基于分布感知空间枢轴的学习型索引

孟宇航[1,2]　张丰[1,2]　唐鹏[1,2]　胡宇森[1,2]

1. 浙江大学 地球科学学院，浙江 杭州 310058； 2. 浙江大学 浙江省全省资源与环境信息系统重点实验室， 浙江 杭州 310058

摘　要：空间学习型索引通过学习数据的空间分布特征来构建索引。与传统空间索引相比，它显著降低了存储开销和查询成本。然而在高维复杂分布场景中，现有的学习型索引难以实现空间结构效率与查询泛化性能的全局最优，同时缺乏高效的动态更新机制。为了解决这些问题，我们提出了一种基于分布感知空间枢轴的学习型索引 Distribution-Aware Spatial Pivot Learned Index (DSPI)。空间枢轴在索引中扮演关键角色，其核心意义在于通过少量代表性点覆盖数据的全局分布特征。DSPI 通过自注意力机制捕捉数据的空间分布特性，并采用 Transformer 神经网络进行枢轴分布与剪枝效率的联合优化，实现端到端的查询剪枝。在给定的枢轴搜索空间中，DSPI 通过基于二阶导数和 AIC 优化的轻量化函数映射模型，精确定位查询结果。基于 DSPI，我们还设计了基于编码剪枝的范围查询和自适应搜索半径的 k 近邻查询算法。在真实空间数据集和合成数据集上的实验结果表明，与现有空间学习索引算法相比，DSPI 在存储和查询性能上分别提升了 1.4 至 20.3 倍和 1.1 至 10.2 倍。在高维和大规模查询场景中，DSPI 的查询性能进一步提升了 1.5 至 15.2 倍。我们还为 DSPI 设计了支持数据插入和删除的算法，通过缓冲区减少重训练开销。实验结果表明，在动态更新环境中，DSPI 的检索性能表现稳定。

关键词：空间学习型索引；机器学习；空间枢轴；分布感知；自注意力机制

Distribution-aware Spatial Pivot Learned Index

Abstract: Spatial learned indexes construct indexes by learning the spatial distribution features of data. Compared to traditional spatial indexes, they significantly reduce storage overhead and query costs. However, in high-dimensional and complex distribution scenarios, existing learned indexes struggle to achieve a global optimum in terms of spatial structure efficiency and query generalization performance, while also lacking efficient dynamic update mechanisms. To address these challenges, we propose a Distribution-aware Spatial Pivot Learned Index (DSPI). Spatial pivots play a key role in the index, as they capture the global distribution characteristics of the data using a few representative points. DSPI uses a self-attention mechanism to capture the spatial distribution properties of the data and employs a Transformer neural network for training to optimize pivot distribution fitting and pruning efficiency, enabling end-to-end query pruning. In a given pivot search space, DSPI precisely locates query results using a lightweight function mapping model optimized by second-order derivatives and AIC. Based on DSPI, we also design range query and kNN query algorithms with adaptive search radius, based on encoding pruning. Experimental results on real-world and synthetic datasets show that, compared to existing spatial learned indexing algorithms, DSPI improves storage and query performance by 1.4 to 20.3 times and 1.1 to 10.2 times, respectively. In high-dimensional and large-scale query scenarios, DSPI query performance further improves by 1.5 to 15.2 times. We also design an algorithm for DSPI supporting data insertion and deletion, which reduces retraining overhead through buffering. Experimental results indicate that DSPI retrieval performance remains stable in dynamic update environments.

Key words: Spatial learned index; Machine learning; Spatial pivot; Distribution-Aware; Self-Attention mechanism

基金项目：国家自然科学基金面上项目：大规模高维地理流数据“流批一体”存储、计算与挖掘方法（42271466）

第一作者：孟宇航，博士在读，研究方向为空间数据库。12438034@zju.edu.cn

通信作者：张丰，博士，教授。zfcarnation@zju.edu.cn

空间数据集是包含空间参考信息的多维数据集，广泛应用于地球科学研究及其他领域，能够记录来自不同来源的观测数据。在处理和分析这些空间数据集时，必须考虑其高维特性和复杂的空间关联性，这对数据存储结构和查询算法提出了挑战。传统的空间索引方法通过空间分割来处理高维数据，但随着数据维度和数据量的增加，面临“维度诅咒”问题，在构建时成本过高并且搜索性能表现不佳。近年来，人工智能与数据库的结合日益紧密，学习型索引成为传统索引的高性能替代方案。Kraska 等[1]出了机器学习模型来替代传统的索引结构的递归模型索引。Tian 等[2]进而针对高维数据的相似性查询问题提出了度量空间学习型索引，通过数据聚类和基于枢轴的数据转换，有效降低了存储和查询成本。度量空间能够容纳各种类型的空间数据，只要这些数据可以配备满足三角不等式的距离度量，是处理大规模高维空间数据集的理想解决方案[3]。然而，目前基于枢轴的度量空间学习型索引采用数据聚类和枢轴选择解耦的架构，难以适应复杂的空间数据分布。此外，现有索引无法根据数据和枢轴的分布优化查询过程，缺乏高效的自适应空间查询策略。

为了解决现有学习型索引结构和性能上问题，本文提出了基于分布感知空间枢轴的学习型索引（Distribution-Aware Spatial Pivot Learned Index，DSPI）。DSPI 采用基于自注意力机制的 Transformer 神经网络，在高维空间中基于数据分布和剪枝效率进行枢轴选择，并在枢轴定义的度量空间中实现端到端的查询剪枝。DSPI 还设计了配套的自适应查询和更新方案，以提高索引查询性能和适应性。

1 相关研究现状

空间学习型索引利用机器学习方法来分析数据的分布及查询负载的特性，并通过拟合数据分布来直接构建查找函数，优化了传统的间接式索引查询方法[4]。这种从索引值到物理存储位置映射的函数方法减少了索引占用的空间，并且提升了查询速度。Ferragina 等[5]在理论上证实了在理想情况下，学习型索引模型的大小主要与其结构相关，空间复杂度为 $O(1)$；学习型索引的检索时间主要受模型推理预测误差的影响，时间复杂度为 $O(\log \mathrm{err})$。因模型无法精确预测数据分布，需要在预测的误差范围内进行进一步的精确搜索，通常通过指数搜索和二分搜索来定位最终的数据位置。

在索引结构方面，空间学习型索引为了简化数据分布，存在不同的优化方向，比如基于维度选择[6]、基于网格分区[7-8]、基于维度映射[2,9-11]等。其中，基于维度选择的方法高度依赖于数据分布特性，牺牲了一定的灵活性；基于网格分区的方法虽然在扫描开销方面具有优势，但需要付出昂贵的检查过程和索引构建时间；基于维度映射的学习型索引通过映射函数将多维数据映射到统一维度，重构数据并规划检索策略，从而有效简化了索引结构，提高了查询效率，但对映射方法的要求较高。目前的维度映射方法主要包括利用空间填充曲线[9-10]和 iDistance[12]等。需要注意的是，在低维度中空间填充曲线方法相对易于理解；但在高维空间中，空间填充曲线可能导致数据点在曲线上的排序失去直观的空间关系，从而增加索引构建和查询的复杂度。iDistance 是一种基于度量空间的降维方法，通过在高维空间中进行聚类，并在基于线性距离的度量空间中进行排序，有效地压缩并保留了高维空间的信息。它基于数据点与某个参考点（通常是簇中的一个点）之间的距离来构建索引，通过计算和索引这些距离，iDistance 方法能够快速定位和检索与查询条件相匹配的数据点。然而，在选择参考点和聚类中心时，iDistance 方法通常依赖预设的策略，这可能导致在数据分布发生变化时，索引性能下降。

