[2]*下 $q \in D$ 的可达距离）

$$R_{\varepsilon,MinPts}(q,p) = \begin{cases} \max\{C_{\varepsilon,MinPts}(p), d(p,q)\}, & \text{if } |N_\varepsilon(p)| \geq \text{MinPts} \\ \infty, & \text{otherwise} \end{cases}$$

可达距离的定义与密度可达的概念有关：令 $p \in D$ 为核心对象，且 $q \in N_\varepsilon(p)$，那么 $R_{\varepsilon,\ MinPts}(q,\ p)$ 是使 q 仍可从p直接密度可达的最小距离阈值。在任何更小的距离下， q 要么会位于 p的邻域之外，要么 p 会失去其核心属性，也就是说，没有对象能从 p直接密度可达。请注意，一般而言，$R_{\varepsilon,\ MinPts}(q,\ p) \neq R_{\varepsilon,\ MinPts}(p,\ q)$。图2展示了这些距离。



(a) 红色： p 的核心距离。 (b) 红色： r 相对于 p 的可达距离。绿色： p 相对于 q 的可达距离。

图2.以 $MinPts = 5$为例说明核心对象距离和可达距离（参见定义 3.7 和 3.8）。

OPTICS 的技巧。 OPTICS 利用了稠密簇完全包含于稀疏簇中的事实。然而，Ankerst等人 [2] 仅提供了对这一性质的直观解释，缺少形式化证明。我们通过命题 3.9 补充了他们的工作。证明见附录A。

Proposition 3.9 (ε-Nested Clusters)。 设 $0 \leq \varepsilon^* \leq \varepsilon$。对于关于 $(\varepsilon^*, MinPts)$ 的每个簇 $^* \subseteq D$，都存在关于 $(\varepsilon, MinPts)$ 的一个簇$\subseteq D$，使得 $^* \subseteq$ 。

OPTICS 试图通过建立 D 中对象的排列来利用命题3.9，使稠密区域被稀疏簇包围。由此，该算法使用可达距离作为密度度量，其中高密度对应较低的可达距离。

算法. OPTICS 可以通过如下嵌套循环实现：在外层循环中，从 D 中任意选取一个对象 o，并将 o 追加到初始为空的簇排序 O中。如果 o 相对于 $(\varepsilon, MinPts)$ 是核心对象，则将 o 的邻居插入优先队列 Q，并切换到内层循环，依次从 Q 中弹出对象，直到队列再次为空。Q 中对象的优先级最初由其相对于 o的可达距离决定。当前可达距离最小的对象 p 总是最先弹出 fi（优先级最高），并追加到排序 O中。如果 p 相对于 $(\varepsilon, MinPts)$ 是核心对象，则将 p 的邻居添加到 Q 中，并计算它们相对于 p 的可达距离。如果某个邻居已包含在 Q中，则将其当前可达距离与相对于 p的新可达距离进行比较。如果相对于 p 的新可达距离小于当前值，则相应地更新该邻居的优先级。

OPTICS 对每个对象恰好处理一次（即，已添加到 O 的对象不会再次插入 Q）。每当一个对象被添加到 O 时，也会记录其核心距离，以及将其从 Q 中弹出时的可达距离。在外层循环中处理的对象，其可达距离设为 ∞。因此，每个已处理对象 $x \in O$ 所记录的可达距离，都是 x 在此前被添加到 O 的任一对象密度可达时的最小距离阈值。

聚类。 算法终止时，命题 3.9 体现在排序 O中：数据集 D 的稠密区域被其可达的稀疏区域所包围。因此，对于给定的 $\varepsilon^* \leq \varepsilon$ 和固定的$MinPts$，可以从 O查询任意聚类。使用所谓的可达性图最能说明查询过程：该图从左到右按处理顺序绘制各对象的可达距离。图 3 展示了图 1 中数据集的这一情况。通过在 ε^* 处截取图的纵轴，并将每个谷地 中的对象分配到单独的簇中，即可提取聚类。此外，对于每个谷地，还会将一个初始核心对象 o添加到相应的簇中；该对象位于谷地正前方，并通过$C_{\varepsilon,MinPts}(o) \leq \varepsilon^*$进行识别。

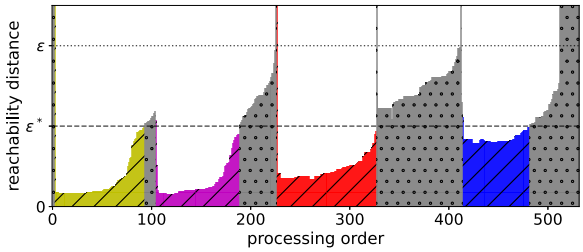


图 3. 对图 1 中的数据集以 $\varepsilon^* = \varepsilon/2$ 为参数进行 OPTICS 聚类所得的可达性图。

3.3 局限性

DBSCAN 和 OPTICS 都 s uffer 存在严重局限性，具体如下。 本节。

效*ffi*率。 要为给定数据集进行聚类，DBSCAN 必须计算数据集中所有对象的 ε邻域。随后，它会评估这些对象的核心状态，并确定检测到的核心对象之间的连通分量。一次邻域计算最多需要进行 $O(n)$ 次距离计算（其中 n 表示数据集大小）。因此，DBSCAN 的复杂度为 $O(n^2)$，这使得在处理大型数据集时无法进行交互式聚类。

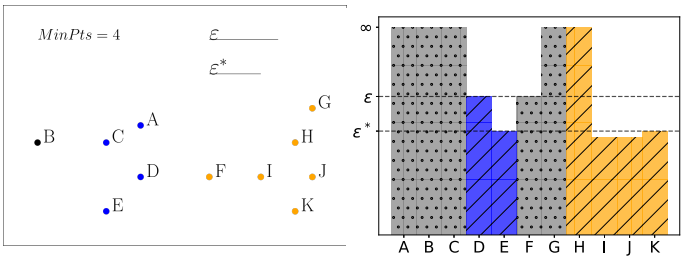
表1.相对于 ε 和 $MinPts = 4$ 的核心对象，来自图4（相对于 ε的欧几里得距离）。

对象	核心距离	排序后的 ε-邻域
C	1	(A, $\sqrt{5}/4$), (D, $1/\sqrt{2}$), (B, 1), (E, 1)
D	$3/4$	(C, $1/\sqrt{2}$), (E, $1/\sqrt{2}$), (A, $3/4$), (F, 1)
H	$1/\sqrt{2}$	(G, $\sqrt{5}/4$), (J, $\sqrt{5}/4$), (I, $1/\sqrt{2}$), (K, 1)
I	$3/4$	(H, $1/\sqrt{2}$), (K, $1/\sqrt{2}$), (F, $3/4$), (J, $3/4$)
J	$3/4$	(H, $\sqrt{5}/4$), (K, $\sqrt{5}/4$), (I, $3/4$), (G, 1)
K	1	(J, $\sqrt{5}/4$), (I, $1/\sqrt{2}$), (H, 1)

参数调优与近似。 尽管可以在 $O(n)$ 时间内查询 OPTICS 排序以获得聚类结果，但我们发现了两个主要问题：(i) 无法更改 $MinPts$ 参数，这限制了从 OPTICS 排序中可探索的聚类方案。(ii) 根据 $\varepsilon^* \leq \varepsilon$查询 OPTICS 排序所得的聚类仅是 DBSCAN 生成的精确聚类的近似结果（参见定义 3.5）。我们在示例3.10中展示了问题 (ii) 的严重性。

示例*3.10*。 令 $\varepsilon^* = 3\varepsilon/4$ 和 $MinPts = 4$。图4展示了一个小型二维数据集，其中的对象根据相对于 (ε^*, $MinPts$) 的精确聚类着色。簇为K_1 = {A, C, D, E}（蓝色）和K_2 = {F, G, H, I, J, K}（黄色），噪声组为 N = {B}（黑色）。图4b展示了相对于 $(\varepsilon, MinPts)$ 计算得出的 OPTICS 排序的可达性图，柱条根据从 OPTICS 排序中查询得到的聚类着色。**我们观察到：** OPTICS 的簇明显小于精确聚类，K_1 漏掉了其中 50% 的对象，而K_2 则漏掉了三分之一的对象。所有缺失对象均标记为噪声。

为便于复现示例3.10，我们在表1中列出了本示例中核心对象的 ε-邻域。因此，感兴趣的读者可以按照第3.2节中的描述或Ankerst等人 [2]中的算法，重构OPTICS 排序及其聚类。值得一提的是，示例3.10并未展示最坏情况。事实上，基于 OPTICS 的聚类可能会遗漏每一个边界对象，而本例并非如此。



(a) 真实聚类。 (b) OPTICS近似。

图4. 关于 $\varepsilon^* = 3\varepsilon/4$ 和 $MinPts = 4$ 的聚类。

4 OPTICS 的形式化

我们以OPTICS为起点开发一种新的索引 FINEX，以克服第3.3节中概述的局限。因此，我们深入研究并形式化OPTICS 排序，并讨论被 OPTICS 错误归类fi为噪声的对象的性质。

算法 1： 查询聚类(O, ε^*) [2]

输入： 簇排序 O，距离阈值 ε^*
输出： 簇集合和噪声组

```
1  Noise ←;
2  Clustering ← ;
3  S ←;                                                    // S 收集簇对象
4  对于每个 x ∈ O 执行
5  |  如果 xR > ε* 则
6  |  |  如果 xC ≤ ε* 则
7  |  |  |  如果 S ≠ ∅ 则
8  |  |  |  |  Clustering ← Clustering ∪{ S}
9  |  |  |  |  S ← ∅                                      // 开始新簇
10 |  |  |  S ← S ∪{ x}
11 |  |  else
12 |  |  |  Noise ← Noise ∪{ x}
13 |  else
14 |  |  S ← S ∪{ x}
15 如果 S ≠ ; 则
16 |  Clustering ← Clustering ∪{ S}
17 Clustering ← Clustering ∪{ Noise}
18 返回 Clustering
```

簇排序。 根据第3.2节，我们知道 OPTICS 会计算输入数据集 D的一个排列，并为其中每个对象 $x \in D$附加核心距离和可达距离。我们在定义 4.1 中给出一种精确的形式化定义，其灵感来自 Achtert 等人 [1]。

定义*fi4.1*（*OPTICS* 排序）。设 D 为一个数据集，min ; $= \infty$。一个*OPTICS* 排序$O_{\varepsilon,\ MinPts}$ 是 D 的一个排列，其中每个对象 $x \in D$ 都带有以下属性：

xP：$\{1, \ldots, |D|\}$中的置换编号xC：核心距离，$C_{\varepsilon,}$
$MinPts(x)xR$：可达距离，$\min_{p \in D,\ p.P < xP}\{R_{\varepsilon,}$
$MinPts(x,\ p)\}$

请注意，定义 4.1 使用属性 xP 来实现排列，这将简化我们后续的理论讨论。然而，在实践中，可以通过按照处理顺序排列对象来避免使用这一属性。OPTICS 算法 [2] 根据定义 4.1 计算出一个排序，以报告关于 $\varepsilon^* \leq \varepsilon$ 和 $MinPts$的近似聚类。

簇提取. 我们研究从 OPTICS 排序中查询得到的簇，因此先简要介绍 Ankerst等人提出的原始 OPTICS 查询技术，并采用符合本文其余部分记号的版本。其伪代码见算法 1。该查询技术基本上按处理顺序遍历所有对象$x \in O_{\varepsilon,MinPts}$，并利用它们的核心对象距离和可达距离属性（$xC$ 和 xR），分别将其分配给当前簇 S 或噪声组。请注意，一旦发现一个相对于（ε^*,$MinPts$）的核心对象x，便会开始一个新簇 $xR > \varepsilon^*$（参见第 6 行）。随后扩展该簇，直到遇到下一个可达距离大于 ε^* 的对象（参见第 5 行）。直观地说，算法 1 可以想象为

