CRISP：通过子空间划分实现抗相关索引

Dimitris Dimitropoulos 约阿尼纳大学与 Archimedes、Athena RC 希腊约阿尼纳 ddimitropoulos@cs.uoi.gr

Achilleas Michalopoulos 约阿尼纳大学 希腊约阿尼纳 a michalopoulos@cs.uoi.gr

Dimitrios Tsitsigkos Archimedes、Athena RC 希腊雅典 dtsitsigkos@athenarc.gr

Nikos Mamoulis 约阿尼纳大学与 Archimedes、Athena RC 希腊约阿尼纳 nikos@cs.uoi.gr

arXiv:2603.05180v1 [cs.DB] 5 Mar 2026

摘要

随着现代学习型表征的维度增至数千，最先进的近似最近邻（ANN）索引暴露出严重局限。基于图的方法（如 HNSW）面临高得令人难以承受的内存消耗和路由性能下降，而近期的随机量化和学习型旋转方法（如 RaBitQ、OPQ）则带来了显著的 $O(ND^2)$ 预处理开销。我们提出 CRISP，这是一种专为超高维空间中的近似最近邻搜索设计的新框架。与不加区别地应用高成本正交旋转的僵化流程不同，CRISP 采用轻量级、具备相关性感知能力的自适应策略，仅在必要时重新分配方差，从而有效降低预处理复杂度。我们将这一自适应机制与缓存友好的压缩稀疏行（CSR）索引结构相结合。此外，CRISP 还集成了多阶段双模式查询引擎：保证模式可维持严格的召回率理论下界；而优化模式 则利用基于排名的加权评分和提前终止来降低查询延迟。针对超高维数据集（最高达 $D = 4096$）的大量评估表明，CRISP 实现了最先进的查询吞吐量、较低的构建成本和卓越的内存效率。

PVLDB 引用格式：
Dimitris Dimitropoulos、Achilleas Michalopoulos、Dimitrios Tsitsigkos 和 Nikos Mamoulis。CRISP：通过子空间划分实现抗相关索引。PVLDB，14(1)：XXX-XXX，2020。doi:XX.XX/XXX.XX

PVLDB 工件可用性：
源代码、数据和/或其他工件已在 https://github.com/dabouledidia/CRISP 上提供。

1 引言

高维空间中的近似最近邻（ANN）搜索是现代数据系统的核心组件，支持从基于内容的图像检索到面向大语言模型（LLM）的检索增强生成（RAG）等各种应用。

本作品采用知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议授权。访问 https://creativecommons.org/licenses/by-nc-nd/4.0/ 查看该许可协议的副本。对于超出本许可协议范围的任何使用，请发送电子邮件至 info@vldb.org 获取许可。版权所有者为作者。出版权授权给 VLDB基金会。《VLDB基金会会刊》，第 14 卷，第 1 期，ISSN 2150-8097。doi:XX.XX/XXX.XX

尽管近期的近似最近邻搜索索引方法在中高维数据（例如，$D \leq 256$）上非常有效，但将这些索引扩展到更高维度仍是一项重大挑战。现代基础模型通常会输出维度极高的向量，例如 OpenAI 的文本嵌入（$D = 3072$）或 Trevi 等特定领域的描述符（$D = 4096$）。扩展向量数据库以管理此类向量，会打破现有近似最近邻算法的性能权衡 [25, 26]。

具体而言，基于图的方法（如 HNSW） [22],目前因其较高的查询吞吐量而主导着业界部署，但在超高维场景（$D \geq 600$）中，需要大量内存来存储邻接表和原始向量，索引构建的顺序处理速度较慢，并且在复杂的数据分布下，路由效率通常会降低 [31, 36]。

为了解决内存限制，近期研究转向了基于子空间划分 [4, 5, 34]和量化 [10, 11, 16, 23]的方法。与图方法不同，这些方法通过倒排索引结构提供紧凑的内存占用，该结构可随数据集规模线性扩展，并支持缓存高效的线性访问模式。两个典型示例是 SuCo（子空间碰撞）[34] 和 RaBitQ（随机比特量化） [10]。然而，将这些方法应用于数千维数据时，会暴露出两个实际局限：它们要么难以有效处理高度相关的特征，要么预处理成本过高。

SuCo 的框架 [34] 假设各维度承载相互独立的信息，但现实世界中的高维嵌入往往不满足这一假设。在相关特征空间（例如 Gist 和 Fashion-MNIST）中，方差高度集中于少数维度，导致子空间捕获的信息大多冗余，使碰撞代理指标无法区分邻居。我们的评估结果（图5）展示了这一现象：在相关数据集上，SuCo 的检索质量存在明显的召回率上限。

为处理高度相关的特征空间，优化乘积量化（OPQ） [11] 以及近期的随机量化框架（如 RaBitQ） [10] 会应用全局正交旋转，在所有维度间重新分配方差。尽管这种变换效果显著，但对于维度为 D 的 N 个向量，该变换需要耗费 $O(ND^2)$ 时间。这种二次复杂度给现代表征学习模型带来了显著的预处理开销，因为这类模型会生成维度达数千的嵌入。由于 RaBitQ 会不加区分地对所有数据集应用这种变换，

即使对于天然不相关、原始分布本就适合建立索引的数据，也要支付 $O(ND^2)$ 开销。

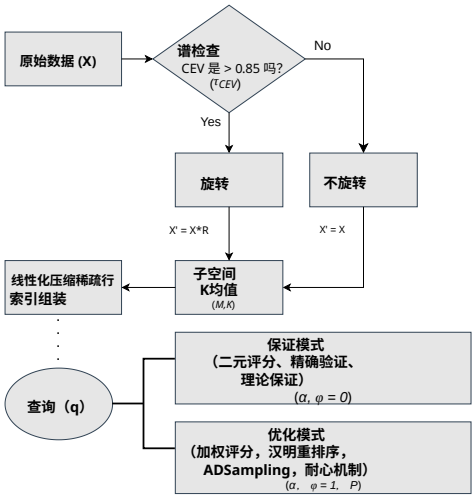


图 1：CRISP 架构概览。

本文指出，一刀切的预处理流程从根本上不适用于超高维近似最近邻搜索的需求。我们提出 CRISP，这是一种专为维度极高的特征空间设计的自适应框架，可根据数据的相关结构动态调整预处理策略，从而实现稳健检索，并在无需预处理时绕过其 $O(ND^2)$ 代价。