在空间索引的性能优化领域，如何通过索引有效执行范围查询、k 近邻查询并进行及时的更新是核心任务。现有的空间索引的范围查询算法比较成熟，通过包围盒过滤[13-14]、网格过滤[7-8]、三角不等式[2,11]等方法进行范围查询剪枝，并在结果集中进行精确查询。对于空间学习型索引，k 近邻查询一般采用迭代范围查询的方法，从一个初始查询距离开始，在找到的结果集未达到所需数量时，持续增加查询距离，直至收集到足够的数据。初始查询半径的选择可以基于精选的固定值[2]、与查询点最近的数据点距离[11]，或通过线性插值预测[8]。这些方法虽然提高了 k 近邻查询的效率，但仍面临自定义超参数不准确或数据分布不均等问题。现有的空间学习型索引更新方法包括直接更新[2,13]、缓冲区更新[8,10]、增量更新[5,15]等。这些方法多偏重于单一优化目标，往往无法兼顾查询性能与更新效率的平衡，导致在动态、高维数据场景下难以满足实际需求。

2 DSPI

2.1 DSPI 概述

给定一个包含 n 个对象的 d 维数据集 P 和一个查询点 q。范围查询问题旨在找到查询点 q 所在的空间区域内，满足与查询点 q 的距离不超过 r 的所有数据点，即找到一个子集 $R(q)\subseteq P$，对于任意 $x\in R(q)$，都满足 $dis(x,q)\le r$。k 近邻问题旨在找到查询点 q 所在的空间区域内前 k 个最近邻点，即找到一个子集 $R(q)\subseteq P$，满足 $|R(q)|=k$，对于任意 $x\in R(q)$ 和 $y\in P-R(q)$，都满足 $dis(x,q)\le dis(y,q)$。通常，经典树索引的查询可以分为以下两个步骤：

1) 自顶向下遍历索引结构并返回一组可能满足范围查询和 k 近邻条件的候选叶节点。

2) 计算所有候选节点中每个点到 q 的距离，并检索满足范围查询和 k 近邻条件的数据点。

DSPI 的思想是通过基于空间枢轴的剪枝模型和数据拟合模型来替代这两个步骤，整体工作流程如图 1 所示。

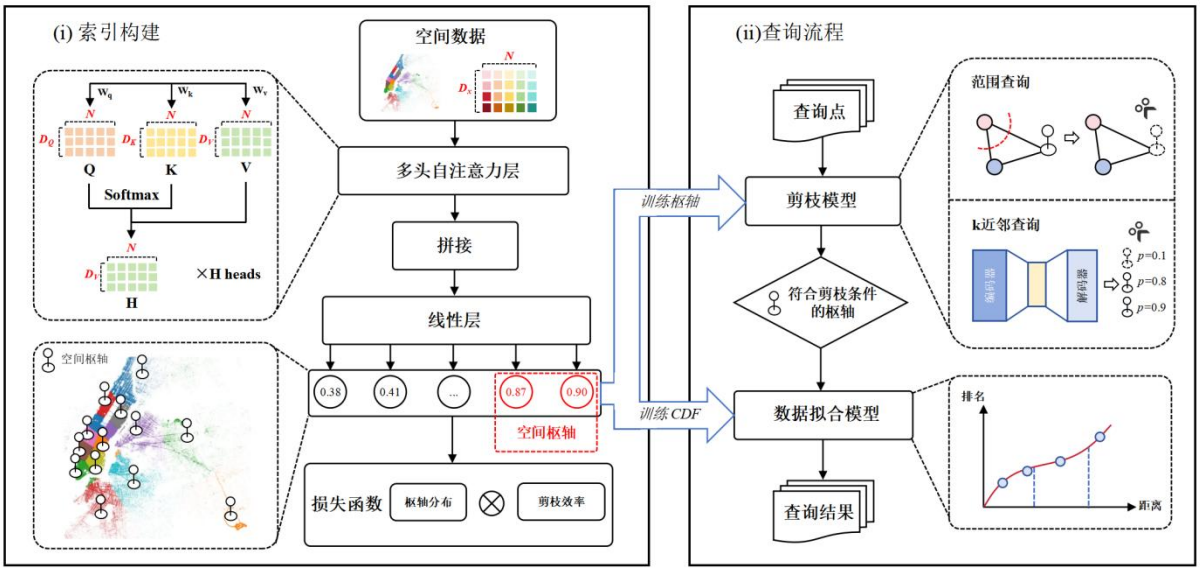


图 1　DSPI 工作流程

Fig. 1 DSPI workflow

枢轴是在数据结构或算法中用于划分数据集或组织数据的关键参考点[16]。DSPI 通过基于自注意力机制的 Transformer 神经网络评估各数据点的代表性，从中选择能有效覆盖全局数据分布且具备良好剪枝性能的空间枢轴集 $O \subseteq P$ 。在训练阶段，DSPI 利用生成的空间枢轴和对应的数据样本，训练剪枝模型 $f_{pruning}$ 和分布拟合模型 $f_{distribution}$ 。剪枝模型用于初步过滤候选数据点；分布拟合模型通过学习每个枢轴范围内的数据分布，支持对查询空间进行准确预测。DSPI 的查询流程分为两个步骤。对于查询点 q，DSPI 通过剪枝模型 $f_{pruning}$ 基于空间枢轴的距离关系对查询空间进行初步过滤，缩小待检索的候选枢轴集。随后，利用分布拟合模型 $f_{distribution}$ 对符合条件的数据点进行预测和筛选，最终获得查询结果。

2.2 枢轴选择

基于枢轴的方法的空间索引方法一般先对数据进行聚类，然后把簇中每个对象的预先计算的距离存储到一组簇枢轴上，最后利用这些距离在搜索过程中进行剪枝。这种方法存在两个问题：1）传统聚类方法在空间意义和效率方面表现不佳。例如，K-Center[17]等基于中心点的聚类方法在簇内紧密性方面表现较差，忽视了簇内数据点的分布和密度；DBSCAN[18]等基于密度的聚类方法方法对参数的选择较为敏感，且难以控制簇的大小，在负载均衡和重叠区域处理上存在显著不足。2）在传统的枢轴选择方法中，簇级的优化与枢轴的选择往往是相互独立的，未能充分利用簇划分的结构性信息。如 LIMS 使用 K-Center 算法聚类，采用最远距离方法（Farthest-First Traversal, FFT）[19]在簇中进行枢轴选择。这种方法无法在全局范围内优化查询效率，当查询点恰好位于不同簇之间的空隙处时，枢轴点的空间意义将会减弱。

