空间 AkNN 连接的最优纯边界剪枝

Dominik Winecki
winecki.1@osu.edu 俄亥俄州立大学 美国俄亥俄州哥伦布

arXiv:2602.10027v1 [cs.DB] 10 Feb 2026

摘要

我们提出了一种仅基于界限的剪枝测试，用于分区空间数据集上的精确欧几里得 AkNN 连接。数据仓库通常会对大型表进行分区，并存储相应的行组统计信息，以加速搜索和连接，而不是维护索引。通过构建界限，并在加载分区以建立空间索引之前，将连接评估局限于少数几个分区，AkNN 连接可以利用这些统计信息。现有剪枝方法对仅含界限的空间数据过于保守，因为它们未能充分捕捉其方向语义，因此错过了在连接的早期阶段跳过不必要分区的机会。我们提出了一种三边界邻近性测试，用于确定某个分区内的所有点 是否在一个分区中比在另一个可能被遮挡的分区中有更近的邻居。我们证明了该算法既最优又高效。

关键词

最近邻，AkNN 连接，空间数据库，数据仓库

1 引言

K 近邻（kNN）搜索是空间数据分析中无处不在的操作。鉴于其重要性和简洁性，K 近邻和全 K 近邻（AkNN）连接已得到深入研究。大多数相关工作都着重于构建和搜索空间索引；而在未建立索引或仅建立少量索引的数据集上执行全 k 近邻连接，则研究较少。

我们专注于在未建立索引的分区式空间数据集上执行全 k 近邻连接。例如，在数据仓库架构中，大型表会拆分为许多分区文件，并存储在对象存储中。这种模式在更小的规模上也日益普及，例如使用 DuckDB 在笔记本电脑上查询 Parquet 文件目录。这种数据分析方式因避免在插入和更新时维护索引而带来的性能优势 [16] 以及一些实际考量而日益流行，例如存储层的可移植性和互操作性，以及无需预处理即可下载和查询数据集的能力。索引仍会使用，但只会在查询运行期间临时创建。序列化层中提供了一些元数据，可通过行组统计信息加速此类查询。例如，Parquet 会存储每个行组的最小值和最大值，使连接能够通过下推过滤操作，跳过对行组或分区的扫描。我们关注在这些分区式数据存储架构中跨空间数据集执行的全 k 近邻查询；这类架构提供聚合统计元数据，而错过剪枝机会的代价很高：需要读取、解压缩数千行数据，并将其插入索引。

分区剪枝可作为全 k 近邻连接的第一阶段，在读取任何行或构建索引之前剪除分区。随后，可使用算法在底层分区上执行全 k 近邻连接

现有的 k 近邻剪枝方法并不适用于这一第一阶段；它们根据两个边界构造单一距离范围，未能捕捉方向性，因此导致剪枝过于保守。简单来说，当原点分区与更远处的某个分区之间“夹着”一个分区时，中间分区便可用于剪除更远处的分区。

我们提出一种简单的三界限测试，用于确定对于分区 O、E 和 B，O 中的所有点是否都比 B 中的任意点更接近 E 中的所有点。我们证明，在仅考虑分区界限时，这是对全 k 近邻连接进行剪枝的最优方法。我们的测试要么判定某个分区可以剪枝，要么构造示例点，以说明为何不能根据分区界限进行剪枝。这种早期的仅基于界限的剪枝，与后续逐点执行的近似 AkNN 相辅相成且相互正交，因为近似方法无法对未索引数据的分布作出假设。因此，我们的方法为现代空间数据仓库提供了一种简单、最优且实用的改进。

在本文中，我们提出一种三界限测试算法，并证明其正确性和最优性。随后，我们给出该算法的优化版本，将运行时间降至 $O(R)$，其中 R 是维度数量，与信息论下界相匹配。接着，我们证明界限邻近性测试构成严格偏序，从而能够在一般情况下确定加载分区的最优顺序。

2 相关工作

2.1 空间数据仓库

在线分析处理数据库的发展趋势更重视数据布局和压缩，而不是广泛使用索引 [16, 25]。随着核心数越来越多的 CPU 和 GPU 逐渐普及，支持单指令多数据指令的 CPU 也日益可用，这使得蛮力并行处理优先于索引遍历。GPU 上的 k 近邻连接在简单的数据布局 [9, 18, 19, 26] 上速度更快。许多采用列式存储的关系型在线分析处理系统会维护行组的分布信息，其中较为典型的是 Snowflake [6] 和 AWS Redshift。这些机制有时被称为区域映射 [25]。类似地，许多空间数据仓库通过 MapReduce [7] 执行架构 [10, 24], 实现近似 AkNN，通常运行于 Spark [11, 15, 17, 20, 23] 中。这些系统利用分区感知数据处理，高效查询共享磁盘/对象存储层中的数据。

2.2 最近邻

空间最近邻、K 近邻和全 k 近邻查询已在各种空间应用场景中得到广泛研究 [8, 12]。我们将研究范围限定在使用欧氏/L2 距离的空间低维精确全 k 近邻。分区界限剪枝是执行全 k 近邻连接的最早阶段，与后续阶段的逐点执行正交

；后续阶段可能超出这些限制（例如，近似 AkNN），但不会影响前期的分区剪枝。此外，大多数 k 近邻研究侧重于创建空间互不相交分区 [2, 3], , 例如四叉树 [17], , 但我们的应用场景通常包含高度重叠的分区。同样，大多数 k 近邻剪枝研究侧重于针对索引进行点级剪枝；然而，在此应用中，我们严格关注在加载任何点或构建索引之前对分区进行剪枝。

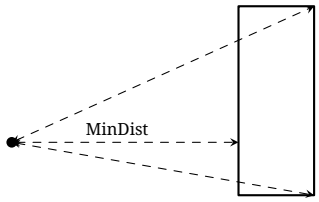


图 1：点到边界距离度量。图中展示了到一个分区的最小距离和最大距离，以及 AABB 上针对 $k = 1$ 的最大距离这一特殊情况（MinMaxDist）。

2.2.1 $k = 1$ 点到边界。Roussopoulos 等人关于近邻查询剪枝的开创性研究，提出通过 MinDist 和 MaxDist 函数进行查询 [13]。这两个函数都返回点到 AABB 的距离界限。MinMaxDist 函数可用于找到更紧的界限，但仅当边界是最小外接矩形时才适用，而且 $k = 1$（这也意味着在需要任何谓词时无法使用）。许多高效的近似 AkNN 算法都基于点到边界剪枝 [21, 22]。