CRISP 的架构如图 1 所示。在预处理阶段，CRISP 使用一种轻量级启发式方法分析数据相关性，以决定是否需要进行方差重分布。CRISP 从根本上融合了两种互补范式：它兼具基于碰撞的方法的低索引构建成本 [34] 以及量化方法的分布鲁棒性 [10]。

应用旋转决策机制并视情况进行旋转后，CRISP 将特征空间划分为 M 个互不相交的子空间。在每个子空间内，我们采用倒排多重索引（IMI）分区方案，将子空间分解为更低维的片段，并分别对每个片段执行独立的 k 均值聚类，从而构建细粒度的单元格网格。这种方法以 SuCo 提出的子空间碰撞范式为基础，将其作为欧几里得距离的近似度量，同时针对现有索引无法扩展到的高维相关分布进行了专门优化。

我们将子空间划分结构与高度优化的系统架构相结合，以最大限度地提高缓存效率和指令级并行度。我们为每个子空间引入一种缓存一致的压缩稀疏行（CSR）索引结构，以顺序内存访问模式取代传统基于哈希的倒排列表所带来的指针追逐开销。在我们的实现中，给定子空间的所有点 ID 都存储在一个连续数组中，而另一个偏移量数组则明确标记每个分区桶的起始和结束位置。这种设计显著减少了候选投票和碰撞计数阶段的地址转换后备缓冲器（TLB）未命中；在这一阶段，系统必须快速检索并汇总来自已激活分区桶的点 ID。通过将索引线性化后，硬件预取器可以高效地将分区桶内容流式加载到缓存中，从而在超高维场景下显著加快查询速度。

为最大限度地提高查询效率，我们设计了一个多阶段查询流水线，逐步将数据集筛选为候选集。首先，通过子空间碰撞计数检索最有可能包含真实最近邻的点集。随后，我们对筛选出的候选项应用双模式验证。在保证模式下，系统对候选集执行穷举式精确欧几里得验证，同时保留根据子空间碰撞统计推导出的严格召回率理论下界。在优化模式下，我们通过引入基于排名的加权评分机制来优先提升吞吐量。该机制细化了 SuCo 中使用的粗略碰撞计数，通过提高邻近子空间中发生的碰撞的重要性，实现距离感知排名。此模式还通过汉明距离重新排序、ADSampling 和动态耐心终止机制进一步加速查询。

我们的贡献可概括如下：

- **自适应预处理策略：** 我们提出了一种具有相关性感知能力的预处理机制，仅在检测到较强的维间相关性时才选择性地应用随机正交旋转。对于不相关数据，则完全跳过预处理，从而避免 $O(ND^2)$ 预处理开销。这种自适应策略与 RaBitQ 和 OPQ 形成对比，后两者不加选择地应用代价高昂的全局变换。

- **严谨的理论保证：** 我们利用 Hoeffding 的 [14] 不等式推导出条件召回率下界，证明检索失败概率会随着子空间数量的增加而呈指数级下降。这一保证严格优于此前子空间碰撞框架提供的多项式界（切比雪夫） [34], ，并能在满足我们的自适应独立性条件时确保稳健的检索质量。

- **多阶段双模式查询引擎：** 我们设计了一条用于高效候选项排序的查询流水线，并配以针对不同模式的验证策略。保证模式执行穷举验证，以维持严格的召回率下界。优化模式则采用三种互补的加速技术：（i）基于排名的加权评分，（ii）AD 采样 [9], 以及（iii）动态耐心终止，以最大限度提高吞吐量。

- **超高维向量上的评估：** 我们在维度高达 $D = 4096$ 的数据集上，严格评估 CRISP，并将其与业界标准的图基线方法 HNSW、学习式旋转 OPQ、随机量化 RaBitQ 以及原始子空间碰撞框架 SuCo 进行比较。结果表明，在吞吐量、召回率和构建时间方面，CRISP 实现了更优的帕累托最优权衡，同时保持线性内存占用，而图方法则难以扩展。

2 相关工作

高维空间中的近似最近邻（ANN）搜索问题已得到广泛研究，相关方法通常分为三大类：基于划分的方法、基于量化的方法和基于图的方法。

2.1 划分与子空间方法

早期的近似最近邻搜索方法侧重于通过空间划分来高效剪枝候选项。向量近似文件（VA-File） [33] 率先采用标量量化压缩向量，并依据下界筛选候选项。VA-File 后来发展为更复杂的基于划分的索引，例如倒排多索引（IMI） [5], 。该索引将向量空间分解为正交子空间，并在每个子空间中独立应用 K 均值聚类，从而生成细粒度的单元格笛卡尔积。每个数据点都会被分配到各个子空间中离它最近的质心；在查询时，系统会找出各个子空间中距离查询最近的质心，并取其倒排列表的并集，作为候选近邻。

在这一划分范式的基础上，子空间结构主要用于候选剪枝，而非紧凑向量编码。一系列研究致力于改进子空间邻近性的度量方式及其利用方法。查询感知局部敏感哈希（QALSH）[15] 和 PM-LSH [37] 通过子空间投影改进了碰撞界，但在分布高度偏斜时，两者都面临负载均衡方面的挑战。SuCo [34] 则进一步推进了这一方向，引入碰撞计数度量作为欧几里得距离的代理指标。然而，所有这些方法都存在一个共同局限：它们假设局部子空间邻近性意味着全局邻近性。正如我们在第 6 节所展示的，这一假设会被现实世界极高维数据中固有的强烈维间相关性所打破。在这种情况下，子空间会捕获冗余信息，导致碰撞代理指标失效。

2.2 基于量化的方法

与主要利用子空间分解进行候选剪枝的划分方法不同，基于量化的方法利用同一分解原理将向量压缩为紧凑码，从而大规模降低内存占用。乘积量化（PQ） [16] 率先将向量分解为子空间，以便进行局部码本学习；这一技术也构成了 FAISS [17]等成熟高性能库的基础。为降低相关数据上的量化误差，优化乘积量化（OPQ） [11] 学习一个参数化旋转矩阵，以最小化失真。然而，OPQ 依赖迭代优化，随着维度增加，计算成本会变得过高。对于超高维向量（$D \geq 1000$），优化稠密的 $D \times D$ 旋转矩阵会造成显著的训练瓶颈。