为克服传统聚类方法在空间感知与效率上的缺陷，DSPI 提出一种基于 Transformer 架构[20]的枢轴选择模型。该模型摒弃了预设聚类结构的约束，通过自注意力机制动态捕捉高维数据点间的全局依赖关系，从而生成具有空间代表性的枢轴集。具体而言，给定 d 维空间数据集 $P = \{p_1, p_2, ..., p_n\}$，模型首先对每个数据点 p_i 进行线性投影，生成对应的查询向量 $Q_i = W_Q \cdot p_i$ 、键向量 $K_i = W_K \cdot p_i$ 及值向量 $V_i = W_V \cdot p_i$，其中 $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$ 为可训练参数矩阵，d_Q, d_K, d_V 为查询、键和值向量的维度，这里三个维度设置为相等。通过缩放点积计算点对间注意力权重：

$$\alpha_{ij} = \frac{Q_i \cdot K_j}{\sqrt{d_k}} \qquad (1)$$

权重随后通过 Softmax 函数归一化，并与值向量聚合，以构建空间感知的表示：

$$\text{head}_k = \text{Attention}(Q, K, V) = \sum_{j=1}^{n} \text{Softmax}(\alpha_{ij}) V_j \qquad (2)$$

为了增强模型对高维异构子空间（如方向、尺度、密度分布）的感知，DSPI 采用多头注意力机制并行学习互补的模式，异构特征通过连接和线性投影融合：

$$\text{MHA}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h) W_O \qquad (3)$$

其中 $W_O \in \mathbb{R}^{hd_v \times d}$ 表示输出投影矩阵，它将 $h \cdot d_v$ 维度的连接向量压缩回原始维度 d，同时保留关键的空间信息。

基于注意力机制，模型实施三阶段枢轴选择策略：首先，直接从多头注意力（MHA）机制的输出中计算每个点的得分，其中得分 s_i 通过一个线性层计算，得分公式为 $s_i = H_i \cdot W + b$ ，其中 $H_i \in \mathbb{R}^{1 \times d_{\text{MHA}}}$ 是每个点的最终表示，$W \in \mathbb{R}^{d_{\text{MHA}} \times 1}$ 是线性层的权重矩阵。然后按得分降序选择前 m 个候选枢轴。

训练目标通过组合损失函数来优化枢轴分布和剪枝效率：

$$\mathcal{L} = \lambda_1 \cdot \mathcal{L}_{\text{cover}} + \lambda_2 \cdot \mathcal{L}_{\text{prune}} \qquad (4)$$

其中，$L_{distribution}$ 表示分布拟合损失，旨在最大化所有枢轴点之间的最小距离，并最小化枢轴点与数据点之间的距离，以确保枢轴点在数据空间中均匀分布，并充分覆盖整个数据分布区域，避免枢轴点过度集中或过于分散；$L_{pruning}$ 表示查询剪枝损失，旨在通过减少查询时所需考虑的候选集大小，从而提高查询效率。

2.3 剪枝模型

2.3.1 范围查询剪枝

在范围查询剪枝过程中，DSPI 基于枢轴点的空间覆盖特性，通过动态分析枢轴点 O_i（覆盖半径为 r_O）与查询点 q（搜索半径为 r_q）之间的几何关系实现高效过滤。设两者欧氏距离为 d，DSPI 依据几何约束规则执行剪枝：当 $d \geq r_O + r_q$ 时，判定 O_i 的覆盖区域与查询区域完全无交集，直接剪除该枢轴点对应的数据簇；当 $d < r_O + r_q$ 时，通过三角不等式推导枢轴的有效距离区间为 $[\max(0, d - r_q), r_O]$，仅保留该区间内的候选数据以压缩计算规模。该方法的核心在于单次距离计算（时间复杂度为 $O(1)$）与几何驱动的自适应剪枝策略，在避免结果遗漏的前提下，提升效率与精度。

2.3.2 k 近邻查询剪枝

在 k 近邻查询剪枝中，DSPI 的优化策略与范围查询存在本质差异。范围查询通过预设半径 r_q 明确定义搜索范围，其剪枝依赖于三角不等式导出的静态几何边界，边界条件可通过单次距离计算直接判定。而 k 近邻查询的目标是动态寻找距离查询点最近的 k 个点，其有效搜索半径无法预先确定，且结果受数据分布密度影响显著。若直接应用三角不等式，仅能基于当前候选点距离生成保守估计区间，难以避免因初始半径选择偏差导致的多次迭代搜索。为此，DSPI 引入邻域学习方法[21]，将 k 近邻问题转化为对枢轴匹配概率的排序问题。具体而言，给定查询点 q，首先通过公式(3)的多头注意力编码后，生成全局感知特征 $H_q = \text{MHA}(Q_q, K_O, V_O)$，其中 K_O 和 V_O 为枢轴集 O 的键/值向量投影。邻域概率预测模块通过式(4)的融合特征 H_q，计算每个枢轴 O_i 的匹配概率，即邻域分数 ρ：

$$\rho = \text{Prob}(O_i \mid q) = \text{Softmax}(H_q \cdot W_{knn})_i \qquad (5)$$

其中 $W_{knn} \in \mathbb{R}^{d \times m}$ 为可训练参数矩阵，m 为枢轴数量。该概率值综合了查询点与枢轴的空间关联性（由自注意力权重 α_{ij} 隐式编码）及全局分布特性（由多头注意力子空间投影所捕获）。模型通过交叉熵损失函数直接优化概率预测精度。系统依据概率分布对枢轴排序，优先检索高概率枢轴内的候选数据，从而解决 k 近邻查询由于半径不确定性和数据分布不均导致的检索偏差问题。

2.4 数据拟合模型

传统树状索引难以通过单一模型准确拟合复杂的数据分布。为了克服这一问题，学习型索引尝试采用线性或非线性模型来拟合累积分布函数（Cumulative Distribution Function，CDF），从而提升预测性能。然而，线性模型（如高次函数）在处理数据分布中变化较大的部分时效果不佳，导致拟合精度低，且增加了误差修正所需的时间。而非线性模型（如神经网络）虽然能够捕捉复杂的模式，但在大量数据拟合任务中使用时会导致较长的计算时间和较高的资源消耗。

为解决这些不足，DSPI 采用基于二阶导数和赤池信息准则（AIC）[22]的样条函数优化算法。如图 2 所示，算法首先在数据域内建立键值-位置映射（蓝点），通过评估数据段并计算每个分割点组合

的 AIC 值，优化分割点的选择。其中公式 $AIC = 2N + K\ln(RSS/K)$ 用于权衡模型复杂度（此处 N=2）和拟合准确性。在确定分割点数量后，DSPI 通过计算数据点与其相邻点之间的二阶导数选择并排序分割点，分析曲率变化并识别数据中的非线性特征。为了防止过度分割并确保分割点之间的平滑过渡，算法应用邻域抑制以惩罚毗邻的分割点。最终，通过最小二乘法拟合优化后的断点，得到最佳拟合曲线。与高阶多项式拟合或神经网络不同，分段线性模型需要更少的参数和操作，从而加快了推理速度。通过仅在需要的地方进行数据分割，DSPI 最小化了不必要的复杂性，同时保持了准确性，减少了内存使用和计算开销。