在可达性图的 ε^* 处切开纵轴，并将一个初始核心对象添加到由此形成的谷中（参见图 3）。

簇的性质. 接下来，我们列出定义 4.2中算法1 的簇集合 S 中的元素的精确性质。请注意，我们将 S称为近似簇，因为正如我们随后将展示的那样， S仅近似于一个基于密度的簇。

定义*fi4.2*（近似簇）。我们将子集 $S = \{x_1, \ldots, x_n\} \surd \quad D$称为相对于 $\varepsilon^* \leq \varepsilon$和 $MinPts$ 的近似簇，如果其元素满足以下条件:

（1） $x_{i+1}.P = x_i.P + 1$ 对于所有 $i = 1, \ldots, n$ 1（2） $x_1.C \leq \varepsilon^*$ 且 $x_1.R > \varepsilon^*$（3） $x_i.R \leq \varepsilon^*$ 对于所有 $i = 2, \ldots, n$（4）要么 $x_n.P = |D|$，要么对于满足 $y.P = x_n.P + 1$ 的 $y \in D$，有 $y.R > \varepsilon^*$

定义 4.2 中列出的性质可表述如下：(1) S 由在$O_{\varepsilon,\ MinPts}$中彼此相邻的对象组成。(2) x_1 是关于（ε^*，$MinPts$）的核心对象，并且 x_1 相对于（ε^*，$MinPts$）不从任何之前已处理的其他对象密度可达。(3) 每个对象 x_i、 $i = 2$，……，n都相对于（ε^*，$MinPts$）从某些置换编号更低的其他对象密度可达。(4) x_n 要么是$O_{\varepsilon,\ MinPts}$中的最后一个对象，要么紧接着x_n处理的对象 y不相对于（ε^*，$MinPts$）从具有更低置换编号的另一个对象密度可达。

缺失对象。 OPTICS 仅处理每个对象 $x \in D$ 一次，因此无法全局最小化可达距离属性 xR，算法 1 使用该属性来确定 x的簇成员资格。例如，如果 x 是关于（ε，$MinPts$）的边界对象，并且在 OPTICS 外层循环中被处理，那么 $xR = \infty$会被分配（参见第 3 节）。因此，即使 $\varepsilon^* = \varepsilon$， x 也将始终 被算法 1 错误地标记为噪声。在定理 4.3 中，我们准确地说明了由 OPTICS 计算得到的近似簇 $S \surd O_{\varepsilon,\ MinPts}$ 与根据定义 3.4 得到的基于密度的簇之间的关系。证明见附录 A。

Theorem 4.3 (Approximate OPTICS Clusters). 设$O_{\varepsilon,MinPts}$ 为由 *OPTICS* 计算得到的簇排序。根据定义 *4.2*，我们考虑关于 (ε^*,$MinPts$) 的近似簇 S。以下命题成立:

(a) ∃ 根据定义 *3.4*，相对于 ε^* 和 $MinPts$ 的一个基于密度的簇，使得 $S \subseteq$.(b) $\forall p \in K$: $p.P \leq x_n.P$(c) $\forall p \in K: |N_{\varepsilon^*}(p)| \geq MinPts \implies p \in S$

定理 4.3 可以被理解为一条将 OPTICS 的近似聚类转换为精确聚类的指令：它指出，每个近似簇 S 都是某个基于密度的簇 K 的子集。此外，$K \setminus S$ 中的所有对象都在 $x_1 \in S$ 之前已处理。最后， S 包含 K 的所有核心对象，因此，$K \setminus S$ 仅由边界对象组成。换言之，缺失边界对象的搜索空间可能由所有在 x_1 之前已处理、且不属于任何其他近似簇的对象组成（因此被标记为噪声）。根据定义 3.5，将 OPTICS 的近似聚类转换为精确聚类的一种朴素方法，是检查搜索空间中的每个对象是否实际上位于某个核心对象的 ε^*-邻域内。然而，取决于噪声的数量，这在计算上可能代价高昂。因此，我们自己的索引的设计理念是尽可能限制搜索空间，从而减少查询时的工作量。

5 使用 FINEX 进行聚类

在本节中，我们介绍一种名为FINEX 的新型线性空间索引，它能够针对变化的阈值参数快速提取精确的基于密度的聚类，即 $\varepsilon^* \leq \varepsilon$ 或 $MinPts^* \quad MinPts$。我们介绍针对给定对（$\varepsilon, MinPts$）的 FINEX 构造，并讨论如何使用 FINEX提取比OPTICS结果更准确的近似聚类。我们还展示了，当$\varepsilon^* = \varepsilon$ 且 $MinPts^* = MinPts$ 时，FINEX 会返回精确聚类（不同于OPTICS）。最后，我们讨论如何使用FINEX 高ffi效地推导分别关于 $\varepsilon^* < \varepsilon$ 或 $MinPts^* > MinPts$ 的精确聚类。本节中所有理论的证明均可见附录 A。

5.1 构建 FINEX

本节总结 FINEX 索引结构，以及如何为给定数据集 D 和生成对（ε， $MinPts$）构建该结构。

索引结构。 提取精确的基于密度的聚类是我们的最终目标，这也反fl映在 FINEX 的结构和接口上。从本质上讲，FINEX 是输入数据集 D 的一个排列，并且每个对象 $x \in D$ 都扩展为一个五元组 $(P,\ C,\ R,\ N,\ F)$，其中 $x.P$和 $x.N$是整数， $x.C$和 $x.R$是 fl浮点数，而 $x.F$是一个引用（参见定义 5.1）。因此，FINEX 所需空间与数据集大小 $|D|$成线性关系。

我们的索引接受两个参数：一个数据集 D和一个生成对（$\varepsilon, MinPts$），并返回一个 ***FINEX* 排序** 作为输出。FINEX 排序增强了 OPTICS 排序，并按如下方式定fi义。

定义*fi5.1*（*FINEX* 排序）。令 D 为一个数据集。*FINEX* 排序 $\mathrm{e}^{O_{\varepsilon,\ MinPts}}$ 是 D 的一个排列，其中每个对象 $x \in D$ 都具有以下属性：

$x.P$：置换编号， $P \in \{1, \ldots, |D|\}$ $x.C$：相对于 $(\varepsilon, MinPts)$ 的核心距离， $C_{\varepsilon,MinPts}(x)$ $x.R$：
$$\begin{cases} : \min_{p \in D}\{R_{\varepsilon,MinPts}(x,p)\}, & \text{若 } x \text{ 相对于 } (\varepsilon, MinPts) \text{ 为非核心对象} \\ \min_{p \in D, p.P} \begin{cases} x.P\{R_{\varepsilon,MinPts}(x,p)\}, & \text{否则} \\ :x, & \text{若 } x \text{ 相对于 } (\varepsilon, MinPts) \text{ 为噪声} \end{cases} & y \in N_\varepsilon(x), \text{取最大 } |N_\varepsilon(y)|, \text{否则} \end{cases}$$
$x.N$：相对于 ε, $|N_\varepsilon(x)$ 的邻域大小 $|x.F$：

最初，我们设置 $\min$； $= \infty$，且对象 $o \in D$ 相对于 $(\varepsilon, MinPts)$ 是非核心对象，当且仅当 $|N_\varepsilon(o)| < MinPts$。重要的是，FINEX 对相对于 $(\varepsilon, MinPts)$ 的非核心对象采用不同的处理方式，即，其可达距离 $x.R$ 在全局所有 $p \in D$上取最小值。这使我们能够解决如何正确标记所有簇对象这一挑战。此外，每个对象 $x \in D$还具有两个额外属性，用于实现精确的 $MinPts^*$-查询（参见第 5.4 节）：(1) $x.N$ 表示邻域大小，即邻居的数量。(2) $x.F$ 是 *finder reference*，即指向具有最多邻居且相对于 $(\varepsilon, MinPts)$ 可达 x 的（核心）对象的引用。

索引构建。 算法 2 和算法 3 展示了我们索引的构建过程。算法 2 的整体结构让人联想到 OPTICS 算法 [2]。不过，更新优先队列 Q 的方式有很大不同ff——参见算法 3。在算法 2 的外层循环中，任意选取一个尚未处理的对象 $o \in D$，并计算其核心距离 $o.C$ 和邻域大小 $o.N$。由于 o 尚未从任何其他对象到达，我们将 $o.R$ 初始化为 ∞

并使 finder 引用 $o.F$ 指向自身（隐式地）。之后，我们将 o 追加到 FINEX 排序 eO中。如果 o 是相对于生成对 $(\varepsilon, MinPts)$ 的核心对象（第 9 行），则将 $N_\varepsilon(o)$ 的所有对象插入优先队列 Q，并执行内层循环（第 11–17 行）。

在每次迭代中，内层循环会弹出可达距离最小（优先级最高）的对象 $p \in Q$，计算 $p.C$ 和 $p.N$，并将 p 追加到 eO。注意，$p.R$ 和 $p.F$ 已经计算完成，同时优先队列 Q 也已更新（参见算法 3）。如果 p 也是核心对象，则将 N_ε（p）中的所有对象传递给算法 3（第 16–17 行）。

算法 3 展示了优先队列 Q的更新过程。它会遍历某个核心对象 2 的所有对象 $q \in N_\varepsilon$(2)（记为 $neighbors_c$）。如果 $o \in$ eO，则将对象 o 标记为已处理；如果 $o \notin$ eO，则 o 尚未处理。与 OPTICS 不同，我们可以将非核心对象从 FINEX 排序中移除，并重新插入 Q，以便在全局范围内最小化它们的可达距离。因此，算法 3 针对 q区分了三种情况：(1) q 尚未处理且 $q \notin Q$（第 3 行）：将可达属性 $q.R$ 设为$R_{\varepsilon,MinPts}$（q, 2），然后将 q 插入队列。注意，$q.R$ 表示 q 的优先级（可达距离越低，优先级越高）。(2) $q \in Q$（因此，q 仍未处理，第 7 行）：如果$R_{\varepsilon,MinPts}$（q, 2） $< q.R$，则更新 $q.R$，并相应提高 q的优先级。(3) q 已处理（因此，$q \notin Q$，第 11 行）：我们检查 (a) q 相对于（$\varepsilon, MinPts$）是否为非核心对象，以及 (b) $R_{\varepsilon,MinPts}$（q, 2） $< q.R$（第 12 行）。在这种情况下，相应地更新 $q.R$，并将 q 从 FINEX 排序 eO中移除、再次标记为尚未处理，然后重新插入队列 Q。实际上，情况 (3) 在全局范围内最小化了非核心对象的可达距离 $q.R$（参见定义 5.1）。注意，这不会改变渐近时间复杂度，因为每个非核心对象重新插入 Q的次数至多为（$MinPts$\u0000 1）次。第 16–17 行用于执行精确的 $MinPts^*$查询，相关讨论见第 5.4 节。对于所有 $o \in D$，将 $o.F$初始化为自引用，并将 $o.N$初始化为 0（参见算法 2），可确保这些行在所有情况下都有明确定义。

5.2 线性时间聚类

OPTICS 的一个主要缺点是，边界对象可能会被错误地标记为噪声。本节将介绍如何缓解这一问题，并重点说明 FINEX 相较于 OPTICS 的两个优势：(i) 在线性时间内提取更准确的聚类。(ii) 对于至少一种参数设置，即 $\varepsilon^* = \varepsilon$，FINEX 可以在线性时间内直接提取精确的聚类。这两点都是我们构建索引（即 FINEX 排序）方式的直接结果，并且不会在运行时引入开销。