最近，RaBitQ [10] 提出了一种随机量化框架，利用全局正交旋转来平坦化向量能量谱，从而为按位距离估计提供理论误差界。然而，对于较大的 D值，旋转的 $O(N \cdot D^2)$ 复杂度会带来巨大的预处理成本。ScaNN [13] 和 RoarGraph [7], 等混合方法将量化与各向异性评分或图结构相结合。ScaNN 通过专门的损失函数和复杂的查询端评分来处理量化误差，而 RoarGraph 则继承了图索引较高的构建成本。

2.3 基于图的方法

基于图的索引，尤其是分层可导航小世界（HNSW） [22], ，是内存中近似最近邻搜索的当前行业标准

表 1：关键符号说明汇总

符号	描述
N	数据集基数
D	数据维度
M	子空间数量
K	每个子空间的质心数量
k_{size}	真实最近邻数量
X	原始数据集矩阵（$N \times D$）
C	由碰撞阈值 τ筛选的候选集
R	随机正交旋转矩阵 ($D \times D$)
τ_{CEV}	累计解释方差阈值
ϕ	双模式执行标志
α	最低所需子空间碰撞百分比

[4]。这些方法依赖邻近图，搜索算法通过贪心方式在图中向目标移动。Vamana 和 DiskANN 等扩展 [24] 将这种结构调整为适用于 SSD 存储，以处理更大规模的数据。然而，基准测试表明，随着维度增加，图的性能会显著下降 [31], ，因为邻居之间的距离差异逐渐缩小，导致贪心路由的效果变差 [6, 32]。这一根本性限制因传统距离比较在大规模场景下统计可靠性不足且计算成本高昂而进一步加剧，因此需要更稳健的距离比较操作 [9]。此外，将图结构与高维向量一同存储会导致内存占用难以管理 [1]。SPANN 等系统[8]通过将数据转存到磁盘来减少内存使用量，但在最后的搜索阶段会因磁盘读取而产生较高延迟 [8, 30]。

3 预备知识

本节将形式化定义近似最近邻搜索问题，并介绍构成自适应框架基础的核心原语。表 1 总结了本文中使用的关键符号。

精确与近似相似性搜索。 相似性搜索通常称为最近邻（NN）搜索，它根据距离函数 dist(·)，在数据库中查找与指定查询对象最相似的项目。形式化地，设 $X \in \mathbb{R}^{N \times D}$ 为一个数据集矩阵，其中包含 N 个实值 D 维向量，并设 $q \in \mathbb{R}^D$ 为一个查询向量。目标是确定 X 中与查询的距离最小的数据向量：

$$\text{NN}(q) = \arg\min_{x \in X} \text{dist}(q, x)$$

欧几里得距离（L2）是大多数应用中标准的距离函数选择 [21, 29]。该问题表述自然扩展为 k-最近邻（kNN）搜索，旨在检索与查询距离最近的 k 个项目集合。

相似性搜索算法通常分为两类：精确方法和近似方法 [28]。精确方法能够保证检索到真实最近邻；然而，它们往往需要很高的计算代价，因此不适用于要求高查询吞吐量的应用。相比之下，近似最近邻方法牺牲少量检索准确率，以换取查询吞吐量的大幅提升。在大多数大规模、高维

场景中，人们更倾向于采用近似方法，因为这类方法能够在保持足够准确率的同时实现低延迟处理 [21, 28]。在检索精度与效率之间取得平衡，是现代近似最近邻系统设计面临的核心挑战 [3]。

子空间碰撞框架。子空间碰撞（SuCo）框架 [34] 通过将 D 维空间划分为 M 个互不重叠的子空间，来近似两个向量之间的邻近度。在每个子空间 m中，倒排多索引方案会进一步将该子空间划分为两半，并通过 K 均值为每一半学习一个包含 K 个质心的码本。每个数据点 x 由其在每个子空间两半中最近的质心表示，并映射为一个由单元索引组成的离散元组 $u(x) = [u_1(x) \ldots, u_M(x)]$。检索期间，算法按网格单元与查询子向量之间聚合欧几里得距离的升序探索网格单元，并使用动态激活算法（多序列算法的一种变体），直到检索到足够大的候选集。查询 q 与数据点 x之间的邻近度通过碰撞计数 $S_{\text{col}}(q, x)$估算：即检索过程中 x被分配到的单元处于激活状态的子空间数量。形式化地说：

$$S_{\text{col}}(q, x) = \sum_{m=1}^{M} \mathbb{1}\left(u_m(x) \in \mathcal{U}_m(q)\right) \tag{1}$$

该整数分数作为欧氏距离的概率估计量，有效地将邻近度量化为 $M+1$ 个离散层级。现有框架通过切比雪夫不等式为检索失败提供理论保证，但这些多项式界假设跨子空间的碰撞是独立事件。当不同子空间的特征存在相关时，该假设不再成立，从而降低 SuCo 的判别能力。

候选项精炼。精炼是指对子空间碰撞过滤阶段识别出的候选集 C进行重新排序。由于初始碰撞计数是离散整数（因而只能提供低分辨率排名），精炼采用更高精度的距离估计器，例如基于二进制签名的汉明距离或基于浮点向量的 ADSampling，以便在最终的 top-k 选择之前严格排序并剪枝候选项。

随机正交旋转。为缓解相关数据中方差分布不均所导致的量化误差，RaBitQ 采用随机正交变换。设 $G \in \mathbb{R}^{D\times D}$ $G_{ij} \sim$ 0, 1 为元素服从 N ( ) 的随机高斯矩阵。旋转矩阵 R通过对 G 进行 QR 分解得到：$G = Q \cdot R'$，其中我们设定 $R \leftarrow Q$。变换后数据集 $X' = \{x \cdot R \mid x \in X\}$ 具有各维度方差近似均衡的性质。虽然这确保后续量化步骤作用于统计性质良好的数据，但应用这一全局变换需要进行复杂度为 $O(ND^2)$ 的稠密矩阵乘法。对于极高维数据集（例如$D = 4096$），该预处理步骤会带来难以承受的计算开销。在 CRISP 中，我们自适应地使用该变换，仅当维度间相关性（相对于 τ_{CEV} 度量）需要时才应用它。

二值量化。二值量化（BQ） [12] 通过符号函数将连续向量映射为紧凑的位串 $b(x) \in \{0,1\}^D$，从而生成高度压缩的向量表示。两个二进制编码之间的差异可通过以下方式高效衡量汉明距离 $\|\cdot\|_H$，该距离可使用快速的硬件级按位异或和人口计数操作计算。在 CRISP 中，二值量化作为欧几里得邻近度的轻量级替代指标，用于在细化阶段进行高成本距离计算之前，对候选集 C 排序并确定优先级。