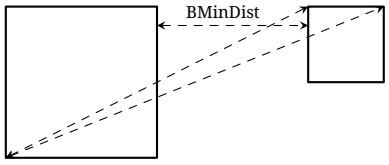


图 2：边界到边界距离度量

2.2.2 边界到边界。严格依据边界进行剪枝的方法出现得较晚。Sankaranarayanan 等人 [14] 使用这些方法定义了最小距离和最大距离函数的边界到边界版本，以下称为 BMinDist/BMaxDist，如图 2 所示。可以按 BMaxDist 排序，并按此顺序选择分区，直到 k 达到饱和。然后，只需搜索 BMinDist 小于或等于饱和分区的 BMaxDist（即它们的剪枝距离）的分区。他们证明，这种方法对于双分区测试是最优的，并且在使用索引时可以高效实现。

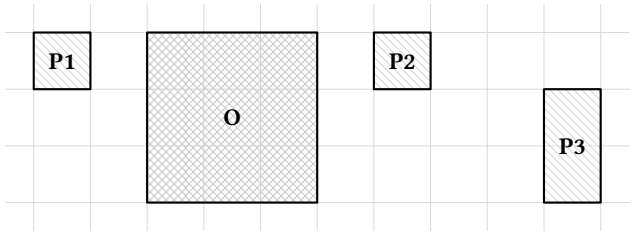


图 3：包含三个邻居搜索分区 P1-P3 的空间分区布局。其中一个分区 P3 可以无需访问即被剪枝，因为另一个分区 P2 中的所有点都比 O 更近。只有我们提出的三分区测试才能识别这一点。

此外，NXNDist [4]（MinMaxMinDist 的简称）是 MinMaxDist[1]的边界到边界对应项。与其点到边界对应项类似，NXNDist 仅适用于 $k = 1$，即具有最小边界且没有额外谓词的情况。

2.2.3 最小外接矩形/轴对齐包围盒术语。大多数 k 近邻研究使用“最小外接矩形”（MBR）这一术语。然而，对于本文而言，“轴对齐包围盒”（AABB）是更恰当的术语，因为我们要求包围盒与坐标轴对齐，但不要求边界最小。在大多数先前的工作中，MBR 实际上都是 AABB；只有 MinMaxDist [13] 和 NXNDist [4] 这两种度量要求边界最小。值得注意的是，省略对边界最小的要求后，便可以对采用空间哈希进行分区的数据集（例如 H3 [1]）进行剪枝。在缺乏行组分布统计信息的情况下，可以根据空间哈希导出 AABB，但无法导出 MBR。

3 问题陈述

我们的任务是判断：在源自第三个边界的 AkNN连接 中，一个边界内的所有点是否都比另一个边界内的所有点更近：

$$\forall o \in O : \forall e \in E : \forall b \in B : \mathrm{dist}(o, e) < \mathrm{dist}(o, b)$$

请看图 3 中的分区布局。一个数据集只有一个分区 **O**；邻居候选数据集有三个分区：**P1**、**P2** 和 **P3**。这三个分区中，哪些可能包含 **O** 的最近邻？显然，必须考虑 **P1** 和 **P2**，但 **P3** 的情况并非显而易见。直观来看，分区 **P2** 大致位于 **O** 与 **P3**之间，这可能 意味着无需加载 **P3**。边界到边界测试无法排除 **P3**。在边界到边界方法中，**P1** 和 **P2** 到 **O** 的距离相同，而不考虑 **P2** 与 **P3**是否位于同一方向。对于图 3 的情况，这一直觉是正确的；无需读取**P3** 。然而，现有的空间算法都无法在从这些分区加载数据之前得出这一结论。

[1]MinMaxDist 指的是 Roussopoulos 等人的定义 [13], , 它不同于 NXNDist 论文中同名的距离度量 [4], ; 正如他们所指出的，后者指的是最近点对定义 [5], , 而非最近邻定义。同样，他们的最小-最小距离即我们的 BMinDist，他们的最大-最大距离即我们的 BMaxDist。

算法 1 AllPointsCloser

输入： 原点轴对齐包围盒 O，评估轴对齐包围盒 E，基准轴对齐包围盒 B

输出： 若 O 中的所有点在 E 中都有最近邻，则返回 true

函数 AllPointsCloser(O, E, B)
  对所有 $p \in$ 角点(O) **执行**
    如果 最小距离(p, B) $\leq$ 最大距离(p, E) **则**
      返回 false
    结束 if
  结束 for
  返回 true
结束函数

4 全点更近分区剪枝

我们希望确定，对于一个 AABB 分区 O（原点），其最近邻是否总是位于分区 E（评估）中，而不是另一个分区 B（基）：

$\forall o \in O\colon \forall e \in E\colon \forall b \in B\colon \mathrm{dist}(o,e) < \mathrm{dist}(o,b)$ 我们在全点邻近定理中提出一个等价关系：

$[\forall o' \in$ 角点$(O)\colon$ 最大距离$(o', E) <$ 最小距离$(o', B)] \Longleftrightarrow$
$\forall o \in O\colon \forall e \in E\colon \forall b \in B\colon \mathrm{dist}(o,e) < \mathrm{dist}(o,b)$

算法 1 直接执行这一测试。根据前述充要等价关系，其正确性和最优性显而易见。图4展示了这一距离测试的二维几何解释。这里有四个圆，每个圆都以一个角点为圆心，且半径恰好足以与基准分区中的最近点相接。这些圆的交集构成一个区域，从原点的角度看，该区域严格比基准边界中的任何点都近。如果该区域内至少存在 k 个点，就无需搜索基准分区。