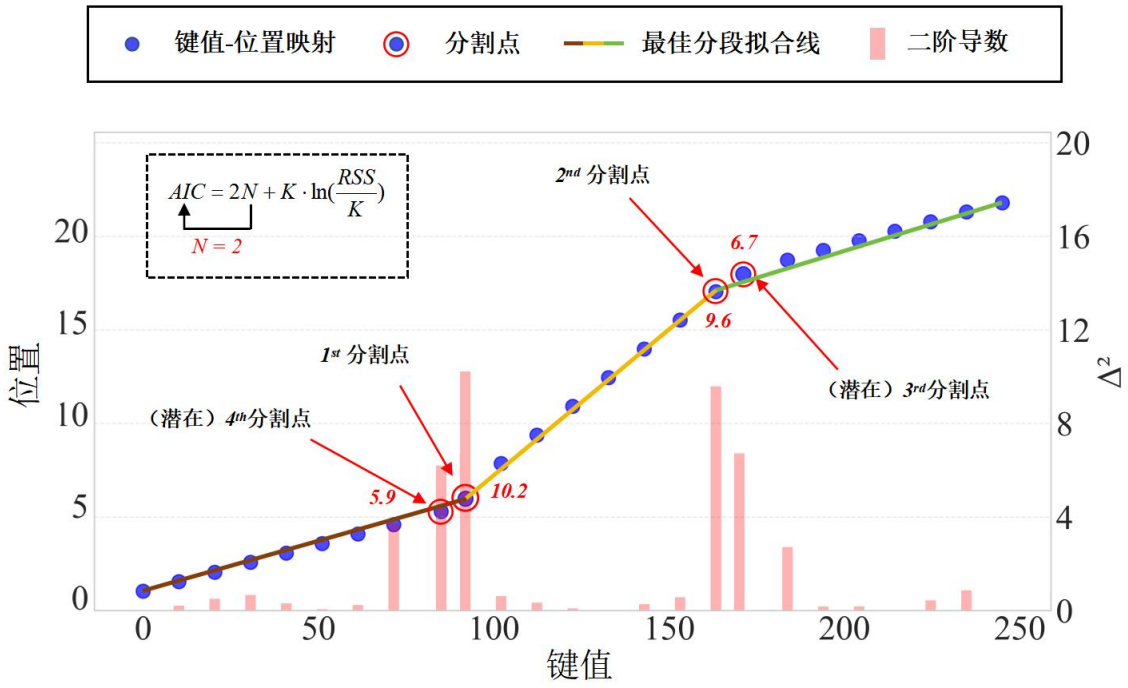


图 2 基于二阶导数和赤池信息准则的样条模型

Fig. 2 Spline Model Training with second derivatives and AIC

3 DSPI 查询与更新算法设计

基于 DSPI 的查询与更新通过调用预先构建的剪枝模型和数据拟合模型进行数据处理。

3.1 范围查询原理

范围查询的目标是从数据集 P 中检索给定查询点 q 周围半径为 r 内的所有对象。图 3 概述了这一过程，包含四个子步骤。

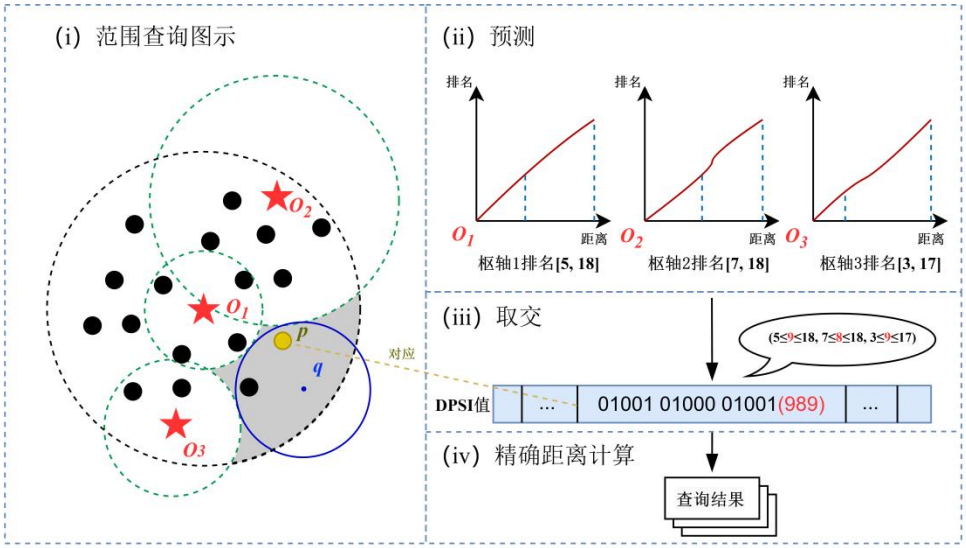


图 3 基于 DSPI 的范围查询示例

Fig. 3 An example of range query based on DSPI

3.1.1 三角不等式剪枝

在范围查询过程中，DSPI 通过三角不等式对枢轴进行初步剪枝，以排除不可能包含查询结果的枢轴，从而提高查询效率。详细的剪枝过程已在前文中描述，此处不再赘述。

3.1.2 排名预测

在初步剪枝后，需要进一步确定范围查询在每个枢轴中的覆盖区域。DSPI 为每个枢轴 O_i 构建了如 2.4 中描述的距离-排名累积分布函数模型（用 CDF 表示），通过查询范围和参考点的相对距离来预测排名：$[rank_{low}, rank_{high}] \leftarrow \mathrm{CDF}(O_i, q, r)$ 。然后，DSPI 确定每个枢轴管理的区域中符合该排名范围的候选数据集：$candidate\_set \leftarrow \{p \in \text{区域 of } O_i \mid rank_{low} \le rank(p) \le rank_{high}\}$ 。通过排名预测，DSPI 能够为每个枢轴确定一个候选数据集 $candidate\_set$ 以供进一步的筛选。

3.1.3 候选集取交

在这一阶段，系统从多个枢轴对应的候选数据集中取交集，从而减少精炼过程的计算负担。每个数据点都有一个唯一的排名编号，称为 DSPI 值。DSPI 值采用多维编码方案，将三个相关枢轴的相对排名信息压缩并记录在一个编号中。例如，DSPI 值可以是类似“010010100001001”的二进制编码。DSPI 值可以通过以下方式进行比较：$DPSI_{intersection}(p) = (\cap_{i=1}^{n}[rank_{low_i}, rank_{high_i}])$ 。通过比较多个枢轴的排名范围，DSPI 能够找到在所有候选数据集 $candidate\_set$ 中都符合条件的数据点 p，并将这些点加入最终集合 $final\_candidates$ 。

3.1.4 精确距离计算

最后，系统使用精确距离计算 $r \ge d(p,q) = \sqrt{\sum_{i=1}^{d}(p_i - q_i)^2}$ 来判断最终集合中的查询点是否在查询范围 r 内，最后返回包含所有符合条件的数据点的查询结果集 R。