在线性时间内实现更准确的聚类。 构建 FINEX后，我们可以应用OPTICS 的簇提取技术（参见算法 1）来提取更准确的基于密度的聚类（与 OPTICS 提取的聚类相比）。值得注意的是，仅将OPTICS排序替换为 FINEX 排序，就已经能够显著改善近似聚类。例如，图 5 分别展示了使用 OPTICS排序（图 5a）和 FINEX排序（图5b）得到的聚类结果。非核心对象的可达距离的全局最小化体现在图5b中 A、B、F 和 G 对应的更短柱状条上。结合图 5b x 轴上的新排列，FINEX 正确识别出黄色簇的所有 对象，以及蓝色簇四个对象中的三个对象。与使用 OPTICS排序提取的聚类相比，这是一个显著的改进；后者分别只能识别出蓝色簇和黄色簇的 50% 和 67%。

边界对象的聚类。 现在，我们通过描述 FINEX 的近似簇以及 FINEX 如何改进 OPTICS 提供的近似，正式阐述图 5 中的观察结果。

算法 2： FINEX 构建(D, ε, $MinPts$)

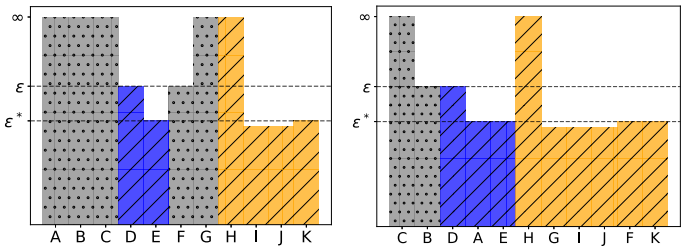
输入： 数据集 D，距离 ε，计数 $MinPts$
输出： FINEX 排序 eO 相对于 (ε，$MinPts$)
// $o.F$ 被初始化为自引用 $\forall o \in D$
// $o.N$ 已初始化为 0 $\forall o \in D$

```
1  Õ ← 新建空的簇排序
2  Q ← 新建空的优先队列
3  当 |Õ| < |D| 执行
4      o ← 中的任意未处理对象 D
5      o.C ← C_{ε,MinPts}(o)
6      o.N ← |N_ε(o)|
7      o.R ← ∞
8      将 mark o 标记为已处理，并将其追加到 Õ
9      如果 o.C ≤ ε 那么
10         Q.update(o, N_ε(o), eO)
11         当 Q ≠ ∅ 执行
12             p ← Q.pop()
13             p.C ← C_{ε, MinPts}(p)
14             p.N ← |N_ε(p)|
15             将 p 标记为已处理并将其追加到 Õ
16             如果 p.C ≤ ε，则
17                 Q.update(p, N_ε(p), eO)
18 返回 Õ
```

算法 3： 优先队列::update(c, $neighbors_c$, $\widetilde{O}$)

输入： 核心对象 c，c $neighbors_c$ 的邻居，FINEX 排序 $\widetilde{O}$
// 注意 $c \in neighbors_c$

```
1  对每个 q ∈ neighbors_c 执行
2      rdist ← R_{ε,MinPts}(q,c)
3      如果 q 尚未处理且不在优先队列中则
4          q.R ← rdist
5          insert(q)
6      else
7          如果 q 在优先队列中 则
8              如果 rdist < q.R 则
9                  q.R ← rdist
10                 increasePriority(q)
11         else
12             如果 q.C > ε 且 rdist < q.R 则
13                 将 q 标记为未处理，并从 q 中移除 Õ
14                 q.R ← rdist
15                 insert（q）
16     如果 c.N > q.F.N 那么
17         q.F ← c
```



(a) OPTICS 排序。 (b) FINEX 排序。

图5. 对于图4a中的数据集，算法 1 所报告的关于 $\varepsilon^* = 3\varepsilon/4$ 和 $MinPts = 4$ 的聚类。

THEOREM 5.2 (APPROXIMATE FINEX CLUSTERS). 令$\tilde{}O_{\varepsilon,MinPts}$ 为通过算法 *2* 计算得到的 *FINEX* 排序。根据定义 *4.2*，考虑关于$(\varepsilon^*,MinPts)$ 的近似簇 S，以下命题成立：

(a) 存在一个关于 $(\varepsilon^*,MinPts)$ 的簇K，使得 $S\surd$

.(b) $\forall p \in K : |N_\varepsilon(p)| \quad MinPts \;=)\quad pP \leq x_n.P.$

(c) $\forall p \in K : |N_{\varepsilon^*}(p)| \quad MinPts \;=)\quad p \in S.$

直观来说， S 是关于 $(\varepsilon^*,MinPts)$ 的某个基于密度的簇的子集，该簇已经包含了 的所有核心对象。根据 (b)，关于 $(\varepsilon,MinPts)$ 的核心对象，可能会在关于 $(\varepsilon^*,MinPts)$ 的边界对象之前被处理。算法 2 全局地 最小化关于 $(\varepsilon,MinPts)$ 的非核心对象的可达距离（与 OPTICS 不同）这一事实，体现了定理 5.2 的强大之处，并导出以下定理：

定理HEOREM 5.3 (CLUSTERING OF NON-CORES). 设 $x \in \tilde{}O_{\varepsilon,MinPts}$ 是相对于 $(\varepsilon, MinPts)$的非核心对象。给定距离~阈值 $\varepsilon^* \leq \varepsilon$，将算法 *1* 应用于$O_{\varepsilon,MinPts}$ 会正确地将 x 分配给

- 噪声组，如果 x 是相对于$(\varepsilon^*,MinPts)$的噪声对象，或者
- 对应的簇，如果 x 是相对于$(\varepsilon^*,MinPts)$的边界对象。

定理 5.3 基于对可达距离的全局最小化，并保证相对于 $(\varepsilon^*,MinPts)$ 属于非核心对象、但相对于 $(\varepsilon,MinPts)$ 属于边界对象的对象不会 被错误地标记为噪声。OPTICS （参见第 4 节）则无法做到这一点，这也凸显了 FINEX 排序的优势。

另一类对象是所谓的前核心，即相对于 $(\varepsilon,MinPts)$ 属于核心对象、但相对于 $(\varepsilon^*,MinPts)$ 变为非核心对象的对象。形式上，如果 $|N_{\varepsilon^*}(o)| < MinPts$ 且 $|N_\varepsilon(o)|MinPts$，则对象 $o \in D$ 相对于 $\varepsilon^* \leq \varepsilon$ 和 $MinPts$ 是一个前核心 。定理 5.4 形式化地说明，在各自的近似结果中，OPTICS 和 FINEX 都会将相同的前核心正确地归类为边界对象。

THEOREM 5.4 (CLUSTERING OF FORMER-CORES). 设 $x \in D$ 是相对于 $\varepsilon^* \leq \varepsilon$ 和 $MinPts$ 的曾为核心对象，且$\tilde{}O_{\varepsilon,MinPts}$和$O_{\varepsilon,MinPts}$ 分别为 *FINEX* 排序和 *OPTICS* 排序。对于 ε^*，算法 *1* 使用$\tilde{}O_{\varepsilon,MinPts}$ 时会将*fi*识别x 为簇成员，而算法 *1* 使用$O_{\varepsilon,MinPts}$ 时会将*fi*识别为簇成员。

请注意，定理5.4 要求相对于插入顺序的稳定优先队列，即具有相同优先级的并列元素按照插入顺序弹出。因此，在算法 2 中相对于 $(\varepsilon,MinPts)$ 重新处理非核心对象，既不会影响其他对象的可达距离，也不会影响前核心对象的总体顺序（即，它们的顺序在e$O_{\varepsilon,MinPts}$和 $O_{\varepsilon,MinPts}$中相同）。结合定理5.2、定理 5.3 和定理5.4 可知，FINEX 的近似聚类至少与 OPTICS 的近似聚类一样准确。

线性时间内的精确聚类。 FINEX 排序使我们的解决方案能够在特殊情况下 $\varepsilon^* = \varepsilon$，在线性时间内返回关于 $(\varepsilon^*, MinPts)$ 的精确聚类。这直接由定理 5.2 和定理 5.3 得出，即核心对象和非核心对象均被正确标记，并突显了 FINEX 相较于 OPTICS 的另一项优势。我们以相应的推论作结：

Corollary 5.5 (Linear Exact Query). 令 $\tilde{O}_{\varepsilon,MinPts}$ 是通过算法 *2* 计算得到的*FINEX* 排序。带有 $\varepsilon^* = \varepsilon$ 的算法 *1* 根据定义*3.5*，关于$(\varepsilon, MinPts)$返回一个精确聚类，其时间复杂度为线性时间（以对象数量计）。

5.3 精确 ε^*-Qu序列

在第 5.2 节中，我们讨论了FINEX 会为 $\varepsilon^* = \varepsilon$返回精确聚类。然而，当 $\varepsilon^* < \varepsilon$时，一些曾为核心对象的对象仍可能被错误地标记为噪声（参见图 5b 中的对象 C）。在本节中，我们引入一种名为 ε^*-查询 的新查询类型，该类型对精确查询进行了泛化，使 FINEX能够为任何 $\varepsilon^* \leq \varepsilon$返回精确聚类。

ε*-查询算法。 给定我们的索引和 $\varepsilon^* \leq \varepsilon$，我们分两步高效地推导出关于（$\varepsilon^*$,$MinPts$）的精确聚类：（1）首先，使用算法 1 获取关于（ε^*,$MinPts$）的近似簇 $S_1, \ldots, S_m$。（2）对于每个S_i,，我们会针对关于 $\varepsilon^* \leq \varepsilon$的曾为核心对象以及作为 S_i边界对象的 $MinPts$执行定向搜索。通过检查 $\varepsilon^* < o.C \leq \varepsilon$，可以高效地识别曾为核心对象$o \in e^{O}_{\varepsilon,MinPts}$。当 o 被标记为噪声，且在 S_i 中的第一个对象之前已处理时，曾为核心对象 o 会被视为 S_i 的候选对象（参见定理 5.2）。随后，通过检查 o 的 ε^*-邻域是否包含关于（ε^*,$MinPts$）的 S_i 核心对象，验证这些候选对象。

命题 3.9 进一步限制了候选对象，即，只有某个稀疏聚类的前核心对象 $K, S_i \surd K,$ 才会成为候选对象，因为 K 核心对象围绕着 S_i 在 $e^{O}_{\varepsilon,MinPts}$中的对象。以下定理总结了如何为 $\varepsilon^* \leq \varepsilon$ 和 $MinPts$推导出精确聚类。

定理heorem 5.6（推xact ε^*-Query论）。设$\tilde{O}_{\varepsilon,\ MinPts}$ 是由算法 *2* 计算得到的 *FINEX* 排序。算法 *1* 使用$\varepsilon^* \leq \varepsilon$返回相对于（$\varepsilon^*$,$MinPts$）的近似簇 $S_1, \ldots, S_m$。对于每个 S_i,，如果 o相对于（ε^*,$MinPts$）可从 S_i 中的某个核心对象直接密度可达，且满足以下条件，则我们将 $o \in \tilde{O}_{\varepsilon,\ MinPts}$添加到 S_i：

(1) o 被标记为噪声，并且 $\varepsilon^* < o.C \leq \varepsilon$(2) o 在 S_i
(3)∃ 中相对于$(\varepsilon, MinPts)$ 的第一个对象之前已处理，该对象属于一个簇，且满足 $S_i \cup \{o\} \subseteq$(4) o 尚未被加入其他 $S_j, j \neq i$

根据定义 *3.5*，结果是相对于$(\varepsilon^*, MinPts)$ 的精确聚类。

讨论. 定理 5.6 中的条件形式化地描述了候选集合所受的约束。条件 (1)– (3) 可以按照前文所述推导得出。条件 (4) 是必需的，以便将有歧义的边界对象分配给且仅分配给一个簇。