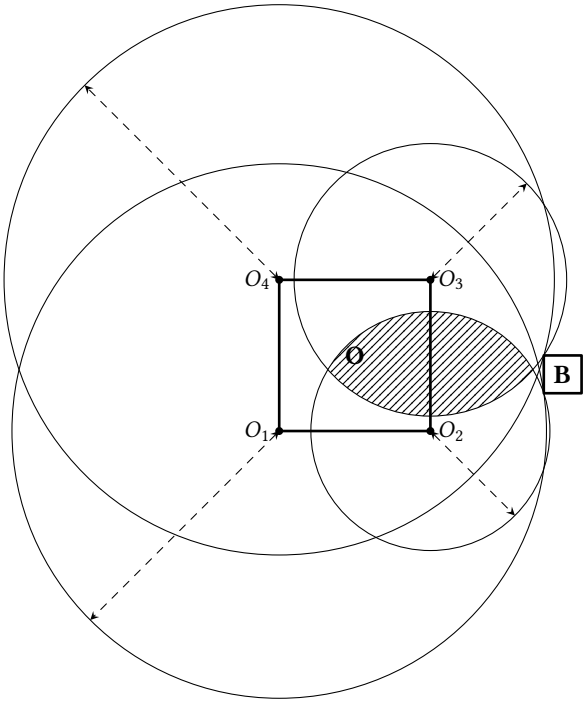


图4：剪枝测试的几何视图。以原点分区的每个角点为圆心，绘制半径至基准分区最近点的圆；完全位于这些圆交集内的任何几何体都必定比基准分区更近。

4.1 所有点邻近定理

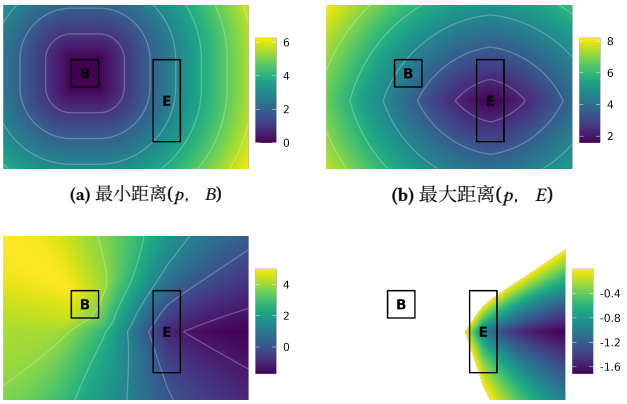
4.1.1 证明概要。轴对齐包围盒是欧几里得空间中的凸集，其角点构成其成员点的凸包。因此，我们可以将任意成员点表示为一个包围该点的轴对齐包围盒各角点的仿射组合。在定理 4.3 中，我们利用詹森不等式的有限形式，将应用于所有成员点的凸函数值限制在该函数应用于角点所得的值之间。具体而言，我们使用图5c所示的曲线“到E 的最大距离减去到 B的最小距离”，并在定理 4.4 中证明该曲线是凸函数。将这个凸的距离差函数应用于凸包，等价于评估轴对齐包围盒的所有成员点。

4.1.2 定义。我们有空间 $\mathbb{R}^r$。该空间包含原点 O、基 B 和评估 E 边界，它们均为轴对齐包围盒。对于一个轴对齐包围盒 M，Corners(M) 表示该边界的 2^r 角点/顶点。 $p \in M$ 表示一个点 p 位于 M的闭边界内。

对于给定的 [13], 轴对齐包围盒 M，根据先前的工作，

$$\mathrm{MaxDist}(p, M) = \max_{q \in M} \mathrm{dist}(p, q) \tag{1}$$

$$\mathrm{MinDist}(p, M) = \min_{q \in M} \mathrm{dist}(p, q) \tag{2}$$



(a) 最小距离(p, B) **(b)** 最大距离(p, E)

(c) 最大距离(p, E) – 最小距离(p, B) **(d)** 最大距离(p, E) < 最小距离(p, B)

图5：所用距离函数的可视化。图5a和图5b展示了现有的点到边界函数。图5c展示了由AllPointsCloser内部使用的距离差函数。

此外， M_d 是该轴对齐包围盒的第 d个维度边界。维度边界在相应维度上有低界（M_d）和高界（M_d）。所有轴对齐包围盒均构造良好，因此对于每个边界，$\check{} M_d \leq \hat{} M_d$。

4.1.3 全点邻近定理的证明。 [2]

定理 4.1（全点邻近定理）。

[∀o' ∈ 角点(O)：最大距离(o', E) ＜最小距离(o', B)] ⇐⇒

∀o ∈ O： ∀e ∈ E： ∀b ∈ B：距离(o, e) ＜距离(o, b)

证明。我们分别证明每个蕴含关系，其中直接的反向蕴含见定理4.2，而非平凡的正向蕴含见定理4.3。□

引理4.2。

$$[\forall o \in O : \forall e \in E : \forall b \in B : \text{dist}(o,e) < \text{dist}(o,b)] \implies$$
$$\forall o' \in \text{Corners}(O) : \text{MaxDist}(o', E) < \text{MinDist}(o', B)$$

证明。我们假设 ∀o ∈ O： ∀e ∈ E： ∀b ∈ B：距离(o, e) ＜距离(o, b)为真，并证明 ∀o' ∈角点(O)：最大距离(o', E) ＜最小距离(o', B)。

所有点都位于各自的边界框内：

$$o' \in \text{Corners}(O) \implies o' \in O \tag{3}$$
$$\text{MaxDist}(o', E) = \max_{e \in E} \text{dist}(o', e) \tag{4}$$
$$\text{MinDist}(o', B) = \min_{b \in B} \text{dist}(o', b) \tag{5}$$

令 $o := o', e := \text{argmax}_{e \in E} \text{dist}(o', e)$，且 $b := \text{argmin}_{b \in B} \text{dist}(o', b)$。根据它们的定义，我们知道 $o \in O, e \in E$ 和 $b \in B$。代入这些值重写后，

$$\text{MaxDist}(o', E) = \text{dist}(o, e) \tag{6}$$
$$\text{MinDist}(o', B) = \text{dist}(o, b) \tag{7}$$