3.2 k 近邻查询原理

k 近邻查询的目标是从数据集 P 中识别并返回给定查询点 q 周围指定数量 k 的最近对象。图 4 概述了这一过程，包含三个子步骤。

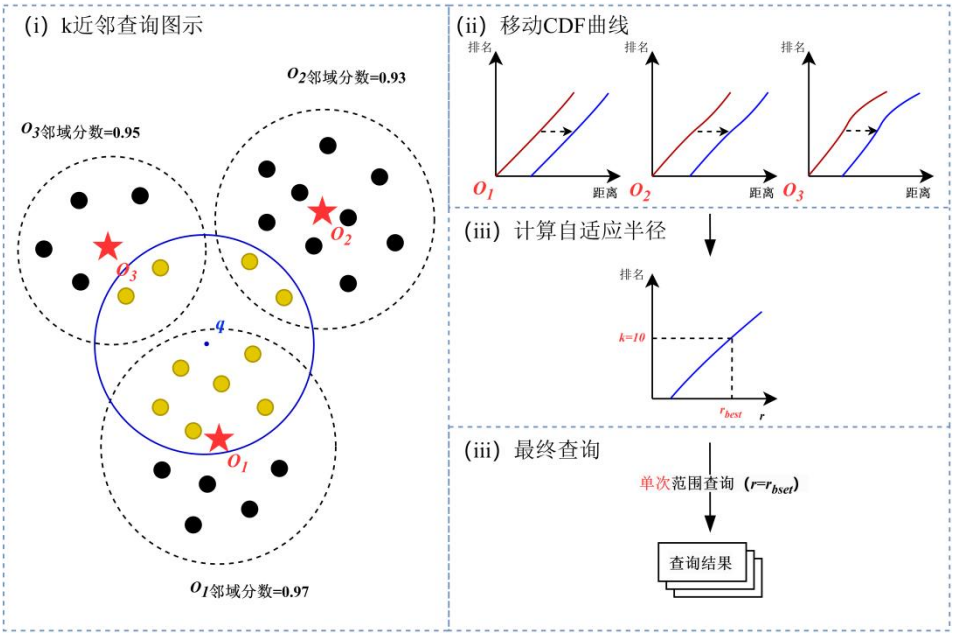


图 4 基于 DSPI 的 k 近邻查询示例

Fig. 4 An example of kNN query based on DSPI

3.2.1 邻域学习剪枝

在 DSPI 中，邻域学习模型 $f(p)$ 用于计算每个枢轴 O_i 的邻域分数 ρ_i，该分数代表了该枢轴近邻的概率。每个枢轴的邻域分数是通过评估查询点与枢轴的相对距离以及周围数据点的分布来确定的，具体原理如 2.3.2 所述。DSPI 选择具有最高邻域分数的枢轴集来进一步执行查询，从而减少计算负担。DSPI 还会统计这些枢轴中的点数，直到满足所需数量的近邻数 k。

3.2.2 自适应查询半径优化

为了在每个枢轴中准确高效地选择最近邻对象，DSPI 引入了一种基于累积分布函数的自适应查询半径优化策略取代了传统的迭代调整。具体而言，DSPI 首先根据查询点 p 与筛选出的各枢轴点 O_i 的距离，调整每个相关枢轴的 CDF。通过对所有枢轴的 CDF 进行平移 $d_{qO_i} = \| q - O_i \|$，DSPI 累加所有枢轴的 CDF 曲线，形成全局 CDF 曲线。根据所需的近邻数 k 求解全局 CDF 的逆，即 $r_{best} = \mathrm{CDF}^{-1}(k)$，$r_{best}$ 为单次范围查询覆盖所有符合条件的查询点的近似最佳半径。

3.2.3 精确查询

基于近似最佳查询半径 r_{best}，并在此半径内执行单次范围查询以筛选出前 k 个最近邻点，具体算法同 3.1。

3.2 更新原理

DSPI 支持动态的数据更新操作，包括数据点的插入与删除。系统采用增量缓冲机制以避免频繁重建索引。每个枢轴维护一个独立的缓冲区 B_i，用于临时存储新增或删除的数据点。当某一枢轴的缓冲区中累积的更新点数量超过预设阈值 τ，即 $|B_i| \geq \tau$，系统将触发该枢轴的重训练过程，以更新其结构和 DSPI 编码。

在数据插入过程中，系统首先判断新数据点 p_x 是否已存在于任一枢轴中。如果不存在，则通过邻域学习模型 $f(p_x)$ 预测该数据点应插入的枢轴。神经网络模型根据输入特征预测每个枢轴的归属概率 ρ_i，选择最大概率所对应的枢轴进行插入。新数据点插入指定枢轴后，通过拟合函数为其分配 DSPI 值，并将枢轴状态标记为“未排序”，以便在需要时进行重新排序和优化，同时更新枢轴中维护的距离数组。在数据删除过程中，系统通过运行点查询找到需要删除的数据点，将其标记为“已删除”。

在查询过程中，除了返回查询结果外，DSPI 还会根据当前查询的上下文自动更新枢轴的状态和标记，确保数据的实时性和准确性。

4 实验与结果分析

4.1 实验设置

DSPI 使用 Python 实现。所有实验均在一台配备 Intel Xeon Silver 4210 CPU (2.20GHz, 40 核心) 的服务器上进行，操作系统为 Ubuntu 22.04。该设备搭载两块 NVIDIA GeForce RTX 3090 显卡，每块显卡均配备 24GB 显存，用于加速实验中的计算任务。

本文使用了四个数据集进行实验评估。两个合成数据集（偏斜分布和高斯分布）以及两个真实世界数据集（美国地质勘查局（United States Geological Survey，USGS）全球地震数据集[23]和纽约市出租车 NYCT 数据集[24]）。合成数据集用于模拟不同的统计分布，从而测试 DSPI 处理多种数据场景的能力。真实世界数据集用于评估系统在涉及复杂时空场景中的有效性。表 1 总结了这些数据集的详细统计信息，包括其类型、维度和数据集规模。

表 1 实验数据集

Tab.1 Experiment Datasets

数据集	类型	维度	数据集规模
偏斜分布数据集	合成数据集	2,4,8,16	100000
高斯分布数据集	合成数据集	2,4,8,16	100000
USGS 数据集	地理时空数据集	2,4,6,8	997361
NYCT 数据集	地理时空数据集	2,3,4,6	1000000

本文采用平均查询时间来评价索引性能，并选择了几种具有代表性的索引方法来对比评估 DSPI 的改进。在索引结构方面，R*树[25]和 KD 树[26]被选为传统树结构索引的典型代表。LISA 被选为最优的基于网格的空间学习型索引的代表。LIMS 代表了最优的基于聚类的度量空间学习型索引。在范围查询方面，R*树代表包围盒过滤策略；LISA 代表网格过滤策略；LIMS 代表三角不等式过滤策略。在 k 近邻查询方面，R*树、KD 树代表递归空间遍历策略；LIMS 代表固定初始半径的迭代范围查询策略；LISA 则代表线性插值预测初始半径的迭代范围查询策略。