请注意，这些对候选对象的严格限制仅适用于我们的 FINEX 排序，而不 适用于 OPTICS。显然，基于 FINEX 的 ε^*查询所需的邻域计算次数少于从头开始运行 DBSCAN 所需的计算次数。此外，针对候选对象 o 的一次邻域计算效率更高，原因有二：(i) 在我们的场景中，邻域计算只需考虑 S_i 的核心对象，而非 D中的所有对象。(ii) 一旦在 o的 ε^*邻域中找到一个核心对象，我们就可以终止计算，无需计算到其余核心对象的精确距离。总体而言，与从头开始运行 DBSCAN 相比，ε^*查询所需的邻域计算次数更少，而且通常效率更高。

5.4 精确 $MinPts^*$-Query

到目前为止，我们假设 $MinPts$ 为常数（类似于 OPTICS）。然而，允许修改 $MinPts$ 会增加可探索聚类的数量。因此，我们引入 $MinPts^*$-查询，这是一种新的查询类型，对于常数 ε 和任意 $MinPts^*$ $MinPts$，均可产生精确聚类。请回顾，OPTICS 完全不支持 $MinPts^*$-查询，因此我们剩下的主要竞争对手是 DBSCAN。FINEX 的技巧再次在于利用基于密度的聚类的一个基本性质，即稠密簇包含于稀疏簇中。类似于关于 ε-嵌套簇的命题 3.9，下面的命题形式化地定fi义 $MinPts$-嵌套簇。

Proposition 5.7 ($MinPts$-Nested Clusters)。令 $MinPts^* \geq MinPts$。对于每个簇 $^* \subseteq D$，相对于 $(\varepsilon, MinPts^*)$，存在一个簇 $\subseteq D$，相对于 $(\varepsilon, MinPts)$，使得 $^* \subseteq$ 。

一个关键观察是，我们可以使用精确（稀疏）聚类数据集 D的一个子集上计算聚类。不过，FINEX 还提供了更多优化选项。

MinPts*-查询算法。 给定 $MinPts^*$ $MinPts$，FINEX 能够高效地针对 $(\varepsilon, MinPts^*)$ 得出精确聚类。我们的 $MinPts^*$-查询分为三个步骤：(1) 根据 $\mathrm{e}O_{\varepsilon,MinPts}$，我们应用算法 1，得到针对 $(\varepsilon, MinPts)$ 的精确聚类，其中包含 l个稀疏簇 $E_1, \ldots, E_l$，并丢弃 D中不属于任何E_i, $1 \leq i \leq l$的所有噪声对象。(2) 对每个 E_i, $1 \leq i \leq l$，我们确定针对 $(\varepsilon, MinPts^*)$ 的核心对象密度连通集，即在 E_i 中为 $MinPts^*$ 找出所有稠密簇。(3) 针对 $(\varepsilon, MinPts^*)$ 缺失的边界对象将被识别，并利用相应的 finder 引用添加到各自的稠密簇中。

在步骤 (1) 中，我们提取关于 $(\varepsilon, MinPts)$ 的精确聚类，暂不考虑 $MinPts^*$ $MinPts$。因此，我们只根据 $(\varepsilon, MinPts)$ fi滤除噪声对象。在步骤 (2) 中，我们在稀疏簇 $E_1, \ldots, E_l$ 中识别密度更高的簇，即 $MinPts^*$ $MinPts$。换言之，稀疏簇可能会关于 $MinPts^*$ 分解为多个稠密簇。步骤 (2) 仅涉及关于 ε 和 $MinPts^*$ 的核心对象，这些对象通过邻域大小属性识别。相应的伪代码如算法 4 所示，第 6 行和第 12 行执行（部分）邻域计算。步骤 (3) 将边界对象正确地分配给步骤 (2) 得到的稠密簇。最终，这将得到关于 $(\varepsilon, MinPts^*)$ 的精确聚类。

讨论. 请注意，精确聚类会将含糊的边界对象恰好分配给一个簇。因此，对于相对于 $(\varepsilon, MinPts)$ 的精确簇 E_i，存在某个基于密度的簇 K_i，使得 $E_i \surd K_i$。因此，在子集 E_i 上对 $(\varepsilon, MinPts^*)$ 应用标准 DBSCAN 可能会产生错误结果，因为相对于 $(\varepsilon, MinPts^*)$ 的核心对象可能会由于 E_i 中缺少含糊的边界对象而失去其核心状态。我们通过在索引构建期间为所有 $o \in \mathrm{e}O_{\varepsilon,MinPts}$ 预先计算$o.N$来解决这一问题（参见算法 2）。$o.N$表示 $N_\varepsilon(o) \surd D$ 的大小，因此，如果 $o.N$\u0000 $MinPts^*$，核心对象 $o \in E_i$即可被正确识别fi。这还使 FINEX 能够在 $o.N < MinPts^*$时完全避免邻域计算，从而提高 $MinPts^*$-查询的效ffi率。

我们的 $MinPts^*$查询还受益于这样一个事实：我们只针对关于 $(\varepsilon, MinPts^*)$ 的核心对象集合执行邻域计算，以检测 E_i 中关于 $(\varepsilon, MinPts^*)$ 的密度相连簇子集。随后，根据 fi指向引用 $o.F$，将所有边界对象 o添加到这些子集中，无需进行任何邻域计算。在索引构建期间，每当处理到一个邻居更多的核心对象时，都会更新 fi指向引用 $o.F$（参见算法 3 第 17 行）。算法 2 终止后，$o.F$指向 o 的最密邻居。

算法 4： ComputeCoreClustering($Cores$)

输入： 相对于 $(\varepsilon, MinPts^*)$ 的核心对象 $Cores$

输出： 已聚类核心对象的集合

```
S ←;                                                    // S 收集簇对象
Stack ←;
while Cores ≠ ∅ do
    x ← Cores.pop()
    S ← S ∪{ x}
    neighbors_x ← N_ε(x)\ Cores
    Cores ← Cores \ neighbors_x
    Stack.push(neighbors_x)
    while Stack ≠ ; do
        y ← Stack.pop()
        S ← S ∪{ y}
        neighbors_y ← N_ε(y)\ Cores
        Cores ← Cores \ neighbors_y
        Stack.push(neighbors_y)
    Clustering ← Clustering ∪{ S}
    S ← ∅
return Clustering
```

6 实证评估

本节在来自不同领域的 12 个大型真实世界数据集上评估 FINEX，并将其聚类运行时间与从头开始计算聚类的精确基线 DBSCAN 和 AnyDBC [17]进行比较。此外，当使用算法 1 以线性运行时间查询两个索引时，我们还从构建时间和聚类准确率方面，将 FINEX 与最先进的索引技术 OPTICS 进行比较。我们的实证评估涵盖两种不同的数据类型和距离函数：采用*Jaccard* 距离 的集合数据，以及采用欧几里得距离的多维向量数据。

数据集和距离函数。我们的研究受到过程挖掘领域的启发，在该领域，业务流程事件日志中的活动被建模为由唯一令牌组成的集合 [12, 15]。例如，事件序列 (Start,Order, Pay,Return,reFund, End)被建模为集合 $\{(S,O), (O,P), (P,R), (R,F), (F,E)\}$，其中集合中的元组 $(X,.)$表示从事件 X 到事件Y的转换 [15]。出于效fi率方面的考虑，这些转换被双射地映射为整数令牌。我们将这种建模方法称为集合数据。对于集合的聚类，我们使用 Jaccard 距离 d_J，它在相似性搜索中广为人知 [3, 18, 25]。给定两个集合 r、$s \in D$，Jaccard 距离被定fi义为

$$d_J(r,s) = 1 - \frac{|r \cap s|}{|r \cup s|}$$

直观地说，如果 r和 s相似，那么相对于各自的大小，它们共享许多词元，因此 d_J会很小。由于集合数据与 Jaccard 距离相结合，也是流程挖掘公司特别感兴趣的处理对象，因此我们在实验中使用了该领域的四个真实世界数据集：CELONIS-1/2 和 CELONIS-NSP-1/2。此外，我们还使用了集合相似性搜索领域中四个常见且可免费获取的数据集，即 ENRON、REUTERS、Flickr 和 KOSARAK。

为证明 FINEX 能够推广到其他数据类型和距离函数，我们使用欧几里得距离对四个多维向量数据集进行评估，即 HOUSEHOLD、HT-SENSOR、PRECIPITATION 和 GAS-SENSOR。我们已将向量数据集中的变量标准化为均值为零、方差为一，这是

表 2。我们实验中数据集的特征。

	数据集	Size	去重后的大小
Sets	CELONIS-1	4。9 · 107	8.2 · 106
	CELONIS-2	1。6 · 107	2.6 · 106
	CELONIS-NSP-1	1.4 · 108	1.2 · 105
	CELONIS-NSP-2	1。2 · 107	6.5 · 106
	ENRON [24]	5.2 · 105	2.5 · 105
	REUTERS [24]	7.8 · 105	7.4 · 105
	Flickr [24]	1。7 · 106	1。2 · 106
	KOSARAK [24]	9.9 · 105	6.1 · 105
	数据集	**Size**	**# 维度**
Vectors	降水量 [17]	5.7 · 105	12
	家庭用电 [17]	2.0 · 106	7
	HT-Sensor [20]	9.3 · 105	10
	气体传感器 [17]	4.2 · 106	16

在使用欧几里得距离时确保各变量权重相等的常见做法 [11, 22]。表 2 列出了所有数据集及其特征。

数据去重. 作为集合数据的一种优化，所有参与比较的算法都利用了数据集包含大量重复集合这一特性（参见表2）。我们的实现会在将集合读入内存时对其进行去重，并为每个集合存储一个重复计数。这确保了聚类结果的正确性，同时数据只需部分物化在内存中。重复计数用于正确确定邻域大小。多维向量数据不太可能包含重复项，因此我们不会对向量数据集进行去重。

实现. 我们在统一fi的 C++ 框架中实现了 FINEX、DBSCAN 和OPTICS。此外，AnyDBC 的C++代码由原作者慷慨提供，我们对其进行了调整，使其能够高效ffi地处理集合数据。给定通过算法2计算得到的 FINEX 排序，可以使用要么 $\varepsilon^* \leq \varepsilon$（精确和近似）要么 $MinPts^* \quad MinPts$（仅精确）查询任意数量的聚类。类似地，可以查询预计算的 OPTICS 排序，以获得任意 $\varepsilon^* \leq \varepsilon$ 的近似聚类。生成的聚类可以选择以表格形式保存到磁盘，其中将每个对象映射到一个簇标识fi符，同时提供相应对象的核心状态信息。请注意，所有算法均以单核心方式执行。

***AnyDBC* 配*fi*置**. AnyDBC [17] 需要配fi置两个额外的输入参数（α，β），用于定fi义邻域计算的批大小。正如原始论文 [17], 所提出的，我们统一设置 $\alpha = 512$ 和$\beta = 4096$。此外，AnyDBC 使用随机种子来选择执行初始邻域计算的对象。为考虑 AnyDBC 的随机种子，所有 AnyDBC 运行时间均为 fi次运行的平均值。

邻域计算。 从运行时间来看，基于密度的聚类中开销最大的是计算 D中对象的邻域。对于集合数据，我们使用一种先进的倒排列表索引，该索引利用前缀*fix*和长度 fi滤波 [21]。我们的 DBSCAN、OPTICS 构建和 FINEX 构建实现会预先将邻域计算作为单独步骤执行，并将所有邻域物化。因此，我们利用短前缀*fix*，这是一种针对集合数据的特定优化，可为每个 $x \in D$部分计算邻域。通过利用 Jaccard 距离 [18]的对称性，可以推导出 x 的完整邻域。邻域计算