根据我们的假设，$\text{dist}(o, e) < \text{dist}(o, b)$。 □

引理 4.3。

$$[\forall o' \in \text{Corners}(O) : \text{MaxDist}(o', E) < \text{MinDist}(o', B)] \implies$$
$$\forall o \in O : \forall e \in E : \forall b \in B : \text{dist}(o,e) < \text{dist}(o,b)$$

证明。

我们假设 ∀o' ∈ 的角点(O)：最大距离(o', E) ＜ 最小距离(o', B)，并证明 ∀o ∈ O：∀e ∈ E：∀b ∈ B：$\text{dist}(o, e) < \text{dist}(o, b)$。

令 $\text{Corners}(O) = c_1, c_2, \ldots\ldots, c_{(2^r)}$。由于 O 所表示的轴对齐包围盒是凸的，因此可以将任意点 $o \in O$ 表示为这些角点的仿射组合：

$$o = \sum_{i=1}^{2^r} \alpha_i c_i \tag{8}$$

其中 $\forall i \in 1..2^r$： $\alpha_i \geq 0$，且 $\alpha_1 + \alpha_2 + \cdots + \alpha_{(2^r)} = 1$。

由于最大距离和最小距离均为非负数，我们可以针对任意点 p 将其改写为：

$$\text{MaxDist}(p, E) < \text{MinDist}(p, B) \tag{9}$$
$$\text{MaxDist}(p, E)^2 < \text{MinDist}(p, B)^2 \tag{10}$$
$$\text{MaxDist}(p, E)^2 - \text{MinDist}(p, B)^2 < 0 \tag{11}$$

[2]补充材料中提供了使用 Mathlib 在 Lean 4 中进行的机器校验证明。

对其进行参数化，令 $g(p) = \text{最大距离}(p, E)^2 - \text{最小距离}(p, B)^2$。由于 $g(p)$根据定理 4.4 为凸函数，我们可以应用詹森不等式：

$$g\left(\sum_{i=1}^{2^r} \alpha_i c_i\right) \leq \sum_{i=1}^{2^r} \alpha_i \cdot g(c_i) \tag{12}$$
$$g(o) \leq \sum_{i=1}^{2^r} \alpha_i \cdot g(c_i) \tag{13}$$

根据前面的改写，从等式（ 9 11 $g\ p\ <$）到等式（），我们知道 ( ) ⇐⇒ 最大距离(p,) 最小距离(p,)。因此，我们可以写成：

$$\forall o' \in \text{Corners}(O) : \text{MaxDist}(o', E) < \text{MinDist}(o', B) \tag{14}$$
$$\forall o' \in \text{Corners}(O) : g(o') < 0 \tag{15}$$
$$\forall i \in 1..2^r : g(c_i) < 0 \tag{16}$$

由于 $\alpha_i \geq 0$ $g\ c_i\ < 0$ $\alpha_i\ g\ c_i\ \leq$ 且 ( ), $\sum \alpha_i = 1$ 我们知道 $\forall i \in 1..2^r : \cdot$ ( )。由于，至少有一项严格为负，因此总和严格为负：

$$\sum_{i=1}^{2^r} \alpha_i \cdot g(c_i) < 0 \tag{17}$$
$$g(o) \leq \sum_{i=1}^{2^r} \alpha_i \cdot g(c_i) < 0 \tag{18}$$
$$g(o) < 0 \tag{19}$$
$$\text{MaxDist}(o, E) < \text{MinDist}(o, B) \tag{20}$$
$$\max_{e \in E} \text{dist}(o, e) < \min_{b \in B} \text{dist}(o, b) \tag{21}$$

∀$e \in E, b \in B$：距离(o, e) ≤最大$_{e' \in E}$距离(o, e') ＜最小$_{b' \in B}$距离(o, b') ≤距离(o, b)

(22)

$$\forall e \in E : \forall b \in B : \text{dist}(o,e) < \text{dist}(o,b) \tag{23}$$

□

引理 4.4. 对于 $g(p) = \text{最大距离}(p, E)^2 - \text{最小距离}(p, B)^2$， $g(p)$ 是凸的。

证明。我们首先引入最大距离和最小距离的平方定义：

$$\text{MaxDist}(p, M)^2 = \sum_{d=1}^{r} \max \left\{ \begin{array}{l} \left(\check{M}_d - p_d\right)^2, \\ \left(\hat{M}_d - p_d\right)^2 \end{array} \right\} \tag{24}$$

$$\text{MinDist}(p, M)^2 = \sum_{d=1}^{r} \left\{ \begin{array}{ll} \left(\check{M}_d - p_d\right)^2, & p_d < \check{M}_d \\ \left(\hat{M}_d - p_d\right)^2, & \hat{M}_d < p_d \\ 0, & \check{M}_d \leq p_d \leq \hat{M}_d \end{array} \right\} \tag{25}$$

只需证明每个维度分别是凸的，因为凸函数之和同样是凸的。每个维度 d都有一个类型为 $\mathbb{R} \to \mathbb{R}$的关联函数 h_d。令 e：$\mathbb{R} = E_d$ 以及 b：$\mathbb{R} = B_d$：

$$g(p) = \sum_{d=1}^{r} h_d(p) \tag{26}$$

$$h_d(p) = \max\left\{ \begin{array}{l} (\check{E}_d - p)^2, \\ (\hat{E}_d - p)^2 \end{array} \right\} - \left\{ \begin{array}{ll} (\check{B}_d - p)^2, & p < \check{B}_d \\ (\hat{B}_d - p)^2, & \hat{B}_d < p \\ 0, & \check{B}_d \le p \le \hat{B}_d \end{array} \right\} \tag{27}$$