4.2 参数设置

本文研究了顶层架构、枢轴数、范围查询框比例、最近邻 k 值这些参数对 DSPI 性能的影响。每次实验中仅有一个参数变化，其他参数均保持最佳默认值。

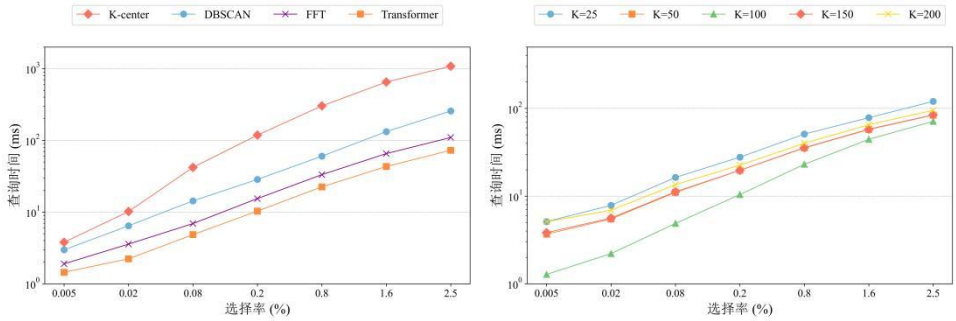
表 2 实验参数设置

Tab.2 Parameter Settings

实验参数	参数设置
顶层架构	传统聚类：K-center, DBSCAN；枢轴选择：FFT,**Transformer**
枢轴数量	25, 50, **100**, 150, 200
范围查询框比例	0.005%, 0.02%, 0.08%, 0.2%, 0.8%, 1.6%, 3.2%, 6.4%
近邻数 k 值	1, 10, 100, 1000, 10000, 50000, 100000

实验比较了不同顶层架构对范围查询性能的影响，一种是传统聚类方法，一种是直接进行枢轴选择。图 5a 实验结果表明，随着选择率的增大，传统聚类方法的查询时间迅速上升。相比之下，直接进行枢轴选择的 Transformer 和 FFT 方法在整个选择率范围内表现较为稳定，并且随着选择率的增加，查询时间的增长较为平缓。基于 Transformer 的枢轴选择方法在所有顶层架构中表现最佳，因此选择其为默认的顶层架构。

图 5b 为索引在不同选择率上范围查询性能对比实验。实验结果表明，随着枢轴数的增加，查询时间在多数情况下呈现降低趋势，增加枢轴数有助于减少每个枢轴中的数据量，从而降低查询的复杂度，但当枢轴数增加到 200 时，查询性能未能继续显著提升，这可能是由于分区管理的额外开销抵消了性能的提升。并且在查询选择率较大的情况下，不同枢轴数的查询时间趋于一致。这表明，在大范围查询中，枢轴数对查询性能的影响变得不再明显。选择枢轴数 100 是性能与管理开销之间的折中，能够在多数场景下提供良好的查询效率。因此枢轴数默认设为 100。



(a) 顶层架构　　(b) 枢轴数

图 5 不同参数范围查询性能对比

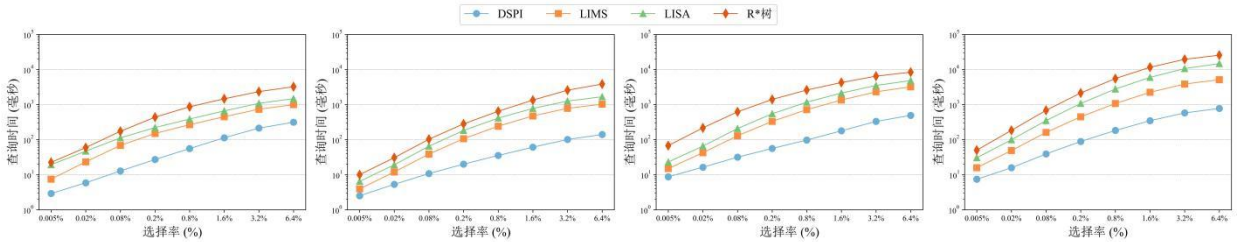
Fig. 4 Range query performance with parameter

4.3 范围查询性能

4.3.1 不同选择率下的性能

选择率作为衡量查询范围大小的关键指标，直接影响索引结构的剪枝能力与查询效率。图 6 为索引在不同选择率上范围查询性能对比实验。R*树作为传统树结构索引的代表，在较低选择率的情况下，其查询时间与其他索引接近。然而，随着选择率和数据维度的增加，R*树的剪枝效率逐渐降低，导致其查询性能迅速下降。这主要是由于 R*树在高维场景下需要遍历大量树节点，无法有效缩小搜索空间，因而在较高选择率下表现不佳。LISA 采用基于网格的空间划分策略，能够快速定位与查询区域重叠的

网格单元，这使得其在较低选择率下也能与 DSPI 相当。然而，随着选择率的增加，LISA 的过滤成本迅速增加，使其在高选择率场景中性能显著下降。LIMS 在低选择率场景下，其性能略优于 LISA、R*树。然而，随着选择率的增加，LIMS 无法有效拟合数据分布，在度量空间中逐渐退化为线性遍历，导致搜索空间无法得到有效缩减。相比之下，在所有实验设置中，DSPI 的查询时间增明显低于其他方法，特别是在较高选择率下，DSPI 始终保持较低的查询时间，得益于 DSPI 在枢轴选择和剪枝策略上的改进，使其能够更好适应复杂数据分布，尤其在高选择率场景中表现优异。根据实验数据，DSPI 的查询时间平均为 R*树的约 10.8%，LISA 的约 20.2%，LIMS 的约 39.7%。



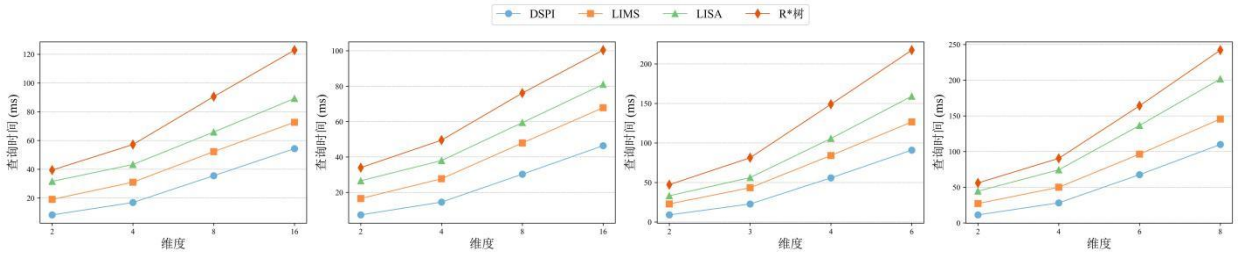
(a) 偏斜分布数据集查询时间　(b) 高斯分布数据集查询时间　(c) NYCT数据集查询时间　(d) USGS数据集查询时间

图 6 不同选择率范围查询性能对比

Fig. 6 Range query performance with selectivity

4.3.2 不同维度下的性能

数据维度是影响范围查询性能的关键因素之一，尤其在高维空间中，许多传统索引结构难以维持有效的剪枝能力。图 7 为索引在不同维度上范围查询性能对比实验。R*树在低维场景中表现稳定，但随着维度增加，其查询时间迅速上升，特别是在高维数据（如 8 维及以上）中，剪枝效率显著下降，导致性能退化。LISA 在低维时表现接近 R*树，在高维空间中无法维持有效的网格划分，导致查询复杂度增加，性能逐渐恶化。LIMS 在低维时表现略优于 LISA，但在高维场景中，受限于剪枝能力，其查询时间快速增长。相比之下，DSPI 在不同维度下始终表现稳定，其查询时间增长幅度远低于其他方法。根据实验数据，DSPI 的查询时间平均为 R*树的约 31.6%、LISA 的约 45.8%和 LIMS 的约 52.2%。