表 3. FINEX 和 OPTICS 在所有数据集上的边界对象平均召回率（$\varepsilon = 0.25, MinPts = 64$）。

ε^*	0.25	0.23	0.21	0.19	0.17	0.15	0.13	0.11	0.09	0.07
FINEX	1.000	0.977	0.966	0.957	0.949	0.936	0.932	0.923	0.909	0.884
OPTICS	0.944	0.942	0.945	0.944	0.943	0.935	0.932	0.923	0.909	0.884

表 4. FINEX 构建和 OPTICS 构建相对于从头开始运行 DBSCAN 的运行时间（$\varepsilon = 0.25, MinPts = 64$）。

	FINEX 构建	OPTICS 构建
CELONIS-1	1.03	1.03
CELONIS-2	1.01	1.02
CELONIS-NSP-1	1.00	1.00
CELONIS-NSP-2	1.00	1.00
ENRON	1.00	1.00
REUTERS	0.97	0.97
Flickr	1.12	1.10
KOSARAK	1.03	1.02
降水量	1.23	1.17
家庭用电	1.60	1.39
HT-Sensor	1.25	1.16
气体传感器	1.16	1.12

对于 FINEX' $MinPts^*$-查询，可以采用类似方式进行优化。AnyDBC旨在仅对所有对象的一个子集执行邻域计算，其中针对对象 $x \in D$ 的每个范围查询都必须返回完整的邻域 x。因此，AnyDBC无法利用短前ɑ缀优化。同样，由于候选对象和核心对象彼此不相交，短前ɑ缀优化也不适用于 FINEX'ε^*-查询。

在所有向量数据实现中，邻域计算均由 Kennel [14] 提出的内存型、缓存友好的*kd*树 加速。对于向量数据，不会预先物化邻域。请注意，所有报告的聚类运行时间均包含相应的邻域计算时间。

机器配置. 我们使用一台64位机器，配备2颗物理Intel Xeon E5-2603v4 CPU，主频1.70GHz，每颗有6个核心对象。这些核心对象共享15 MiB的L3缓存，并各自拥有额外的256 KiB独立L2缓存。该系统配备96 GiB随机存取存储器，运行Debian Buster（Linux 4.19.0-14-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30)）。

6.1 索引比较

在本节中，我们从 ε^*-查询准确率和索引构建时间两方面，将 FINEX 与当前最先进的 OPTICS 索引进行比较。

准确率。 OPTICS 和 FINEX 都可以使用算法 1 在线性时间内查询，以获取近似聚类。从第 5.2 节可知，FINEX 对精确聚类的近似程度优于 OPTICS。为说明这一点，表 3 汇总了 FINEX 和 OPTICS 在所有数据集以及不同 $\varepsilon^* \leq \varepsilon(\varepsilon = 0.25, MinPts = 64)$下边界对象的平均召回率。与 FINEX 相比，OPTICS 平均会漏掉多达 5.6 % 的边界对象，并且在较高 ε^* 值下，差异更为显著。根据定理 5.3，FINEX 能正确分配所有相对于 ε^* 而言、同时也是相对于 ε而言的边界对象。OPTICS 则无法保证这一点。请注意，FINEX 的准确率为 100%

对于 $\varepsilon^* = \varepsilon$。随着 ε^* 减小，FINEX 和 OPTICS 的召回率趋于一致，因为非核心边界对象的数量不断减少，直到剩余的边界对象仅为曾为核心对象，而这两种索引对它们的处理方式相同（参见定理 5.4）。

构建时间。与 DBSCAN 类似，FINEX 构建（算法 2）和 OPTICS 构建的运行时间复杂度主要取决于邻域计算的复杂度。因此，三种算法具有相同的渐近时间复杂度。表 4 展示了 FINEX 构建和 OPTICS 构建相对于从头开始执行 DBSCAN 的运行时间，适用于$\varepsilon = 0.25$ 和 $MinPts = 64$。小于（大于）1 的数值表示构建相应索引比从头开始执行 DBSCAN 更快（更慢）。我们观察到，在集合数据上，FINEX 构建、OPTICS 构建和 DBSCAN 的性能几乎完全相同。然而，在向量数据上，FINEX 构建和 OPTICS 构建分别比 DBSCAN 慢最多 60% 和 39%。我们将其归因于 FINEX 构建和 OPTICS 构建中的优先队列操作，以及修改队列的核心对象占比更高（在我们的向量数据集和集合数据集上，核心对象的平均占比分别为 85.8% 和 46.2%，$\varepsilon = 0.25$ 以及 $MinPts = 64$）。此外，与 OPTICS 相比，FINEX 在队列上执行了更多插入和弹出操作，因此构建时间略高，即比从头开始执行 DBSCAN 高出 4% 至 21%。

6.2 ε^*-Qu系列

本节评估集合数据和向量数据上的 ε^*查询。对于 Jaccard 距离和欧几里得距离，生成的（ε，$MinPts$）对分别固定为 fi，$\varepsilon = 0.25$ 和 $MinPts = 64$。我们使用

$$\varepsilon^* \ 2\{\ 0.25,\ 0.23{,}0.21,\ 0.19{,}0.17{,}0.15{,}0.13{,}0.11,\ 0.09{,}0.07\}.$$

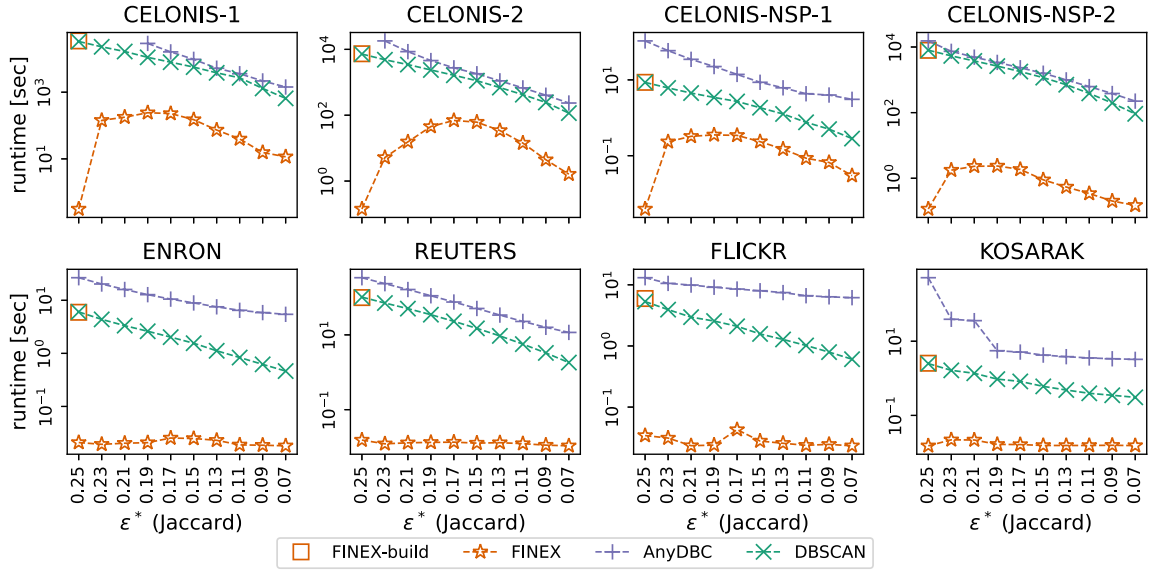


图 6. 集合数据上的聚类运行时间（Jaccard 距离；生成参数为 $\varepsilon = 0.25$ 和 $MinPts = 64$）。

集合数据上的结果。图 6 对比了精确 FINEX 聚类与竞争方法 DBSCAN 和 AnyDBC 的运行时间。在所有设置下，FINEX 的表现都优于竞争方法，且通常快几个数量级。这些运行时间

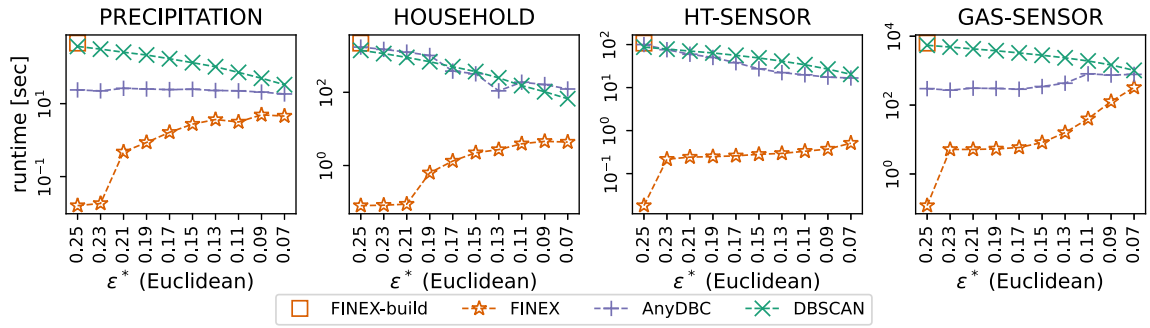


图 7. 向量数据的聚类运行时间（欧几里得距离；生成 $\varepsilon = 0.25$ 和 $MinPts = 64$）。

对于较低的 ε^* 值，DBSCAN 和 AnyDBC 的运行时间会缩短，因为基于前缀的邻域计算在距离阈值较低时效率更高。AnyDBC 在集合数据上的表现最差，我们认为有两个原因：(i) 与 Jaccard 距离结合时，AnyDBC 无法有效剪枝邻域计算。在图 6 所示的所有 AnyDBC 运行中，被避免的邻域计算的中位数为 0.4%（最小值为 = 0%，最大值为 = 18.4%）。我们认为这是因为 AnyDBC 用于区分核心对象的 $3\varepsilon^*$ 上界在 Jaccard 距离下效果较差。(ii) AnyDBC 无法使用第 6 节开头所述的短前缀优化，而 DBSCAN 可以使用。CELONIS-1 和 CELONIS-2 的图中缺少 AnyDBC 在较高 ε^* 值下的数据点，因为执行在超时 12 小时后终止。

当 $\varepsilon^* = \varepsilon = 0.25$ 时，FINEX 的效率最高。在这种情况下，只需对 FINEX 排序进行线性扫描，即可获得精确聚类。在其他所有情况下，还需要额外的候选验证步骤（参见第 5.3 节）才能获得精确聚类。此外，图 6 显示 FINEX 的运行时间呈钟形。这是由于候选对象与核心对象的匹配所致，该过程涉及 $O(n_k \cdot n_c)$ 次验证，其中 n_k 是候选对象的数量，n_c 是核心对象的数量。然而，根据定理 5.6，每个候选对象都曾是核心对象。因此，当 n_k 较低时，n_c 较高（反之亦然）。因此，当 n_k 和 n_c 均为中等规模时，查询成本最高。

向量数据上的结果. 在图 7 中，AnyDBC 在 PRECIPITATION 和 GAS-SENSOR 数据集上的表现优于 DBSCAN，而在 HOUSEHOLD 和 HT-SENSOR 数据集上，两者表现相近。与集合数据相比，AnyDBC 能够有效地剪枝邻域计算。图 7 所示的所有 AnyDBC 运行中，避免的邻域计算中位数为 48.6%（最小值 = 17%，最大值 = 90.9%），这说明 AnyDBC 是向量数据上更优的基线算法。然而，在所有向量数据集和参数设置下，FINEX 的表现均优于这两种竞争算法，运行时间差异最高可达三个数量级（参见 HOUSEHOLD）。当 ε^*取值较小时，GAS-SENSOR 数据集上的算法差异最小。这是因为在生成距离 $\varepsilon = 0.25$下，GAS-SENSOR 几乎完全由核心对象构成（超过 99.8%）。因此，减小 ε^* 时会产生许多候选对象，FINEX 需要执行相对大量的邻域计算，以根据剩余核心对象验证这些候选对象。