ADSampling. ADSampling [9] 通过逐步增加所用 t维度的子集，估算查询 q与候选项 x之间的平方欧几里得距离。为避免过早剪除真实最近邻，该框架采用基于相对误差界的剪枝条件：

$$d_t^2(q, x) > r_k^2 \cdot \frac{t}{D}\left(1 + \frac{\epsilon_0}{\sqrt{t}}\right)^2 \tag{2}$$

其中，r_k^2是当前第 k近邻的平方距离，而 $d_t^2(q, x) = \sum_{i=1}^{t}($ $q_i - x_i)^2$ 是部分和。CRISP按照 [18]中的实现设置安全裕度 $\epsilon_0 = 2.1$。引擎每经过固定数量的 $t = 32$ 维度便执行这些检查。这样，细化阶段就能以较高的统计置信度逐步剔除非邻居候选项，大幅减少所需的全维度距离计算次数。

4 CRISP框架

本节介绍CRISP——我们的自适应索引框架，它兼顾了子空间划分的效率与随机量化的鲁棒性。CRISP专为超高维数据集（$D \geq 600$）设计，包含三个不同的阶段：

（1）**相关性感知预处理**，自适应地转换数据分布，以恢复子空间独立性。（2）**基于CSR的索引**，将数据映射到缓存一致的倒排结构中，以最大化内存吞吐量。（3）**双重多阶段查询流水线**，使用向量化操作逐步筛选候选项。

图 1 展示了 CRISP 架构的概览。其查询处理步骤由算法 1 总结。

4.1 相关性感知预处理

我们观察到，子空间碰撞方法主要在相关数据集上失效，因为向量的能量集中于少数几个主成分。在高维空间中，这种集中会导致子空间捕获冗余信息，从而降低索引的判别能力。为此，我们在索引之前引入轻量级的**谱相关性检查** 。为确保该步骤产生的开销可以忽略不计，我们在规模有界的随机样本 $X_{sample} \subset X$ 上计算协方差矩阵，样本大小根据经验设定为 $\min(0.1N, 10_5)$。我们执行特征值分解，以计算**累计解释方差（CEV）**，其定义为前 20% 个主成分所解释的累计方差：

$$\text{CEV} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{D} \lambda_j}, \quad \text{where } k = \lfloor 0.2 \cdot D \rfloor$$

基于这一谱分析，该框架会动态选择最优构建路径。我们采用通过实验得出的阈值 $\tau_{CEV} = 0.85$ （并在第6.4小节通过消融实验加以论证）。该值可作为一种保守的代理指标，用于

识别维度间相关性强到足以使我们的理论误差界失效的数据分布；我们在第 5 节中对其进行了正式定义和分析。当 CEV 超过该阈值时，CRISP 会检测到显著偏离各向同性的情况，并触发“方差重分布”。在这种情况下，我们应用第 3 节中定义的随机正交旋转 R，生成变换后数据集 $X' = XR$。这一变换能够有效地将方差均匀地重新分布到整个向量空间中，确保没有任何单一维度主导划分过程。

相反，对于 CEV 低于 0.85 的分布，CRISP 认为其自然分布的离散程度已足够高。在这种情况下，我们完全跳过旋转步骤，从而避免 $O(ND^2)$ 的计算开销，直接进行索引构建。与 RaBitQ 等僵化的量化框架相比，这种集成设计具有明显的架构优势。RaBitQ 将旋转视为独立的外部预处理步骤，需要生成一份变换后数据集副本；而 CRISP 则将旋转矩阵直接保存在索引元数据中。这样，查询引擎便可在原生模式和旋转模式之间切换，而无需依赖外部组件。

至关重要的是，这种集成设计确保了更优的内存可扩展性。独立的处理流水线（例如 RaBitQ）通常将旋转视为外部预处理步骤，需要生成第二份 $N \times D$ 数据集；而 CRISP 则在索引构建过程中原地完成变换。具体而言，系统逐个遍历数据集中的向量，并使用一个大小为 D 的小型线程局部缓冲区存储中间点积，随后覆盖原始内存地址。因此，CRISP 的峰值内存使用量不会超过原始数据集的大小（ND），而分离式流水线的内存使用量往往会达到其两倍（$2ND$）。

4.2 缓存一致CSR索引

高维数据的检索从根本上受内存带宽限制。随着维度增加，向量尺寸变大，导致传统的倒排文件（IVF）结构频繁引发CPU停顿和缓存未命中。以往工作中的IVF实现将倒排列表存储在碎片化的数据结构中，可能产生“指针追逐”开销和TLB未命中。

例如，SuCo [34]使用向量形式的无序映射（`vector<unordered_map<pair<int, 整数>, 向量<整数>>>`）实现其索引；检索子空间单元的ID需要先进行哈希表查找，再访问一个独立分配在堆上的向量，因此会产生分散且不连续的内存访问。RaBitQ [10] 同样存在空间局部性差的问题，因为其搜索函数必须在每次簇探测时，从多个彼此独立的内存区域汇总距离分量。为解决这些“指针追逐”开销，CRISP采用压缩稀疏行（CSR）结构，将索引线性化为单个连续的内存块，如图2所示。

CRISP 的 CSR 将倒排索引实现为稀疏矩阵，其中行对应单元格，列对应数据点。一个单元格被定义为子空间 m 内两个子质心集合（$K \times K$）的笛卡尔积。CRISP 根据 [34] 将每个子分区的质心数量设为 $K = 50$。在构建过程中，每个数据点 $x_n \in \{1, \ldots\ldots, N\}$ 都被分配到特定单元格。为确保物理局部性，我们生成 $(cell, x_n)$ 元组，并严格