(a) 偏斜分布数据集查询时间　(b) 高斯分布数据集查询时间　(c) NYCT数据集查询时间　(d) USGS数据集查询时间

图 7 不同维度范围查询性能对比

Fig. 7 Range query performance with dimensionality

4.4 k 近邻查询性能

4.4.1 不同近邻数下的性能

图 8 为索引在不同近邻数上 k 近邻查询性能对比实验。从实验结果可以看出，DSPI 在 k 近邻查询中表现出较为突出的平均性能优势，但在 k=1 时的查询性能并不占优。这是因为 DSPI、LIMS 等度量空间索引依赖于枢轴的索引方式，在小规模最近邻查询时需要进行额外的搜索和验证操作，从而带来

额外的开销。然而，与 LIMS 相比，DSPI 的查询效率更高的原因在于其仅需一次精确的范围查询即可完成最近邻的定位，而无需迭代扩展搜索范围。DSPI 通过三角不等式和邻域学习模型优先筛选最有可能包含目标数据的枢轴，并在枢轴内高效完成查询，这种单步定位机制显著减少了不必要的计算和枢轴访问次数。而 LIMS 需要通过多次迭代扩展搜索半径逐步收敛到目标数据，导致查询效率下降，尤其在大规模 k 近邻查询和高维场景下尤为明显。

相比之下，R*树和 KD 树在近邻数较少时表现与 DSPI 相近，KD 树。但在处理大规模最近邻查询时，由于其基于逐层遍历的索引方式，需要访问大量节点和进行多次计算，从而导致查询时间迅速增加。LISA 的查询性能整体较差，其主要劣势在于利用数据分布模型预测的初始半径估算误差较大，导致重复或无关页面访问。此外，其迭代式的查询方式在 k 值增长时效率显著下降。根据实验结果，DSPI 的查询时间平均为 R*树的 12.4%，KD 树的 52.3%，LISA 的 30.6%，LIMS 的 45.5%。

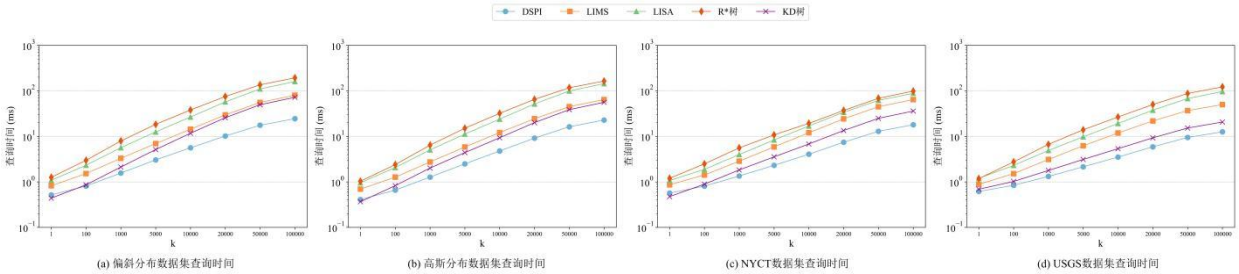


图 8 不同近邻数 k 近邻查询性能对比

Fig. 8 kNN query performance with k

4.4.2 不同维度下的性能

图 9 为索引在不同维度上 k 近邻查询性能对比实验。实验表明，R 树和 KD 树在低维场景中表现稳定，但随着维度升高，其性能急剧下降，主要因高维数据稀疏性导致节点重叠率和搜索路径复杂度激增。LIMS 和 LISA 在不同维度 kNN 查询上的相对性能与它们在不同维度范围查询上的性能相似，在高维场景下无法维持有效的剪枝能力。相比之下，DSPI 在不同维度下始终表现稳定。根据实验数据，DSPI 的查询时间平均为 R*树的约 11.6%、KD 树的约 51.6%、LISA 的约 33.8%和 LIMS 的约 42.2%。

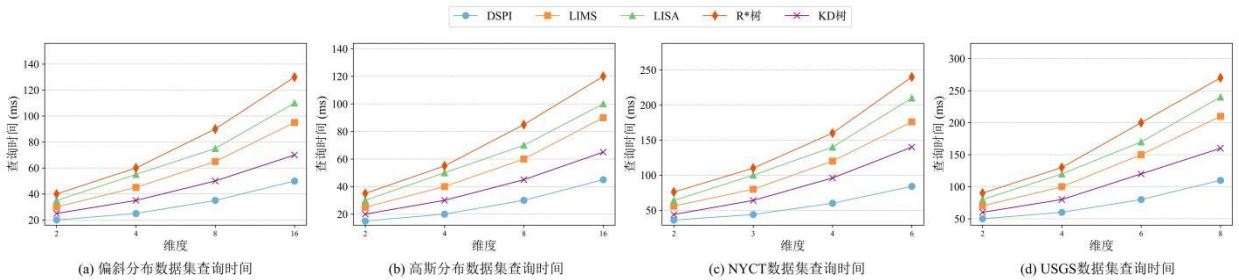


图 9 不同维度 k 近邻查询性能对比

Fig. 9 kNN query performance with dimensionality

4.5 索引构建性能

图 10 为索引在 NYCT 数据集上的构建性能对比实验。实验表明，R*树和 KD 树具有较长的构建时间以及较大的索引大小，这主要源于其传统的空间划分方法，在高维数据中往往需要频繁的重建和调整树结构，从而导致较高的计算和存储开销。相比之下，LISA 基于网格的分区方法也带来了不小的开销，在构建时间上劣势较大。LIMS 采用了先进行聚类的策略，通过枢轴上的线性模型管理数据，从

而在存储和计算开销上得到了一定的优化。在此基础上，DSPI 直接对枢轴进行管理，减少了对聚类结果的依赖。此外，DSPI 使用了早停训练机制，进一步提升了训练效率并减少了构建过程中的计算量。根据实验数据，DSPI 的索引大小相比于 LISA 减少了约 20.3 倍，相比于 LIMS 减少了约 1.5 倍，并且其构建时间比 LISA 缩短了约 5.6 倍，比 LIMS 缩短了约 1.4 倍。

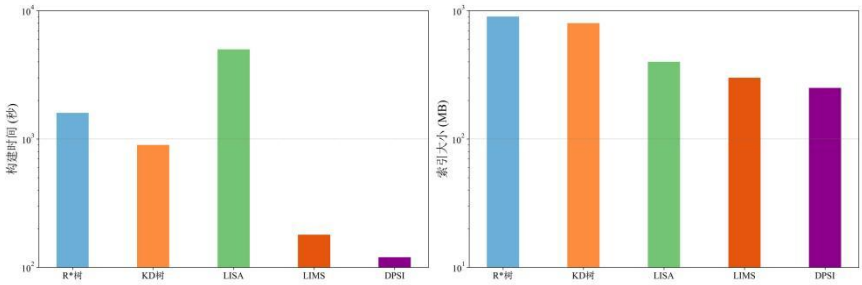


图 10 构建性能对比

Fig. 10 Performance of Building

4.6 更新性能

图 11 为 DSPI 更新性能实验，通过比较在 NYCT 数据集上插入和删除后的范围查询性能来进行评估。假设底层数据的分布在一段时间内不会发生显著变化。在插入实验中，根据实验数据，随着插入数据量的增加，查询时间也有所增加。这是由于插入导致了数据量的增加，使得查询时需要处理的点数变多，索引的结构也逐渐变得不再最优。然而，实验结果显示性能退化的速度是缓慢且稳定的，这证明了 DSPI 插入策略的鲁棒性。对于删除操作，随着删除数据点的增加，查询性能有所下降。虽然逻辑删除的数据点仍然保留在索引中，对查询性能造成了一定影响，但这种影响依然在可接受的范围内。