ˇ ˆ ˇ ˆ 现在只需证明，对于所有界限 $[E_d, E_d]$和 $[B_d, B_d]$，$h_d(p)$ 是凸的。在此定义中，max 表达式关于中点对称。令 $\bar{E}_d = {}^{\check{}}E_d + {}^{\hat{}}_2 E_d$。

$$\max\left\{ \begin{array}{l} \left(\check{E}_d - p\right)^2, \\ \left(\hat{E}_d - p\right)^2 \end{array} \right\} = \left\{ \begin{array}{ll} \left(\hat{E}_d - p\right)^2, & p < \bar{E}_d \\ \left(\check{E}_d - p\right)^2, & \bar{E}_d \le p \end{array} \right\} \tag{28}$$

现在，我们可以将该命题重新定义为一个分段表达式：

$$h_d(p) = \left\{ \begin{array}{ll} \left(\hat{E}_d - p\right)^2, & p < \bar{E}_d \\ \left(\check{E}_d - p\right)^2, & \bar{E}_d \le p \end{array} \right\} - \left\{ \begin{array}{ll} \left(\check{B}_d - p\right)^2, & p < \check{B}_d \\ \left(\hat{B}_d - p\right)^2, & \hat{B}_d < p \\ 0, & \check{B}_d \le p \le \hat{B}_d \end{array} \right\} \tag{29}$$

该函数最多有三个分段断点： $p = \bar{E}_d$、 $p = B_d$ $p = B_d$ h_d p ˇ 和 ˆ。为证明分段函数 () 是凸函数，我们证明函数的每个连续区间都具有非负的二阶导数。这些区间在断点处相接，断点处的一阶导数会瞬间变化；我们将证明右侧导数不小于左侧导数。 $h_d(p)$ 的每种情况都具有 $(a-x)^2 - (b-x)^2$, 的形式；或者，当 ˇ $B_d \le p \le {}^{\hat{}}B_d$ 时，具有 $(a-x)^2$ 的形式。这两种形式，$\forall a,\ b \in \mathbb{R}$： $f(x) = (a-x)^2 - (b-x)^2$ 以及$\forall a$： $f(x) = (a-x)^2$,，关于 x 都是凸的，因此只需检查连接这些区间的断点：

情况： $\bar{E}_d = {}^{\check{}}B_d = \hat{B}_d$

首先，将 ˇ B_d 和 $\hat{B}_d$ 改写为 $\bar{E}_d$:

$$h_d(p) = \left\{ \begin{array}{ll} \left(\hat{E}_d - p\right)^2, & p < \bar{E}_d \\ \left(\check{E}_d - p\right)^2, & \bar{E}_d \le p \end{array} \right\} - \left\{ \begin{array}{ll} \left(\bar{E}_d - p\right)^2, & p < \bar{E}_d \\ \left(\bar{E}_d - p\right)^2, & \bar{E}_d < p \\ 0, & \bar{E}_d \le p \le \bar{E}_d \end{array} \right\} \tag{30}$$

其次，将分段表达式化简：

$$h_d(p) = \left\{ \begin{array}{ll} \left(\hat{E}_d - p\right)^2, & p < \bar{E}_d \\ \left(\check{E}_d - p\right)^2, & \bar{E}_d \le p \end{array} \right\} - \left(\bar{E}_d - p\right)^2 \tag{31}$$

$$\frac{d}{dp}\left[(\hat{E}_d - p)^2 - (\bar{E}_d - p)^2\right] = 2\bar{E}_d - 2\hat{E}_d \tag{32}$$

$$\lim_{p \to \bar{E}_d^-} 2\bar{E}_d - 2\hat{E}_d = 2\bar{E}_d - 2\hat{E}_d \tag{33}$$

x	$\lim_{p\to x^-} \frac{d}{dp} h_d(p)$	$\lim_{p\to x^+} \frac{d}{dp} h_d(p)$	asm.
情况： $\bar{E}_d < \check{B}_d < \hat{B}_d$			
$\bar{E}_d$	$2\check{B}_d - 2\hat{E}_d$	$2\check{B}_d - 2\check{E}_d$	$\check{E}_d \le \hat{E}_d$
$\check{B}_d$	$2\check{B}_d - 2\check{E}_d$	$2\check{B}_d - 2\check{E}_d$	
$\hat{B}_d$	$2\hat{B}_d - 2\check{E}_d$	$2\hat{B}_d - 2\check{E}_d$	
情况： $\check{B}_d < \bar{E}_d < \hat{B}_d$			
$\check{B}_d$	$2\check{B}_d - 2\hat{E}_d$	$2\check{B}_d - 2\hat{E}_d$	
$\bar{E}_d$	$2\bar{E}_d - 2\hat{E}_d$	$2\bar{E}_d - 2\check{E}_d$	$\check{E}_d \le \hat{E}_d$
$\hat{B}_d$	$2\hat{B}_d - 2\check{E}_d$	$2\hat{B}_d - 2\check{E}_d$	
情况： ˇ $B_d < \bar{B}_d < \bar{E}_d$			
ˇ ˇ ˆ B_d	$2B_d - 2E_d$	$2\check{B}_d - 2\hat{E}_d$	
$\hat{B}_d$	$2\hat{B}_d - 2\hat{E}_d$	$2\hat{B}_d - 2\hat{E}_d$	
$\bar{E}_d$	$2\hat{B}_d - 2\hat{E}_d$	$2\hat{B}_d - 2\check{E}_d$	$\check{E}_d \le \hat{E}_d$
情况： $\bar{E}_d < {}^{\check{}}\bar{B}_d = {}^{\check{}}\bar{B}_d$			
$\bar{E}_d$	$2\check{B}_d - 2\hat{E}_d$	$2\check{B}_d - 2\check{E}_d$	$\check{E}_d \le \hat{E}_d$
$\check{B}_d$	$2\check{B}_d - 2\check{E}_d$	$2\hat{B}_d - 2\check{E}_d$	
情况： $\bar{E}_d = \check{B}_d < \hat{B}_d$			
$\bar{E}_d$	$2\check{B}_d - 2\hat{E}_d$	$2\bar{E}_d - 2\check{E}_d$	$\check{E}_d \le \hat{E}_d$
$\hat{B}_d$	$2\hat{B}_d - 2\check{E}_d$	$2\hat{B}_d - 2\check{E}_d$	
情况： $\check{B}_d < \bar{E}_d = {}^{\hat{}}\bar{B}_d$			
$\check{B}_d$	$2\check{B}_d - 2\hat{E}_d$	$2\check{B}_d - 2\hat{E}_d$	
$\bar{E}_d$	$2\bar{E}_d - 2\hat{E}_d$	$2\hat{B}_d - 2\check{E}_d$	$\check{E}_d \le \hat{E}_d$
情况： ˇ $B_d = \hat{B}_d < \bar{E}_d$			
$\check{B}_d$	$2\check{B}_d - 2\hat{E}_d$	$2\hat{B}_d - 2\hat{E}_d$	
$\bar{E}_d$	$2\hat{B}_d - 2\hat{E}_d$	$2\hat{B}_d - 2\check{E}_d$	$\check{E}_d \le \hat{E}_d$