算法 1CRISP 的双模式查询执行

要求： 查询 q，模式 ϕ，索引 $\mathcal{I}$
确保： 前 k 个候选项
初始化得分映射 V $\leftarrow \mathbf{0}$
对 $m \leftarrow 1$ 至 M **执行**
将 $q^{(m)}$ 分解为 q_{left}，q_{right}
  $Dist_1 \leftarrow \text{Dist}(q_{left}, C_{left})$; $Dist_2 \leftarrow \text{Dist}(q_{right}, C_{right})$
  Q $\leftarrow$ 优先队列({ $(Dist_1[0] + Dist_2[0],\ 0,\ 0)$ })
  排名 $\leftarrow 0$; 已检索项 $\leftarrow 0$
  当已检索项 $<$ 预算**时**
    $(cost,\ i,\ j) \leftarrow$ Q。弹出()
    $cell \leftarrow$ 合并$(i,\ j)$; 排名 $\leftarrow$ 排名 $+ 1$
    $w \leftarrow 1$
    如果 $\phi = 1$ **且** 排名 $\leq k_{size}$ **则**
      $w \leftarrow 2$
    结束如果
    对于 $id \in \mathcal{I}.\text{ids}_m[cell]$ **执行**
      $\text{V}[id] \leftarrow \text{V}[id] + w$
      已检索项 $\leftarrow$ 已检索项 $+ 1$
    结束循环
    Q。推入(来自 $Dist_1$，$Dist_2$ 的新候选项)
  结束 while
结束 for
候选项 C $\leftarrow \{x \mid \text{V}[x] \geq \tau\}$
如果 $\phi = 1$ **则**
  C $\leftarrow$按汉明距离排序(C)
结束条件
对 $x \in C$ **执行循环**
  使用 ADSampling 进行验证($\phi = 1$) 或使用精确 L2($\phi = 0$)
  如果 $\phi = 1$ **且** 达到耐心阈值 **则**
    中断
  结束条件
结束 for
返回 Top-k(C)

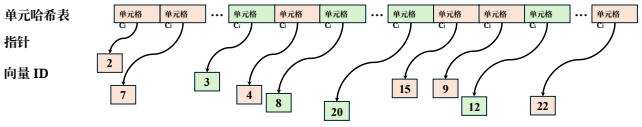
根据单元标识符对数据集进行排序。例如，若点 {2，4，7，9，15，22} 被分配到单元格 C_i，而点 {3，8，12，20} 被分配到单元格 C_j，则它们在内存中被重排为 {3，8，12，20，2，4，7，9，15，22}。这确保属于同一单元格的所有标识符连续存储，从而消除碎片化实现中的指针追逐开销。所得索引被整合为每个子空间的两个数组：

- 偏移量：一个大小为 $K^2 + 1$ 的数组，其中 K 表示每个子分区的质心数量。偏移量[i] 与偏移量[i+1] 的值作为整数偏移量，用于界定数据数组中第 i 个单元格的倒排列表的起止位置。
- 向量 ID：每个子空间一个大小为 N 的单一连续数组，其中已排序的点标识符存放在缓存对齐块中。

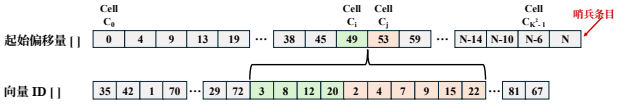
如算法 1 所示，该结构使查询执行能够通过流式读取单个连续内存段，遍历特定单元格的候选项。借助偏移量数组以常数时间确定段边界，CRISP 最大限度地提高硬件预取效率，并将瓶颈从内存延迟转移到内存带宽。CRISP 在 ID 数组中对候选项标识符采用连续布局，使累加阶段能够接近理论内存吞吐量。

4.3 多阶段双模式查询引擎

4.3.1 自适应碰撞评分。为弥合理论严谨性与实践性能之间的差距，CRISP 根据执行模式 ϕ采用双策略评分机制，并将其应用于算法 1 的查询执行流程中。



(a) 传统碎片化布局。



(b) CRISP CSR 布局。

图 2：CRISP 以压缩稀疏行（CSR）布局（b）取代了零散的基于哈希的遍历（a），瓶颈也由内存延迟转移至峰值内存带宽。

二元评分（保证模式，ϕ=0）。当需要严格的理论保证时，我们采用二元碰撞计数。只有当数据点 x 落入查询 q 在子空间 S_i 中指定的质心单元时，才会获得 +1 的评分增量。这确保总碰撞评分服从二项分布，满足第 5 节定理 5.1 的下界证明所需的独立性假设。

加权评分（优化模式，ϕ=1）。在延迟关键型优化路径中，我们放宽二元约束，采用基于排名的加权方案来捕捉更细粒度的邻近度信息。与以往的子空间碰撞和碰撞计数框架相比 [27, 34, 35, 37], , 后者通常采用统一二元评分系统（如 CRISP 的保证模式），CRISP 的优化模式引入了邻近性感知加权方案。通过优先处理在排名最高的区域中找到的候选项，CRISP 能让高质量邻居比其他方法更快地达到碰撞阈值。

具体而言，在查询阶段，对于每个子空间 m，我们按照特定顺序探索距离查询子向量 $q(m)$最近的单元格。我们根据正在访问的单元格的排名分配碰撞权重W。形式上，设 $rank(u_m(x))$表示分配给点 x的单元格的访问顺序。加权函数定义如下：

$$\mathcal{W}(q^{(m)}, x^{(m)}) = \begin{cases} 2 & \text{if } rank(u_m(x)) \le k_{size} \\ 1 & \text{otherwise} \end{cases}$$

为高效执行这种遍历，我们采用了针对第 4.2 节中所述倒排多索引结构改编的 多序列算法 [5]。在每个子空间中，向量空间被分解为两个正交半部，每个半部由独立的码本进行量化。我们没有穷举所有质心组合的笛卡尔积（其规模会随码本大小呈平方级增长），而是采用两阶段策略。首先，分别对查询与每个半部质心之间的平方欧几里得部分距离进行排序。其次，我们使用优先队列，通过累加这些预先计算的部分距离，从距离最近的一对开始，逐步扩展候选单元（i，j）。图3展示了单元遍历顺序及加权方案，适用于

简化的 5×5 质心网格。该机制确保按照单元格的聚合邻近度严格升序访问单元格。

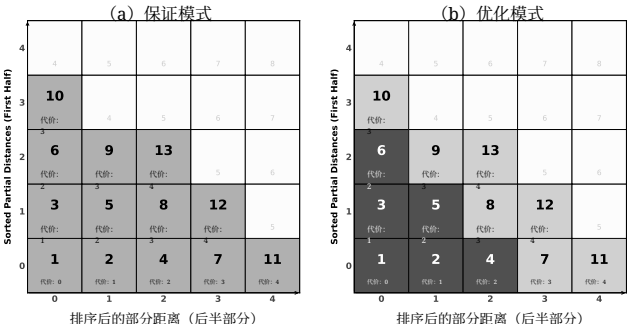


图3：在 5×5 质心网格上进行候选项评分。坐标轴显示每个子空间半区内排序后的部分距离 $(0, 1, 2, \ldots)$。单元格按代价升序访问（排名 1–13），直到检索到足够的候选项。（a）保证模式：所有候选项均采用均匀加权（$w = 1$）（灰色）。（b）优化模式：代价最低的前 k_{size} 个单元格（深灰色，排名 1–6）采用双倍权重（$w = 2$），以优先考虑可能的最近邻。其余单元格（浅灰色）采用标准权重（$w = 1$）。