6.3 $MinPts^*$-Qu系列

接下来，我们在集合数据和向量数据上评估 $MinPts^*$-查询。对于 Jaccard 距离和欧几里得距离，生成的 (ε, $MinPts$) 对固定为 fixed，其中 $\varepsilon = 0.15$ and$MinPts = 16$ 。我们查询 $MinPts^*$ $MinPts$ with

$$MinPts^* \in \{16, 32, 64, 128, 256, 512, 1024\}.$$

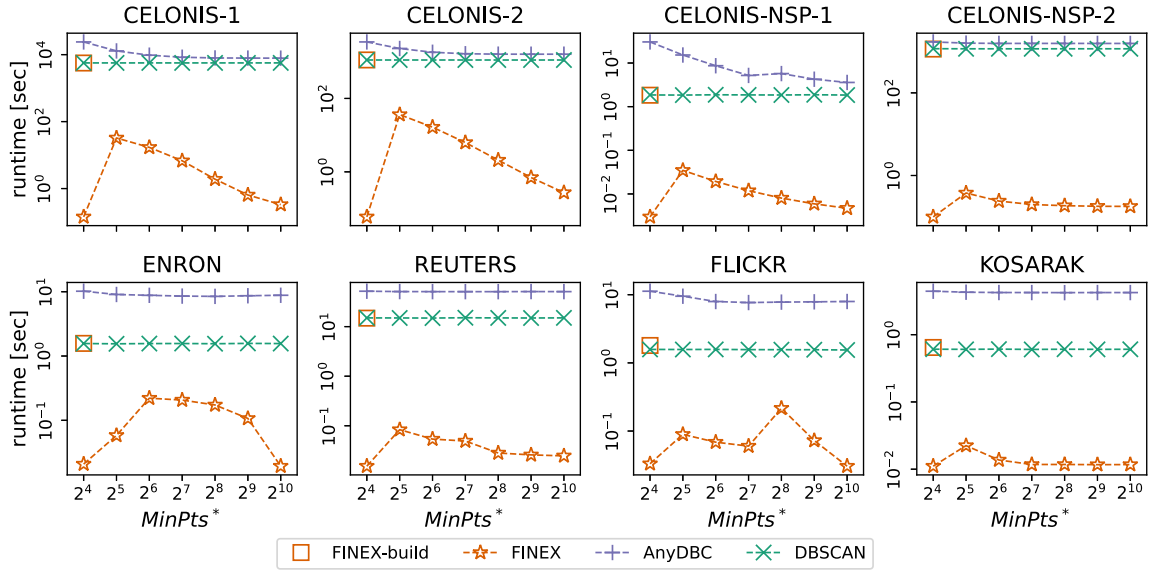


图 8. 对于集合数据，关于 $MinPts^*$ $MinPts$ 的聚类运行时间（Jaccard 距离；生成 $\varepsilon = 0.15$ 和 $MinPts = 16$）。

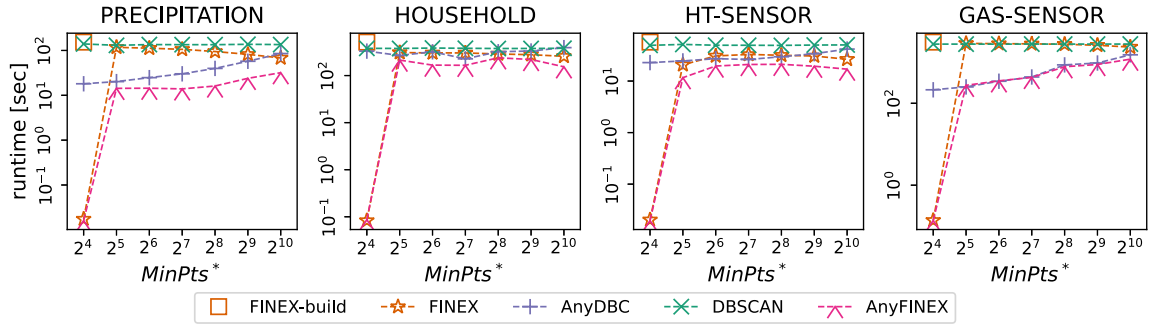


图 9. C 聚类运行时间关于 $MinPts^*$ $MinPts$ 的向量数据（欧几里得距离；生成 $\varepsilon = 0.15$ 和 $MinPts = 16$）

集合数据上的结果. 图 8 显示，FINEX在所有研究场景中都优于 DBSCAN 和 AnyDBC。即使在最坏情况下（ENRON 和 REUTERS），我们也观察到其运行时间相较于 DBSCAN提升了约一个数量级。请注意，DBSCAN的运行时间对 $MinPts^*$ 的变化不敏感，因为邻域计算的计算复杂度仅取决于 ε。这并不一定适用于 AnyDBC，因为

在集合数据上所需时间比FINEX和 DBSCAN 更长。同样，AnyDBC 遭受ff第 6.2 节所述的相同限制。在图 8 所示的集合数据上，所有AnyDBC运行中避免的邻域计算次数的中位数为 0.2%；而在图9 所示的向量数据上，所有 AnyDBC 运行中该数值为 43.8%。当$MinPts^* = MinPts = 16$由于对FINEX 排序进行线性扫描即可得到精确聚类时（参见推论 5.5），FINEX 的效率最高ffi。

对于 $MinPts^* \quad MinPts$，FINEX 的 $MinPts^*$ 查询的计算复杂度取决于对密度连通的核心对象的搜索（算法 4）。核心对象的数量会随着 $MinPts^*$ 的增加而减少。因此，随着 $MinPts^*$ 增大，计算复杂度也预计会降低。我们发现 ENRON 在 $MinPts^* = 2^6$ 处以及 FLICKR 在 $MinPts^* = 2^8$ 处不符合这一规律，运行时间反而增加。我们将此归因于对 $MinPts^*$ 查询的优化：只有当存在一个满足 $MinPts^* > |N_\varepsilon(x)| \quad MinPts$ 的对象 $x \in D$ 时，才需要确定密度连通的核心对象。如果不存在这样的对象，那么所有关于 $(\varepsilon, \quad MinPts)$ 的核心对象，在 $(\varepsilon, \quad MinPts^*)$ 下仍保持其核心状态。因此，可以直接从 FINEX 排序中的稀疏聚类获得密度连通分量。在上述情况中，保持核心状态的核心对象数量大幅减少，因此需要额外进行代价高昂的密度连通分量搜索。

向量数据上的结果。 图 9 展示了我们在向量数据上针对 $MinPts^*$-查询的运行时间结果。我们观察到，在 GAS-SENSOR 和 PRECIPITATION 数据集上（当 $MinPts$ 值较低时），AnyDBC 的性能优于 FINEX。对于 HOUSEHOLD 和 HT-SENSOR，AnyDBC 与 FINEX 的运行时间相近。此外，在几乎所有情形下，AnyDBC 的性能都优于 DBSCAN，因此 AnyDBC 是 $MinPts^*$-查询更合适的基线。因此，我们还评估了*AnyFINEX*，这是一个概念验证实现，展示了 FINEX 的一种变体：它使用 AnyDBC 来识别密度连通分量（而不是类似 DBSCAN 的算法 4）。简而言之，AnyFINEX 将 FINEX 的噪声 fi滤与更快的基线相结合，从而减少总体邻域计算次数。例如，在 PERCIPITATION 上，AnyFINEX 的优fi势有两方面：（a）当 $MinPts$ 值较小时，使用 AnyDBC 识别密度连通分量，因此运行时间较短。（b）当 $MinPts$ 值较高时，FINEX 的运行时间会因噪声 fi滤而降低。相比之下，AnyDBC 的运行时间会随 $MinPts$ 增加，因为它无法利用 FINEX 的噪声 fi滤。在 GAS-SENSOR 上，FINEX 几乎无法 fi滤掉噪声对象，因此 AnyFINEX（分别为 FINEX）与 AnyDBC（分别为 DBSCAN）的运行时间相近。总体而言，我们的实验结果证实，AnyFINEX 整合了 AnyDBC和FINEX 的优fi势。由于 DBSCAN 在集合数据上的速度快于 AnyDBC，我们预计 AnyFINEX 在集合上不会带来运行时间方fi益。

7 结论

我们研究了针对变化参数 ε 和 $MinPts$识别精确的基于密度的聚类这一问题。我们提出了 FINEX，这是第 fi个克服现有解决方案缺陷的线性空间索引，即效ffi率、灵fl活性以及对变化输入参数的支持。FINEX 针对生成对 $(\varepsilon, MinPts)$ 构建，并能够返回关于 $\varepsilon^* \leq \varepsilon$ 或 $MinPts^* \quad MinPts$的精确基于密度的聚类。此外，FINEX 排序使我们的索引能够生成更准确的聚类（与OPTICS 相比），并能够在线性时间内生成针对 $\varepsilon^* = \varepsilon$的精确聚类。此外，FINEX 利用基于密度的簇的基本性质，有效地剪枝开销高昂的邻域计算。在针对 12个大型真实世界数据集（其中包含集合值对象和多维向量对象）的实验中，FINEX 的性能通常比其竞争方案高出多个数量级。

未来工作. 与 OPTICS [1], 不同，FINEX 目前无法将对底层数据集的更新纳入其中，也就是说，当对象被添加到数据集或从数据集中移除时，必须重建索引。因此，我们计划研究适用于 FINEX 的高效更新策略。

FINEX 被设计为一种高效的内存索引。对 FINEX 进行线性（顺序）扫描，以检索 ε^* 查询所需的核心对象和候选对象，这也有利于将其扩展到外存。将这些对象连续存储在外存后，即可扫描它们以执行邻域计算。高效检索邻域是一个正交问题，FINEX 将受益于现有的外存邻域索引。*MinPts* 查询以类似的方式实现。将 FINEX 扩展为外存索引并对其进行测试，是一个有趣的未来研究方向。