表 1：定理 4.4 的其余证明情况

$$\frac{d}{dp}\left[(\check{E}_d - p)^2 - (\bar{E}_d - p)^2\right] = 2\bar{E}_d - 2\check{E}_d \tag{34}$$

$$\lim_{p \to \bar{E}_d^+} 2\bar{E}_d - 2\check{E}_d = 2\bar{E}_d - 2\check{E}_d \tag{35}$$

最后，左侧导数小于或等于右侧导数：

$$2\bar{E}_d - 2\hat{E}_d \le 2\bar{E}_d - 2\check{E}_d \tag{36}$$

$$-\hat{E}_d \le -\check{E}_d \tag{37}$$

$$\check{E}_d \le \hat{E}_d \tag{38}$$

ˇ$E_d \le {}^{\hat{}}E_d$ 是一个已有假设，因此证明了这种情况。其余情况重复且缺乏启发性，因此我们在表 1 中概述。如果完成证明需要的不只是改写相等表达式，我们会列出所用的假设（asm.）。□

4.2 优化的 AllPointsCloser 算法

我们提出了一种优化算法，其运行时间为 $O(R)$，如算法 2 所示。算法 1 由于需要测试每个 2^R角点，运行时间为 $O(R \cdot 2^R)$。我们的优化代码是在原始算法 1 的基础上，将最小距离和最大距离函数内联，并搜索最为对齐的角以寻找例外情况而得到的。未优化的算法会搜索原点的一个角点，使得该点到基的最小距离小于或等于其到评估对象的最大距离。优化算法则针对每个维度独立求解，寻找哪个

算法 2 AllPointsCloser（优化版）

输入： 原点轴对齐包围盒 O，评估轴对齐包围盒 E，基准轴对齐包围盒 B

输出：若 O 中的所有点在 E 中都有最近邻，则为 true

函数 AllPointsCloser(O, E, B)
  $s \leftarrow 0$ ▷ 累加和
  从 $i \leftarrow 1$ 到 r **进行** ▷ 遍历每个维度
    $(o_{\min},\ o_{\max}) \leftarrow O[i]$
    $(e_{\min},\ e_{\max}) \leftarrow E[i]$
    $(b_{\min},\ b_{\max}) \leftarrow B[i]$

    如果 $o_{\min} < b_{\min}$ **则**
      $d_{\min B} \leftarrow (b_{\min} - o_{\min})^2$
    否则如果 $b_{\max} < o_{\min}$ **则**
      $d_{\min B} \leftarrow (b_{\max} - o_{\min})^2$
    else
      $d_{\min B} \leftarrow 0$
    结束 if

    如果 $o_{\max} < b_{\min}$ **则**
      $d_{\max B} \leftarrow (b_{\min} - o_{\max})^2$
    否则，如果 $b_{\max} < o_{\max}$ **则**
      $d_{\max B} \leftarrow (b_{\max} - o_{\max})^2$
    else
      $d_{\max B} \leftarrow 0$
    结束 if

    $d_{\min E} \leftarrow$ 最大$\left((e_{\max} - o_{\min})^2, (e_{\min} - o_{\min})^2\right)$
    $d_{\max E} \leftarrow \max\left((e_{\max} - o_{\max})^2, (e_{\min} - o_{\max})^2\right)$
    $s \leftarrow s +$ 最小$(d_{\min B} - d_{\min E},\ d_{\max B} - d_{\max E})$
  结束循环
  返回$(s > 0)$
结束函数

方向（即朝向给定维度的低界或高界）能使这两个度量之间的差值最大。随后，我们将这一搜索与在累加和中计算最大距离(c_x, E) – 最小距离(c_x, B) 合并。

形式上，优化后的检查为：

$$\textsc{AllPointsCloser}(O, E, B) \iff 0 < \sum_{d=1}^{r} \min \left\{ \begin{array}{l} \text{MinDist}^2(\check{O}_d, B_d) - \text{MaxDist}^2(\check{O}_d, E_d), \\ \text{MinDist}^2(\hat{O}_d, B_d) - \text{MaxDist}^2(\hat{O}_d, E_d) \end{array} \right\} \quad (39)$$

其中，MinDist^2 和 MaxDist^2 分别应用于每个维度的标量边界端点 $\check{}\,O_d$、$\hat{}\,O_d$ 以及维度区间 B_d、E_d。

优化的 AllPointsCloser 算法的执行时间如表2所示。代码使用 Rust 编写，并在 AMD Ryzen 3900X 上运行。对于最常用的数值类型和空间数据集维数，我们的剪枝方法执行时间在 10 纳秒以内（浮点数和 64 位以内的整数；2 维至 4 维）。

4.3 AllPointsCloser 的推广应用

必须考虑以下情况：当 k 超过一个分区的大小时、界限不是最小值时，或使用谓词导致分区中的潜在邻居数量未知时。AllPointsCloser 构成一种严格偏序关系

数值类型	2d	3d	4d	8d	16d	24d	32d
Int8	6	9	12	25	52	77	106
16 位整数	5	7	8	18	38	58	81
32 位整数	7	8	10	20	42	63	87
64 位整数	5	7	10	20	41	62	82
128 位整数	12	19	28	49	96	144	202
32 位浮点数	4	6	9	16	32	45	68
64 位浮点数	4	6	9	19	30	47	69
64 位有理数	62	97	126	244	491	710	955