对于查询，一旦检索到所需的候选项总量，算法即可立即终止搜索，从而避免不必要的距离计算。

4.3.2 执行流水线。CRISP 的双模式执行流水线统筹三个过滤阶段，并利用 AVX-512 向量化和缓存优化结构，在维度增加时保持高性能。

阶段1：候选项生成。查询首先访问基于 CSR 的倒排索引，以累积碰撞分数。索引的线性化特性确保此阶段在累积分数时最大限度地利用内存带宽。评分机制会根据所选模式进行调整：

- **优化模式（$\phi = 1$）：**我们采用基于排名的加权评分方案。高排名单元格（前k个）中的碰撞将获得双倍权重（$w = 2$），优先考虑在最相关分区区域发生碰撞的候选项。
- **保证模式（$\phi = 0$）：**我们采用二进制碰撞计数（$w = 1$ 对所有碰撞计数）。

对于每个子空间，我们首先检索数据集中的一小部分（通常介于0.1%和6%之间），作为潜在候选项。然后，我们应用严格度阈值 τ，这是一个整数，表示候选项必须累积的最小子空间碰撞次数，只有达到该次数的候选项才会被保留在最终候选集C中。这种多阶段过滤确保后续细化阶段仅处理数据集中高度筛选的一小部分，从而在保持高召回率的同时最大限度地降低延迟。鲁棒性回退机制确保 $|\mathbf{C}| \ge k$，即使极端稀疏性导致候选项不足，也会通过检索排名最高的项目来完成。

阶段 2：BQ 加速细化。为优化候选集 C的处理顺序，我们采用条件执行路径。在优化模式（$\phi = 1$）下，我们使用 AVX-512 内部函数高效计算所有候选项的汉明距离 $\|b(q) - b(x)\|_H$。随后按距离对候选项排序，确保

最有希望的向量优先进行评估。这种优先级排序对于耐心机制至关重要，能让搜索迅速找到最近邻并触发提前终止。然而，在保证模式（$\phi = 0$）下，会完全跳过这一步细化，因为理论保证要求对所有候选项进行穷举验证。

阶段 3：模式特定验证。 候选项的最终精炼由用户可配置的执行标志 ϕ控制：

- **优化模式（$\phi = 1$）：** 我们采用 ADSampling，并结合动态终止（耐心）。搜索使用维度子集估计距离，若前k 个结果在连续 P 次验证中保持不变（默认 $P = 40 \cdot k$），则中止。

- **保证模式（$\phi = 0$）：** 耐心机制与 ADSampling 被禁用，因为需要对集合 C中的每个候选项计算精确的 L_2 距离。

4.4 复杂度分析

我们从三个关键维度分析 CRISP 的渐近复杂度：空间效率、构建成本与查询延迟。

空间效率。 CRISP 保持严格线性的内存占用。原始数据集需要 $O(ND)$的存储，而按 CSR 组织的倒排列表为逐子空间倒排索引额外增加 $O(NM)$，总空间复杂度为 $O(ND + NM)$。这与标准子空间划分方法的存储复杂度一致 [34], , 使 CRISP 与随机量化基线处于同一紧凑存储层级 [10]，并确保能够高效扩展到大规模数据集，而不会产生指针密集型结构所带来的过高内存开销。

自适应构建。 CRISP 的构建复杂度取决于数据，并由自适应启发式决定。对于各向同性分布，由于各维度之间本就只有较弱的相关性，CRISP 会跳过旋转，以保持较低的 $O(ND)$ 构建成本，该成本主要来自子空间量化。只有当谱检查检测到较强的特征相关性时，才会有选择地触发 $O(ND^2)$ 随机旋转，此时对齐带来的收益超过变换成本。通过原地执行旋转，我们将峰值内存占用控制在 ND 个浮点数，从而有效地将内存开销减半；相比之下，僵化的框架 [10, 11] 需要 $2ND$ 空间来存储旋转后的数据集副本。

查询延迟。 查询处理的主要开销来自候选项验证，其渐近复杂度为 $O(|\mathbb{C}| \cdot D)$，其中 $|\mathbb{C}|$ 为候选集大小。尽管不同模式下的最坏情况复杂度相同，但我们的优化模式通过 ADSampling 和基于耐心值的终止策略显著降低了代价，有效减少了 D 维距离计算的次数。

5 理论分析

与纯启发式索引方法不同，CRISP 在保证模式下运行时，可为检索质量提供严格的下界。

定理 5.1。 （条件召回率下界）。设 x^* 为查询 q 的真实最近邻，其单子空间碰撞概率为 p^*。设 τ 为由 $\tau = \alpha \cdot M$定义的选择阈值。如果 *CRISP* 配置为保证模式，则 x^* 成功检索到候选集中的概率下界为：

$$P(x^* \in C) \geq 1 - \exp\left(-\frac{2(Mp^* - \tau)^2}{M}\right)$$

前提是期望碰撞次数 $\mu = Mp^*$ 严格大于 τ。

证明。在保证模式下，算法会检索每个满足 $S_{col}(x,\ q) \geq \tau$ 的候选项 x。当且仅当最近邻的随机碰撞次数低于此阈值时，检索才会失败：

$$P_{fail} = P(S_{col}(x^*, q) < \tau)$$

假设各子空间提供独立证据，则随机变量 $S_{col}(x^*,\ q)$ 表示 M 个成功概率为 p^* 的独立伯努利试验之和。我们应用 *Hoeffding* 不等式 [14], , 它可以界定有界独立随机变量之和的尾概率。对于期望为 $E[S]$ 的和 S，Hoeffding 不等式指出：

$$P(E[S] - S \geq t) \leq \exp\left(-\frac{2t^2}{M}\right)$$

令 $S = S_{col}(x^*,\ q)$，其中 $E[S] = Mp^*$，并选择偏差 $t = Mp^* - \tau$（根据定理条件，$t > 0$）：

$$P(Mp^* - S_{col}(x^*, q) \geq Mp^* - \tau) \leq \exp\left(-\frac{2(Mp^* - \tau)^2}{M}\right)$$

这简化为 $P(S_{col}(x^*,\ q) \leq \tau)$。由于 $P_{fail} = P(S_{col} < \tau) \leq P(S_{col} \leq \tau)$，因此尾界可作为失败概率的有效上界。因此，检索概率 $1 - P_{fail}$满足所述下界。□