致谢

本研究全部或部分由奥地利科学基金（FWF）P 34962 资助。为实现开放获取，作者已对因本次投稿产生的任何作者接受稿版本采用 CC BY 公共版权许可。

参考文献

[1] Elke Achtert、Christian Böhm、Hans-Peter Kriegel 和 Peer Kröger。2005。数据仓库环境中的在线层次聚类。载于第五届 IEEE 国际数据挖掘会议（ICDM '05）论文集。IEEE 计算机学会，美国。[2] Mihael Ankerst、Markus M. Breunig、Hans-Peter Kriegel 和 Jörg Sander。1999。OPTICS：通过点排序识别聚类结构。载于 ACM SIGMOD 数据管理国际会议（SIGMOD '99，第 28 卷）论文集。美国纽约州纽约市：ACM，49–60。[3] Nikolaus Augsten 和 Michael Bohlen。2013。关系数据库系统中的相似性连接（第 3 版）。加利福尼亚州圣拉斐尔：Morgan & Claypool 出版社。[4] Jon Louis Bentley。1975。用于关联搜索的多维二叉搜索树。《ACM 通讯》18，9（1975），509–517。[5] Stefan Brecheisen、Hans-Peter Kriegel 和 Martin Pfeifle。2006。复杂对象的并行基于密度的聚类。载于知识发现与数据挖掘进展（PAKDD '06）。Springer Berlin Heidelberg，179–188。[6] Ricardo J.G.B. Campello、Davoud Moulavi 和 Jörg Sander。2013。基于层次密度估计的基于密度的聚类。载于知识发现与数据挖掘进展（PAKDD '13）。Springer Berlin Heidelberg，160–172。[7] Paolo Ciaccia、Marco Patella 和 Pavel Zezula。1997。M-tree：度量空间中相似性搜索的高效访问方法。载于第 23 届超大型数据库国际会议（VLDB '97）论文集。加利福尼亚州旧金山：Morgan Kaufmann 出版社，426–435。[8] Martin Ester、Hans-Peter Kriegel、Jörg Sander 和 Xiaowei Xu。1996。用于在大型空间数据库中发现含噪声簇的基于密度的算法。载于第二届知识发现与数据挖掘国际会议（KDD '96）论文集。AAAI 出版社，226–231。[9] Junhao Gan 和 Yufei Tao。2018。低维空间中具有有界精度的快速欧几里得 OPTICS。载于 2018 年数据管理国际会议（SIGMOD '18）论文集。计算机协会，美国纽约州纽约市，1067–1082。[10] Antonin Guttmann。1984。R 树：用于空间搜索的动态索引结构。载于 1984 年 ACM SIGMOD 数据管理国际会议（SIGMOD '84，第 14 卷）论文集。计算机协会，美国纽约州纽约市，47–57。[11] Jiawei Han、Micheline Kamber 和 Jian Pei。2012。数据挖掘：概念与技术（第 3 版）。Morgan Kaufmann。[12] B. F. A. Hompes、J. C. A. M. Buijs、W. M. P. van der Aalst、P. M. Dixit 和 J Buurman。2015。使用轨迹聚类发现偏离案例和流程变体。载于第 27 届比荷卢人工智能会议（BNAIC 2015）。比利时哈瑟尔特。[13] Jennifer Jang 和 Heinrich Jiang。2019。DBSCAN++：迈向快速且可扩展的密度聚类。载于第 36 届国际机器学习会议论文集，第 97 卷。PMLR，3019–3029。[14] Matthew B Kennel。2004。KDTREE 2：用于在多维欧几里得空间中高效搜索近邻的 Fortran 95 和 C++ 软件。arXiv 预印本 physics/0408067（2004）。[15] Daniel Kocher、Nikolaus Augsten 和 Willi Mann。2021。将基于密度的聚类扩展到大型集合。载于第 24 届数据库技术扩展国际会议论文集。EDBT，109–120。[16] Son T Mai、Ira Assent 和 Martin Storgaard。2016。AnyDBC：一种适用于超大型复杂数据集的高效任意时间基于密度的聚类算法。载于第 22 届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集。1025–1034。[17] Son T. Mai、Jon Jacobsen、Sihem Amer-Yahia、Ivor Spence、Nhat-Phuong Tran、Ira Assent 和 Quoc Viet Hung Nguyen。2022。多核处理器上的增量式基于密度的聚类。《IEEE 模式分析与机器智能汇刊》44，3（2022），1338–1356。[18] Willi Mann、Nikolaus Augsten 和 Panagiotis Bouros。2016。集合相似性连接技术的实证评估。载于《VLDB 论文集》第 9 卷。美国加利福尼亚州旧金山，636–647。[19] Natalie Price-Jones 和 Jo Bovy。2019。使用 DBSCAN 进行盲化学标记：光谱巡天的前景。《皇家天文学会月报》487，1（2019），871–886。[20] UCI 机器学习资源库。2022。HT-SENSOR。http://archive.ics.uci.edu/ml/datasets/gas+sensors+for+home+activity+monitoring。访问日期：2022 年 10 月。

[21] Leonardo Andrade Ribeiro 和 Theo Härder。2009。使用最小前缀的高效集合相似性连接。载于东欧数据库与信息系统进展会议第 *13* 届会议论文集。ADBIS，88–102。[22] scikit-learn。2022。标准化，或均值移除与方差缩放。https://scikit-learn.org/stable/modules/preprocessing.html。访问日期：2022 年 10 月。[23] Kevin Sheridan、Tejas G Puranik、Eugene Mangortey、Olivia J Pinon-Fischer、Michelle Kirby 和 Dimitri N Mavris。2020。DBSCAN 聚类在进近阶段飞行异常检测中的应用。载于*AIAA Scitech 2020*论坛。[24] Xubo Wang、Lu Qin、Xuemin Lin、Ying Zhang 和 Lijun Chang。2019。在精确且动态的集合相似性连接中利用集合关系。载于*The VLDBJournal*。267–292。[25] Chuan Xiao、Wei Wang、Xuemin Lin、Jeffrey Xu Yu 和 Guoren Wang。2011。用于近重复检测的高效相似性连接。*ACM*数据库系统汇刊 36，3（2011）。

A 证明

本节将为本文的所有理论结果提供形式化证明。

Proposition 3.9 (ε-Nested Clusters). 设 $0 \le \varepsilon^* \le \varepsilon$。对于每个相对于 $(\varepsilon^*, MinPts)$ 的簇 $^* \subseteq D$，存在一个相对于 $(\varepsilon, MinPts)$ 的簇$\subseteq D$，使得 $^* \subseteq$ 。

Proof. 设$K^* \surd D$是一个相对于 ε^* 和 $MinPts$ 的簇，并设 $p \in K^*$ 是一个相对于 $(\varepsilon^*, MinPts)$ 的核心对象。由于 $\varepsilon^* \le \varepsilon$，可得

$$N_{\varepsilon^*}(p) = \{x \in D : d(x,p) \le \varepsilon^*\} \subseteq \{x \in D : d(x,p) \le \varepsilon\} = N_\varepsilon(p)$$

因此

$$MinPts \le |N_{\varepsilon^*}(p)| \le |N_\varepsilon(p)|$$

根据 [8],中的引理1，集合

$$K := \{x \in D : x \text{ is density-reachable from } p \text{ w.r.t. } \varepsilon \text{ and } MinPts\}$$

是关于 ε和 $MinPts$的 D中的一个簇。现在，令 $q \in K^*$ 为任意对象。根据定义3.4中的连通性可知，q关于 ε^* 和 $MinPts$从 p密度可达（请记住，p是K^*的核心对象）。也就是说，存在 p_1、……、$p_n \in D$、$p_1 = p$、$p_n = q$，使得

$$p_{i+1} \in N_{\varepsilon^*}(p_i) \wedge |N_{\varepsilon^*}(p_i)| \ge MinPts \quad \forall i \in \{1, \ldots, n-1\}$$

不过，对于所有 i，都有 $N_{\varepsilon^*}(p_i) \surd N_\varepsilon(p_i)$。因此

$$p_{i+1} \in N_\varepsilon(p_i) \wedge |N_\varepsilon(p_i)| \ge MinPts \quad \forall i \in \{1, \ldots, n-1\}$$

因此，q关于 ε和 $MinPts$从 p密度可达。由此可得 $q \in K$。 □

下面两个引理给出了簇排序的其他性质。它们将在定理 4.3 的证明中使用（采用反证法）。

Lemma A.1（Effects of Direct Density-Reachability）。令$O_{\varepsilon,\ MinPts}$ 为一个 *OPTICS* 排序，并令 $\varepsilon^* \in \mathbb{R}_{\ge 0}$ 满足$\varepsilon^* \le \varepsilon$。此外，令 x、$y \in O_{\varepsilon,\ MinPts}$ 满足：

(1) y 相对于 ε^*从 x 直接密度可达(2)
$x.P < y.P$

那么，$\forall o \in O_{\varepsilon, MinPts}$ 其中 $x.P < o.P \le y.P$满足 $o.R \le \varepsilon^*$。

Proof。根据 $x.P < y.P$，我们知道，在处理 x 时， y 尚未处理。由于 x 在 ε^* 处是核心对象，且 $d(x,y) \leq \varepsilon^*$，因此在处理 x 后，会处理满足 $y \in Q$ 且具有 $y.R \leq \varepsilon^*$ 的对象。从这一时刻起，内层循环至少会一直执行，直到 y 被处理（优先队列 Q 非空）。当 y 位于 Q 中时， $y.R$ 可能会进一步减小，但不可能增大。也就是说，在处理 y 时，$y.R \leq \varepsilon^*$ 仍然成立。此外，内层 OPTICS 循环会从 Q 的所有元素中，选择 $o.R$ 最小的元素 o。因此，对于所有在 x 和 y 之间处理的对象 o，都有 $o.R \leq y.R \leq \varepsilon^*$。□

直观地说，引理 A.1 表明，如果对象 y 相对于某个 $\varepsilon^* \leq \varepsilon$ 从另一个对象 x 直接密度可达，并且 x 在 y之前已处理，那么所有在此期间已处理的对象所分配的可达距离至多为 ε^*。对于更一般的密度可达概念，也存在类似的性质。

Lemma A.2 (Effects of Density-Reachability). 设$O_{\varepsilon,MinPts}$ 是一个 *OPTICS* 排序，并且 $\varepsilon^* \in \mathbb{R}_{\geq 0}$ 满足 $\varepsilon^* \leq \varepsilon$。此外，设 x_1，……，$x_n \in O_{\varepsilon,MinPts}$ 满足：

(1) x_n 相对于 ε^*，通过 $x_2, \ldots, x_{n-1}$(2) 从 x_1 密度可达；对于所有 $i = 2, \ldots, n$，都有 $x_1.P < x_i.P$

于是，有 $\forall o \in O_{\varepsilon,MinPts}$，其中 $x_1.P < o.P \leq x_n.P$ 使得 $o.R \leq \varepsilon^*$。

Proof. 我们通过归纳法证明该命题。

$n = 2$: 直接由引理 A.1 得出。

$n \to n + 1$: 我们区分两种情况 .

(i) $x_n.P < x_{n+1}.P$

根据假设， x_{n+1} 必须关于 ε^* 与 x_n 直接密度可达。因此，由引理 A.1 可得

$$\forall o \in O_{\varepsilon,MinPts} \text{ with } x_n.P < o.P \leq x_{n+1}.P : o.R \leq \varepsilon^*$$

此外，根据归纳假设，我们知道

$$\forall o \in O_{\varepsilon,MinPts} \text{ with } x_1.P < o.P \leq x_n.P : o.R \leq \varepsilon^*$$

结合两者即可得出该命题。

(ii) $x_n.P > x_{n+1}.P$

该命题可由归纳假设直接推出。

□

换言之，如果 x_n 关于 ε^* 与 x_1 密度可达，并且在将其连接到 x_n 的所选对象中， x_1 是最先被处理的对象，那么中间处理的所有对象所分配到的可达距离都不超过 ε^*。请注意，除了 x_1 是第一个对象之外，对象链的处理顺序并无其他限制。例如，要使引理 A.2 成立，并不要求 x_n 是最后一个被处理的对象。

定理heorem 4.3（一个pproximate OPTICS 簇lusters）。令$O_{\varepsilon,\ MinPts}$ 为由 *OPTICS* 计算得到的簇排序。我们考虑一个关于 （ε^*,*MinPts*）的近似簇 S，其定义见定*fi*义 *4.2*。以下命题成立：

(a) ∃ 根据定义 *3.4*，相对于 ε^* 和 *MinPts* 的一个基于密度的簇，使得 $S \subseteq$.(b) $\forall p \in K$：

$p.P \leq x_n.P$(c) $\forall p \in K$：$|N_{\varepsilon^*}(p)| \geq MinPts \implies p \in S$

Proof.**命题 (a)**：根据假设， $x_1 \in S$ 是 ε^* 处的核心对象，并且根据 [8], 中的引理1，该集合

$$K := \{p \in O_{\varepsilon,MinPts} : p \text{ is density-reachable from } x_1 \text{ w.r.t. } \varepsilon^* \text{ and } MinPts\}$$

是一个簇。我们接下来通过强归纳法证明以下命题：

$$\forall i \in \{1,\ldots,n\} : x_i \in S \text{ is density-reachable from } x_1 \in S \text{ w.r.t. } \varepsilon^* \text{ and } MinPts$$