表2：优化剪枝测试运行时间（纳秒）。

每个独立原点分区的界限之间的‘<’。

$$E < B \mid O \iff \forall o \in O\colon\ \forall e \in E\colon\ \forall b \in B\colon \text{dist}(o, e)\ < \text{dist}(o, b)$$

我们用它表示“相对于 O，E 比 B 更近”。证明见第 4.3.1 节。通过以这种方式对分区排序，我们可以为每个测试/原点分区构建一个训练分区之间的有向无环图。许多 k 近邻方法都隐式利用了这一性质，并采用现有的剪枝机制，例如 R树k近邻搜索。在图 3 所示的布局中，假设所有分区都只有一个点，带有 $k = 2$ 的 k 近邻查询只需检查 **P1** 和 **P2**。类似地，如果在扫描中添加额外的谓词，那么分区行数就只是一个上界，所需的分区数量也就没有上限。随后可以按照有向无环图的拓扑序搜索分区，直到到达 k。

4.3.1 偏序证明。

引理 4.5。 全点邻近性是非自反的。

$$\neg(A < A \mid O)$$

证明。我们重写定义并证明一个矛盾：

$$\forall o \in O\colon\ \forall a_1 \in A\colon\ \forall a_2 \in A\colon\ \text{dist}(o,\ a_1)\ < \text{dist}(o,\ a_2)$$

This statement is always false when $a_1 = a_2$. □

引理 4.6。 全点邻近性是传递的。

$$(A < B \mid O) \wedge (B < C \mid O) \implies (A < C \mid O)$$

证明。假设 $(A < B \mid O)$ 和 $(B < C \mid O)$。

$$\forall o \in O : \forall a \in A : \forall b \in B : \text{dist}(o, a) < \text{dist}(o, b)$$
$$\forall o \in O : \forall b \in B : \forall c \in C : \text{dist}(o, b) < \text{dist}(o, c)$$

我们要证明 $\forall o \in O\colon\ \forall a \in A\colon\ \forall c \in C\colon \text{dist}(o, a)\ < \text{dist}(o, c)$。

任取 $o \in O$、$a \in A$ 和 $c \in C$。由于 B 是第二个假设中用于比较的集合，令我们任取一个 $b \in B$。于是根据第一个假设，我们有

$$\text{dist}(o, a) < \text{dist}(o, b),$$

根据第二个假设，我们有

$$\text{dist}(o, b) < \text{dist}(o, c).$$

根据实数通常序的传递性，

$$\text{dist}(o, a) < \text{dist}(o, c).$$

由于 o、a 和 c 是从全称量词中任意选取的，因此这表明（$A < C \mid O$）成立，从而证明了传递性。□

5 结论

数据仓库正逐渐成为查询大型数据集的主流架构。它们不再维护持久化索引，而是利用列式分区分布统计来加速查找和连接。全 k 近邻连接可以从仅基于分区边界的剪枝中受益；然而，现有剪枝方法是为索引内剪枝而设计的，因此并不适用于这一场景。

我们提出了 AllPointsCloser 算法，用于判断：对于原点分区中的所有点，一个分区中的每个点是否都比另一个分区中的每个点更近。我们证明了三界限测试是最优的；如果一个分区始终更近，AllPointsCloser 就能证明这一点。此外，AllPointsCloser 不受 MinMaxDist [13] 或 NXNDist [4], 的限制；当 $k > 1$、界限并非最小界限，或查询中使用了额外谓词时，这些限制并不适用。因此，无论后续采用哪种连接算法，AllPointsCloser 都能为分区数据集上的空间 AkNN 连接带来一种简单、最优且实用的改进。

可用性。Rust 实现和 Lean 4 形式化证明可在 https://github.com/dominikWin/apc-aknn 以及 arXiv 附件文件中获取。

参考文献

[1] H3, Uber 的六边形层次空间索引。https://www.uber.com/ en-ID/blog/h3/ 2018.

[2] Baiyou Qiao、Junhai Zhu、Muchuan Shen 和 Yang Chen。2015。基于边界过滤的空间连接查询处理优化算法。载于*2015*年第*12*届模糊系统与知识发现国际会议（*FSKD*）。IEEE，张家界，中国，1764 1769 10 1109 2015 7382214 –。doi: 。/

[3] FSKD。。Georgios Chatzimilioudis、Constantinos Costa、Demetrios Zeinalipour-Yazti、Wang-Chien Lee 和 Evaggelia Pitoura。2016。全k近邻查询的分布式内存处理。*IEEE Transactions on Knowledge and Data Engineering* 28, 4 2016 925 938 10 1109 2015 2503768 （4月），–。doi: 。/

[4] TKDE。。Yun Chen 和 Jignesh M. Patel。。全最近邻查询的高效评估。载于*2007*年*IEEE*第*23*届数据工程国际会议。1056–1065。doi:10.1109/ICDE.2007.368964[5] Antonio Corral、Yannis Manolopoulos、Yannis Theodoridis 和 Michael Vassilakopoulos。2000。空间数据库中的最近对查询。载于*2000*年*ACM SIGMOD*数据管理国际会议论文集（美国得克萨斯州达拉斯）（*SIGMOD '00*）。美国计算机协会，纽约州纽约市，189–200。doi:10.1145/342009.335414[6] Benoit Dageville、Thierry Cruanes、Marcin Zukowski、Vadim Antonov、Artin Avanes、Jon Bock、Jonathan Claybaugh、Daniel Engovatov、Martin Hentschel、Jiansheng Huang、Allison W. Lee、Ashish Motivala、Abdul Q. Munir、Steven Pelley、Peter Povinec、Greg Rahn、Spyridon Triantafyllis 和 Philipp Unterbrunner。2016。Snowflake弹性数据仓库。载于*2016*年数据管理国际会议论文集。ACM，美国加利福尼亚州旧金山，215–226。doi:10.1145/2882903.2903741[7] Jeffrey Dean 和 Sanjay Ghemawat。2008。MapReduce：简化大型集群上的数据处理。*ACM*通讯 51, 1 （2008 年 1 月），107–113。doi:10.1145/1327452. 1327492[8] Ahmed Eldawy 和 Mohamed F. Mokbel。2016。大规模空间数据时代：综述。基础与趋势® 数据库 6, 3-4（2016），163–273。doi:10. 1561/1900000054[9] Michael Gowanlock。2021。混合KNN连接：利用CPU和GPU架构特性的并行最近邻搜索。并行与分布式计算期刊 149 2021 119 137 10 1016 2020 11 004 （3月），–。doi: 。/j.jpdc.。。