独立性假设的有效性。 尽管 Hoeffding 不等式假设各事件相互独立，但真实世界数据通常表现出较强的维度间相关性，从而违背这一假设。我们的自适应旋转起到结构校正的作用：通过应用随机正交变换，我们对特征空间进行去相关处理，确保跨子空间的碰撞表现为独立事件。这确保子空间碰撞满足误差界成立所需的独立性。

解释。该界表明，随着子空间数量 M 的增加，检索失败概率呈指数级下降，提供了比先前研究中使用的多项式界（例如切比雪夫界）更强的保证 [34]。此外，该界为稳健的回退机制（阶段 1）提供了理论依据：当$\tau \geq Mp^*$（违反定理条件）时，理论界将变得无效，因此需要采用基于分数的回退机制，以避免结果集为空。

6 实验分析

实验设置。 我们通过实验将 CRISP 与其他内存索引（RaBitQ [10], SuCo [34], OPQ [11], 和 HNSW [22]）进行比较。我们系统地测试了每种方法的关键参数，以优化搜索准确率与构建时间之间的权衡，

以及峰值内存使用。对于 OPQ 和 HNSW，我们使用 FAISS 库 [17], 提供的实现，而 RaBitQ 和 SuCo 则使用其官方开源实现。所有方法均以 C++ 实现，并在 Ubuntu 22.04 LTS 上使用 gcc（v11.4）编译。我们对所有基线使用了优化标志 -O3 和 -fopenmp，以确保多线程执行。所有方法均使用其可用的最优标志进行编译：CRISP、SuCo 和 FAISS（OPQ/HNSW）使用 -march=native 和 -mavx512f，RaBitQ 使用 -mavx2。实验在一台配备第 11 代 Intel® CoreTM i7-11700K 处理器（3.60 GHz）、32 GB RAM 以及 AVX-512 支持的机器上进行。对于每种对比方法，我们将参数范围扩展到原始文档建议的范围之外，以便在多种运行条件下全面评估性能。

CRISP. 我们改变了数据子空间结构（通过调整子空间数量，在每个数据集上测试五种配置）、控制查询覆盖率的碰撞率 α（范围为 $[0.001, 0.06]$），以及最小碰撞百分比（范围为 $[0.1, 0.6]$）；该百分比控制候选项需要满足多少次子空间碰撞才能入选。我们测试了 CRISP 的两种模式：保证模式（具有理论保证并进行穷举验证）和优化模式（采用加权评分并进行提前终止）。

SuCo. 我们测试的碰撞率 α 范围为 $[0.02, 0.06]$，候选比例 β 范围为 $[0.003, 0.006]$，并采用了与 CRISP 相同的五种子空间配置。这些范围旨在覆盖并略微扩展作者建议的 $\alpha \in [0.03, 0.05]$ 和 $\beta \in [0.001, 0.005]$ “最佳范围”。通过扩展这些边界，我们确保 SuCo 在每个数据集上的最优性能点得以实现，同时遵循论文的指导原则 [34]。我们在这些参数区间内的所有配置上评估了 SuCo，仅保留召回率-QPS和召回率-构建时间权衡曲线上的帕累托最优点。

RaBitQ.我们测试了多种簇规模（N_{list}），范围从32到4,096，覆盖基准测试中数据集基数的广泛差异，并包含作者针对百万级数据集推荐的 $N_{\text{list}} = 4{,}096$配置。我们进一步比较了快速训练（2次K均值迭代）与标准训练（20次迭代），以评估随机码本构建的稳健性。为确保全面的帕累托前沿，我们以2的幂测试了查询探测数（N_{probe}），从1直至簇的总数。

HNSW. 我们测试了最高达 $M = 64$的图密度，超出推荐的最大值48，以评估在可能需要更高连通性的极高维场景中的性能。我们还在实际范围内变化了构建搜索深度（efConstruction $\in \{32, 64, 128\}$）和查询搜索深度（efSearch $\in \{32, 64, 128, 256\}$）。我们使用了FAISS实现 [17]。

OPQ。我们将子量化器数量 M 设为与五个子空间划分相匹配，因为 OPQ 要求 M 能够整除数据维度。我们测试了每个子向量采用 8 位（FAISS 默认值）和 6 位（文档中说明可用于内存受限场景的较低精度选项）时的编码精度。我们还测试了 OPQ 训练迭代次数为 FAISS 默认的 50 次和较低的 20 次，以评估不同数据集上的收敛敏感性。索引构建每次运行的超时时间设为 1 小时；超过此限制的运行均记为失败。

6.1 数据集

为评估 CRISP 在多种高维场景下的表现，我们选取了九个数据集，涵盖不同的基数、模态和内在维度。表2列出了数据集的具体信息，以及我们获取各数据集和相应查询集的来源。我们使用 Levina & Bickel 的最大似然估计（MLE）估算器 [20]，基于随机抽取的 1,000 个查询及其 $k = 100$ 个邻居，计算每个数据集的局部内在维度（LID）。LID 衡量数据邻域结构的有效维度，不受原始嵌入维度 D的影响。

表2：数据集特征和局部内在维度

数据集	Type	基数（N）	D	查询数（Q）	k_{size}	LID
Gist [2]	图像	1,000,000	960	1,000	100	44.81
Simplewiki-OpenAI [19]	Text	260,372	3,072	1,000	100	27.49
Trevi [19]	图像	90,120	4,096	1,000	100	24.85
Ccnews-nomic [19]	Text	495,328	768	1,000	100	22.94
Agnews-mxbAI [19]	Text	769,382	1,024	1,000	100	20.92
ImageNet [19]	图像	1,281,167	640	1,000	100	20.54
Gooaq-distilroberta [19]	Text	1,475,024	768	1,000	100	17.09
Fashion-MNIST [19]	图像	60,000	784	10,000	100	15.26
MNIST [19]	图像	69,000	784	200	100	14.07

如表2所示，**Gist** 是最具挑战性的数据集（局部内在维度 44.8，$D = 960$），远超其中等偏高的水平。**Simplewiki- OpenAI** 和**Trevi** 紧随其后（局部内在维度分别为 ≈ 27.5 和 24。9），Trevi 较高的 $D = 4,096$ 在很大程度上是由维间相关性而非真正的复杂性所致。**Ccnews-nomic**、**Agnews-mxbAI**和**ImageNet** 处于中间范围（局部内在维度 $\approx$ 20–23），体现了典型的密集检索嵌入；而**Gooaq-distilroberta** 略低一些（局部内在维度 ≈ 17.1），尽管其基数很大，拥有 147 万个向量。**MNIST** 和**Fashion-MNIST** 处于低端（局部内在维度 $\approx$ 14–15），这与它们众所周知的流形结构一致。