$i = 1$: 显然成立。

$i = 1, \ldots, k \; ! \; i = k + 1$: 根据假设，我们有 $x_1.R > \varepsilon^*$，即，当 x_1 被处理时，它必定已在优先队列 Q 中

$$\forall p \in Q : \varepsilon^* < x_1.R \leq p.R$$

否则， x_1 就不会被选取。注意，这包括 Q 为空的情况（此时， x_1 会在 OPTICS外层循环中被处理，并且 $x_1.R = \infty$）。因此， x_{k+1} 只有在被插入 Q 和/或被其在 S中的某个前驱对象减小时，才可能与$x_{k+1}.R \leq \varepsilon^*$ 一起被处理。令 $x_m, 1 \leq m \leq k$ 为最后一个插入或减小 x_{k+1}的对象。那么，可达距离

$$R_{\varepsilon,MinPts}(x_{k+1}, x_m) = x_{k+1}.R \leq \varepsilon^*$$

也就是说， x_{k+1} 相对于 ε^* 和 $MinPts$从 x_m 直接密度可达。将此与归纳假设结合，即可得到所需命题。

我们已证明，对于所有 $x_i \in S$，均有 $x_i \in K$。 □

命题(b)和(c)：假设存在一些在 $x_1 \in S$之前已处理的 p 核心对象K。那么，必定存在一个最左侧的核心对象c 2K。因此， x_1 关于 ε^* 和 $MinPts$ 是密度可达的，该过程通过某些在c之后已处理的对象进行。引理 A.2 得出

$$\forall o \in O_{\varepsilon,MinPts} \text{ with } c.P < o.P \leq x_1.P : o.R \leq \varepsilon^*$$

特别地， $x_1.R \leq \varepsilon^*$，这与定义 4.2中的性质(2) 相矛盾。因此，在 x_1 之前不存在已处理的核心对象。

现在假设存在一个对象 $p \in K$，其具有 $p.P > x_n.P$（即，在 $x_n \in S$之后已处理的对象）。 p 必须相对于 ε^* 和 $MinPts$ 从 x_1 密度可达（根据定义 fi3.4）。由于 x_1 是最左侧的核心对象，我们可以再次应用引理 A.2。由此得到

$$\forall o \in O_{\varepsilon,MinPts} \text{ with } x_1.P < o.P \leq p.P : o.R \leq \varepsilon^*$$

这与定义 4.2 中的假设 (4) 矛盾。因此，在 x_n之后不存在已处理的对象。结合这两个结果，即可得出所需的命题。 □

定heorem 5.2（Approximate FINEX 簇lusters）。设$\neg O_{\varepsilon,MinPts}$ 是通过算法 *2* 计算得到的 *FINEX* 排序。根据定义 *4.2*，考虑关于 S 的近似簇$(\varepsilon^*, MinPts)$，有以下命题成立：

(a) 9 一个簇 K 关于 $(\varepsilon^*, MinPts)$ 使得 $S \surd K$.(b)
$\forall p \in K : |N_\varepsilon(p)| \; \blacksquare \; MinPts \; =) \; p.P \leq x_n.P$.(c)
$\forall p \in K : |N_{\varepsilon^*}(p)| \; \blacksquare \; MinPts \; =) \; p \in S$.

Proof. **(a)：** 证明与定理 4.3 的命题 (a) 相同。**(b) 和 (c)：** 证明与定理 4.3 的命题 (b) 和 (c) 类似。我们只需将引理 A.1 中的对象 y 和引理 A.2 中的 x_n 限制为相对于 ε的核心对象。□

Theorem 5.3 (Clustering of Non-Cores). 令 $x \in {}^{\sim}O_{\varepsilon,MinPts}$ 是相对于$(\varepsilon, MinPts)$ 的非核心对象。给定距离 ~阈值 $\varepsilon^* \leq \varepsilon$，将算法 *1* 应用于$O_{\varepsilon,MinPts}$，可正确地将 x 分配给

- 噪声组，如果 x 是相对于 $(\varepsilon^*,MinPts)$ 的噪声对象，或者
- 是相应的簇，当 x 是相对于 $(\varepsilon^*,MinPts)$ 的边界对象时。

Proof。如果 $x \in \mathrm{e}O_{\varepsilon,\ MinPts}$ 相对于 $(\varepsilon^*,\ MinPts)$ 是噪声，那么 x 不能属于定理 5.2 中的任何近似簇 S。因此，该命题成立。

现在，令 x 为相对于（$\varepsilon^*,\ MinPts$）的边界对象。因此，存在一个对象 o，其

$$R_{\varepsilon,MinPts}(x,o) \leq\ \varepsilon^*$$

根据定义 5.1，成立的是

$$x.R\ = \min\{R_{\varepsilon,MinPts}(x,o) : o \in D\}$$

结合这两个结论可得

$$x.R\ \leq \varepsilon^*$$

仍需证明，x 被分配到了其所属的某个簇中（而不仅仅是被分配给任意一个簇）。设 p 为最后一个在优先队列中插入或降低 x 的对象。从

$$x.R\ = R_{\varepsilon,MinPts}(x,p) \leq\ \varepsilon^*$$

我们知道，p 是关于 ε^* 的核心对象，并且 p 和 x 属于同一个簇。根据定理 5.2 (c)，可知 p将被算法 1 检测为簇成员。此外，必须满足 $p.P\ < x.P$。因此，在 p 被处理后，优先队列非空（其中至少包含 x），并且满足

$$\forall\, o \in \widetilde{O}_{\varepsilon,MinPts} \text{ with } p.P\ < o.P\ < x.P\ :\ o.R\ \leq x.R\ \leq \varepsilon^*$$

因此，p 和 x 均被算法 1 分配到同一个簇中。 □

Theorem 5.4 (Clustering of Former-Cores). 令 $x \in D$ 为相对于 $\varepsilon^* \leq \varepsilon$ 和 $MinPts$的曾为核心对象，且令 ~ $O_{\varepsilon,MinPts}$和$O_{\varepsilon,MinPts}$ 分别为 *FINEX* 排序和 *OPTICS* 排序。对于 ε^*，算法 *1* 使用 ${}^{\sim}O_{\varepsilon,MinPts}$ 将*fi*识别x 为簇成员；若*ff* 算法 *1* 使用$O_{\varepsilon,MinPts}$ 将*fi*识别为 x 簇成员。

Proof。我们假设有一个稳定优先队列，在出现平局时（即两个元素的优先级相等时）按插入顺序弹出其元素。令 $x \in D$ 为曾经的核心对象，即相对于 $(\varepsilon, MinPts)$的核心对象，以及相对于 $(\varepsilon^*,MinPts)$ 的边界对象，$\varepsilon^* \leq \varepsilon$。因此，$\varepsilon^* < x.C\ \leq \varepsilon$ 同时属于$\mathrm{e}O_{\varepsilon,MinPts}$ 和 $O_{\varepsilon,MinPts}$（参见定理 4.3 和 5.2）。如果 x 被识fi别为来自$\mathrm{e}O_{\varepsilon,MinPts}$ 的簇成员，则存在一个核心对象 $c 2 \mathrm{e}O_{\varepsilon,MinPts}$ 以及一个对象序列 $x_1,\ldots,x_n \in \mathrm{e}O_{\varepsilon,MinPts}$，具有以下性质

- $c.C\ \leq \varepsilon^*$ 和 $c.R\ > \varepsilon^*$

- $x_1 = c$， $x_n = x$
- $x_{i+1}.P = x_i.P + 1$ 对于 $i = 1$、……、$n-1$
- $x_i.R \leq \varepsilon^*$ 对于 $i = 2$、……、n

换言之，2 初始化了包含 x 的近似簇。在算法 2 中，相对于 $(\varepsilon, MinPts)$ 对非核心对象进行重新处理，不会影响核心对象相对于 $(\varepsilon, MinPts)$ 的可达距离属性。因此，对于 FINEX（即算法 2）和 OPTICS，优先队列会以完全相同的顺序弹出核心对象。于是， x_1、……、x_n 的所有核心对象在 $\mathrm{e}O_{\varepsilon}$、$MinPts$ 和 O_{ε}、$MinPts$ 中的顺序完全相同。此外，对于 x_1、……、x_n 的核心对象， $x_i.R$ 在 $\mathrm{e}O_{\varepsilon}$、$MinPts$ 和 O_{ε}、$MinPts$ 中也相同。特别地， $x.R \leq \varepsilon^*$ 也在 O_{ε}、$MinPts$ 中。由此可知，在 O_{ε}、$MinPts$ 中存在一个近似簇， x 和 2 均属于该簇。因此，当算法 1 应用于 O_{ε}、$MinPts$ 时， x 将被检测为簇成员。证明的另一个方向类似。□

Theorem 5.6 (Exact ε^*-Query)。 设 $^{\sim}O_{\varepsilon,MinPts}$ 是由算法 *2* 计算得到的 *FINEX* 排序。算法 *1* 使用$\varepsilon^* \leq \varepsilon$ 返回近似簇 $S_1, \ldots, S_m$，其参数为 $(\varepsilon^*, MinPts)$。对于每个 S_i,，如果 o 相对于$(\varepsilon^*, MinPts)$ 可从 S_i 中的某个核心对象直接密度可达，且满足以下条件，则将 $o \in {}^{\sim}O_{\varepsilon,MinPts}$添加到 S_i：

(1) o 被标记为噪声，且 $\varepsilon^* < o.C \leq \varepsilon$(2) o 在(3)∃
之前处理，其中$(\varepsilon, MinPts)$对应于一个簇，使得
$S_i \cup \{o\} \subseteq$(4) o 尚未被添加到另一个 S_j, $j \neq i$

根据定义 *3.5*，结果是相对于 $(\varepsilon^*, MinPts)$ 的精确聚类。

Proof。该定理由定理 5.2、定理 5.3 和命题 3.9 共同推出。 □

Corollary 5.5 (Linear Exact Query). 令 $^{\sim}O_{\varepsilon,MinPts}$ 为通过算法 *2* 计算得到的*FINEX* 排序。算法 *1* 使用 $\varepsilon^* = \varepsilon$，根据定义 *3.5*，返回关于$(\varepsilon, MinPts)$的精确聚类，其时间复杂度为线性时间（以对象数量计）。

Proof。不存在对象 o 具有 $\varepsilon^* < o.C. \leq \varepsilon$。因此，该推论直接由定理 5.6 得出。 □

Proposition 5.7 (*MinPts*-Nested Clusters). 设 $MinPts^* \geq MinPts$。对于关于 $(\varepsilon, MinPts^*)$ 的每个簇 $^* \subseteq D$，都存在关于 $(\varepsilon, MinPts)$ 的一个簇 $\subseteq D$，使得 $^* \subseteq$ 。

Proof。设 $K^* \surd D$ 是关于 ε 和 $MinPts^*$的一个簇。设 p 是K^*的核心对象。我们有

$$|N_{\varepsilon}(p)| \geq MinPts^* \geq MinPts$$

根据 [8],中的引理1，集合

$$K \coloneqq \{x \in D : x \text{ is density-reachable from } p \text{ w.r.t. } \varepsilon \text{ and } MinPts\}$$

是 D 中关于 ε 和 $MinPts$的一个簇。现在，任取 $q \in K^*$。由定义3.4中的连通性可知， q 关于 ε 和 $MinPts^*$ 从 p 密度可达（回顾一下， p 是K^*的核心对象），即存在 $p_1, \ldots, p_n \in D$、 $p_1 = p$、$p_n = q$，满足

$$p_{i+1} \in N_{\varepsilon}(p_i) \wedge | \quad N_{\varepsilon}(p_i)| \geq MinPts^* \quad \forall i \in \{1, \ldots, n-1\}$$

显然， $|N_\varepsilon(p_i)|$ $MinPts$对所有i均成立。因此， q关于 ε和 $MinPts$是从 p密度可达的。由此可得 $q \in K$。□