[10] Wei Lu、Yanyan Shen、Su Chen 和 Beng Chin Ooi。。使用MapReduce高效处理k近邻连接。*VLDB*基金会会刊 5, 10 2012 1016 1027 10 14778 2336664 2336674 （6月），–。doi: 。/。。Jesus Maillo 、 Sergio Ramírez 、 Isaac Triguero 和 Francisco Herrera。2017。kNN-IS：一种面向大数据的、基于Spark迭代式设计的k近邻分类器。知识库系统 117 2017 3 15 10 1016 2016 06 012 （2月），–。

[11] doi: 。/j.knosys.。。Varun Pandey、Andreas Kipf、Thomas Neumann 和 Alfons Kemper。。现代空间分析系统的表现如何？*VLDB*基金会会刊 11, 11 2018 1661 1673 10 14778 3236187 3236213 （7月），–。doi: 。/。。

[12] Nick Roussopoulos、Stephen Kelley 和 Frédéric Vincent。。最近邻查询。载于*1995*年*ACM SIGMOD*数据管理国际会议论文集（*SIGMOD '95*）。美国计算机协会，纽约纽约州，NY，美国，71–79。doi:10.1145/223784.223794

[13] Jagan Sankaranarayanan、Hanan Samet 和 Amitabh Varshney。2007。面向大型点云应用的快速全近邻算法。*Computers & Graphics* 31, 2 2007 157 174 10 1016 2006 11 011 (4月)，–。

[14] doi: 。/j.cag. . . Boyi Shangguan、Peng Yue、Zhaoyan Wu 和 Liangcun Jiang。. 使用Apache Spark 处理大规模空间数据。载于*2017*年第*6*届农业地理信息学国际会议。IEEE，美国弗吉尼亚州费尔法克斯，1–4。doi:10.1109/Agro-Geoinformatics. 2017.8047039[15] Mike Stonebraker、Daniel J. Abadi、Adam Batkin、Xuedong Chen、Mitch Cherni-ack、Miguel Ferreira、Edmond Lau、Amerson Lin、Sam Madden、Elizabeth O'Neil、Pat O'Neil、Alex Rasin、Nga Tran 和 Stan Zdonik。2018。*C-store*：一种面向列的 *DBMS*。美国计算机协会和 Morgan & Claypool，491 518 10 1145 3226595 3226638 –。

[16] https://doi-org.proxy.lib.ohio-state.edu/。/。Mingjie Tang、Yongyang Yu、Qutaibah M. Malluhi、Mourad Ouzzani 和 Walid G. Aref。2016。LocationSpark：一种面向大规模空间数据的分布式内存数据管理系统。*VLDB*基金会会刊 9，13（2016年9月），1565–1568。doi:10.14778/3007263.3007310[17] Polychronis Velentzas、Michael Vassilakopoulos 和 Antonio Corral。2020。基于GPU的内存中 k 近邻查询处理：。载于第*6*届地理信息系统理论、应用与管理国际会议论文集。SCITEPRESS - 科学与技术出版公司，捷克共和国布拉格，310 317 10 5220 0009781903100317 –。

[18] doi: 。/ Polychronis Velentzas、Michael Vassilakopoulos 和 Antonio Corral。2021。用于处理磁盘驻留数据中 k 近邻查询的 GPU 算法。载于模型与数据工程，Christian Attiogbé 和 Sadok Ben Yahia（编）。第12732卷。Springer国际出版公司，瑞士查姆，264–278。doi:10. 1007/978-3-030-78428-7_21 丛书名称：计算机科学讲义。[19] Tin Vu、Ahmed Eldawy、Vagelis Hristidis 和 Vassilis Tsotras。2021。通过增量分区实现高效的空间数据分析。*VLDB*基金会会刊 15, 3 2021 713 726 10 14778 3494124 3494150 (11月)，–。doi: 。/。

[20] Chenyi Xia、Hongjun Lu、Beng Chin Ooi 和 Jing Hu。. Gorder：一种高效的 k近邻连接处理方法。载于第三十届超大型数据库国际会议论文集——第*30*卷。756–767。[21] Dong Xie、Feifei Li、Bin Yao、Gefei Li、Liang Zhou 和 Minyi Guo。2016。Simba：高效的内存空间分析。载于*2016*年数据管理国际会议论文集。美国计算机协会，美国加利福尼亚州旧金山，1071–1085。doi:10.1145/2882903.2915237[22] Jia Yu、Jinxuan Wu 和 Mohamed Sarwat。2015。GeoSpark：用于处理大规模空间数据的集群计算框架。载于第*23*届*SIGSPATIAL*地理信息系统进展国际会议论文集。美国计算机协会，美国华盛顿州西雅图，1 4 10 1145 2820783 2820860 –。doi: 。/。

[23] Chi Zhang、Feifei Li 和 Jeffrey Jestes。. 使用 MapReduce 对大数据进行高效并行 k 近邻连接。载于第*15*届扩展数据库技术国际会议论文集——*EDBT '12*。ACM出版社，德国柏林，38。doi:10.1145/2247596.2247602[24] Mohamed Ziauddin、Andrew Witkowski、You Jung Kim、Dmitry Potapov、Janaki Lahorani 和 Murali Krishna。2017。基于维度的数据聚类与区域映射。*VLDB*基金会会刊 10，12（2017），1622–1633。[25] Erlend Åmdal。2020。*GPU*上的 *top-k* 空间连接。硕士论文。挪威科技大学。