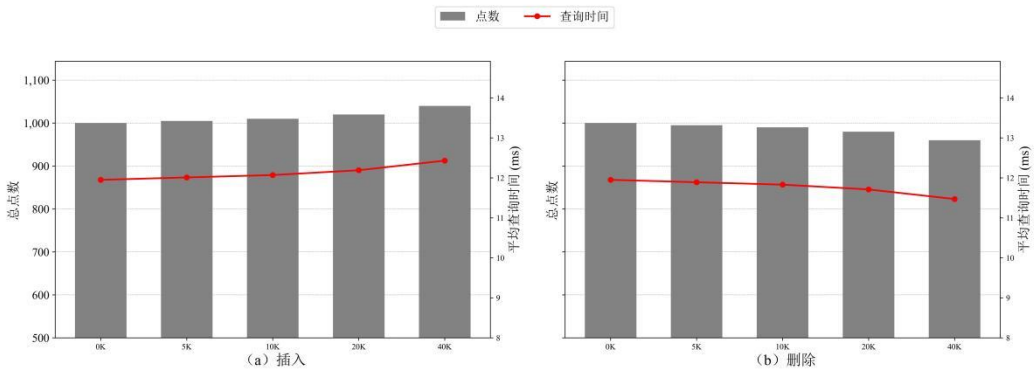


图 11 插入和删除后的范围查询性能

Fig. 9 Range query performance after insertions and deletions

5 结 语

本文提出了基于分布感知空间枢轴的学习型索引 DSPI，以解决高维和大规模空间数据查询带来的挑战。通过基于自注意力机制的 Transformer 神经网络的空间枢轴选择方法和优化的剪枝和查询策略，DSPI 在存储和查询性能上相比现有方法表现出了显著提升，尤其在高维和大规模查询场景中具有较好的适应性。实验结果表明，与现有空间学习索引算法相比，DSPI 在存储和查询性能上分别提升了 1.4

至 20.3 倍和 1.1 至 10.2 倍。在高维和大规模查询场景中，DSPI 的查询性能进一步提升了 1.5 至 15.2 倍。

在未来的研究中，DSPI 将从当前的点数据索引扩展到其他数据格式，包括线数据、面数据、体数据和其他复杂的地理数据类型。对于线数据，枢轴选择不仅需要考虑线段之间的距离，还需结合其相对位置关系，未来可探索引入基于拓扑结构的策略，以更有效地捕捉空间依赖性。对于面数据，需综合考虑其几何形状与相对分布关系，可能引入凸包构建、空间分割等技术手段，提升枢轴选择的代表性与准确性。对于体数据，三维空间结构的复杂性要求枢轴选择过程融入体边界、空间覆盖范围等关键要素。随着上述算法与索引策略的逐步完善，DSPI 有望在更广泛的复杂时空场景下展现更强的适应性与查询效率

参考文献

[1] Kraska T, Beutel A, Chi E H, et al. The case for learned index structures[C]//Proceedings of the 2018 International Conference on Management of Data. 2018: 489-504.

[2] Tian Y, Yan T, Zhao X, et al. A learned index for exact similarity search in metric spaces[J]. IEEE Transactions on Knowledge and Data Engineering, 2022, 35(8): 7624-7638.

[3] Chen L, Gao Y, Song X, et al. Indexing Metric Spaces for Exact Similarity Search[J]. ACM Computing Surveys, 2022, 55(6): 128.

[4] 胡林舒.地理流学习型时空索引高效检索与计算[D].浙江大学,2021.

[5] Ferragina P, Vinciguerra G. The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds[J]. Proceedings of the VLDB Endowment, 2020, 13(8): 1162-1175.

[6] Kraska T, Alizadeh M, Beutel A, et al. Sagedb: A learned database system[J]. 2021.

[7] Nathan V, Ding J, Alizadeh M, et al. Learning multi-dimensional indexes[C]//Proceedings of the 2020 ACM SIGMOD international conference on management of data. 2020: 985-1000.

[8] Li P, Lu H, Zheng Q, et al. LISA: A learned index structure for spatial data[C]//Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020: 2119-2133.

[9] Wang H, Fu X, Xu J, et al. Learned index for spatial queries[C]//2019 20th IEEE International Conference on Mobile Data Management (MDM). IEEE, 2019: 569-574.

[10] Qi J, Liu G, Jensen C S, et al. Effectively learning spatial indices[J]. Proceedings of the VLDB Endowment, 2020, 13(12): 2341-2354.

[11] Davitkova A, Milchevski E, Michel S. The ML-Index: A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries[C]//EDBT. 2020: 407-410.

[12] Jagadish H V, Ooi B C, Tan K L, et al. iDistance: An adaptive B+-tree based indexing method for nearest neighbor search[J/OL]. ACM Transactions on Database Systems, 2005, 30(2): 364-397

[13] Gu T, Feng K, Cong G, et al. The RLR-Tree: A Reinforcement Learning Based R-Tree for Spatial Data[J]. Proc. ACM Manag. Data, 2023, 1(1): 63.

[14] Abdullah-Al-Mamun A, Haider C Md R, Wang J, et al. The “AI + R”-tree: An Instance-optimized R-tree[C]//2022 23rd IEEE International Conference on Mobile Data Management (MDM). 2022: 9-18.

[15] Galakatos A, Markovitch M, Binnig C, et al. Fiting-tree: A data-aware index structure[C]//Proceedings of the 2019 international conference on management of data. 2019: 1189-1206.

[16] Zhu Y, Chen L, Gao Y, et al. Pivot selection algorithms in metric spaces: A survey and experimental study[J]. VLDB Journal, 2022, 31(1): 23-47.

[17] Hochbaum D S, Shmoys D B. A best possible heuristic for the k-center problem[J]. Mathematical Programming, 1985, 10(2): 180-184.

[18] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[J]. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996: 226-231.

[19] Chavez E, Navarro G, Baeza-Yates R A, et al. Searching in metric spaces[J]. ACM Computing Surveys (ACM Comput. Surv.), 2001, 33(3): 273-321.

[20] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems (NeurIPS), 2017, 30: 5998-6008.

[21] Chiu C Y, Prayoonwong A, Liao Y C. Learning to Index for Nearest Neighbor Search[J/OL]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42(8): 1942-1956 (2020-08-01).

[22] Akaike H. Information theory and an extension of the maximum likelihood principle[J]. Second International Symposium on Information Theory, 1973: 267-281.

[23]U.S. Geological Survey. Earthquakes [DB/OL]. [2025-01-16]. https://earthquake.usgs.gov/earthquakes/

[24] Nyc taxi trips [DB/OL]. [2025-01-16]. https://nycopendata.socrata.com

[25] Beckmann N, Kriegel H P, Schneider R, et al. R* tree: An efficient and robust indexing structure for points and rectangles[J]. ACM SIGMOD Record, 1990, 19(2): 322-331.

[26] Bentley J L. Multidimensional binary search trees used for associative searching[J]. ACM Communications, 1975, 18(9): 509-517.