6.2 索引扩展性与检索质量

我们通过计算每种方法的 Recall@100 与构建时间之间的帕累托前沿，评估索引构建的代价。图4展示了达到横轴所示召回率（至少达到该水平）所需的最低构建时间。该实验评估了检索准确率与索引代价之间的权衡。

构建效率。CRISP 的构建成本在不同召回率下几乎保持不变。在**Trevi** ($D = 4096$)上，CRISP 的召回率可从 85% 提升至 99.5%，构建成本仅需 49s–53s，因为其索引构建是固定成本操作，不受搜索时参数（候选比例、最低分数）影响。SuCo 采用简单的分区方案，构建时间最快，仅为 ~5s，但在该数据集上的最高召回率只有 86%。相比之下，HNSW 的构建成本随召回率上升而大幅增加：召回率为 92% 时需 49s，升至 99.4% 时则需 634s，增加了 13× 。RaBitQ 在所有召回率下的构建时间固定为 ~30s，因为其构建成本取决于簇的数量，而控制召回率的参数 NProbe 仅在搜索时使用，不影响构建。在**ImageNet** ($D = 640$)上，CRISP 在所有召回率水平下均可于 14s–15s 内完成索引构建，最高召回率达 98.6%。HNSW 的帕累托前沿似乎

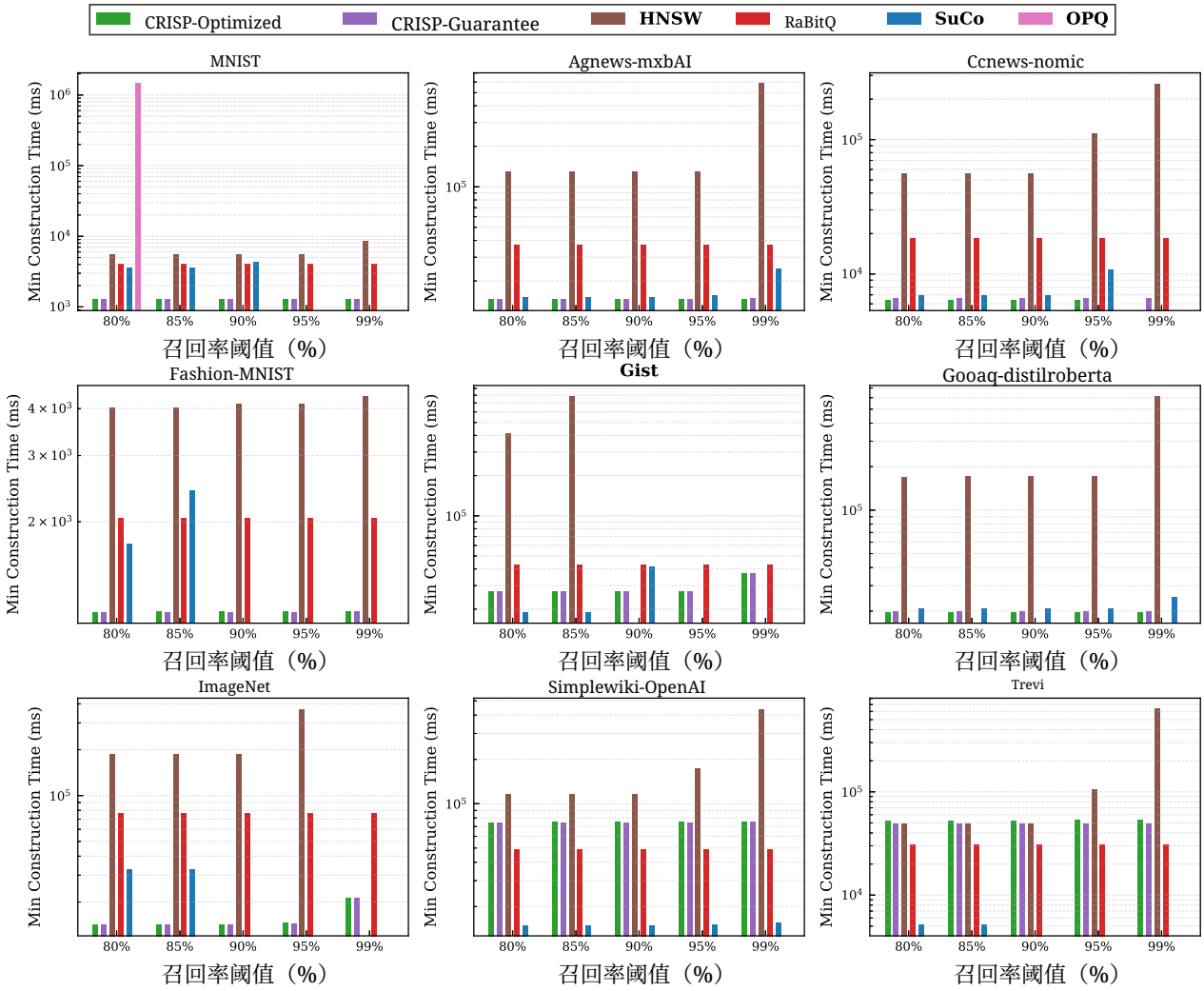


图4：在九个基准数据集上达到特定 Recall@100 阈值（80%、85%、90%、95%、99%）所需的最短索引构建时间（纵轴为对数刻度）。缺失的柱状条表示该方法在所测试的配置下无法达到所需的召回率阈值。越低越好。

仅在该数据集上召回率超过 93% 时出现，构建耗时为 187–395s，具体取决于图连通度。RaBitQ 的召回率可达 99.3%，但需要 77s；SuCo 则以 86% 的召回率上限为代价，将构建成本维持在较低水平（33s）。在**Gooaq-distilroberta**上，由于构建期间内存需求过高，无法评估 RaBitQ。

在**MNIST** ($D = 784$)上，所有方法都实现了较高的召回率。CRISP 的构建在 ~1.3 秒内完成，召回率最高可达 99.9%。SuCo 的速度也相近，为 3.5–4.4 秒，但其召回率停滞在 ~92%。HNSW 需要 5.5–8.6 秒，而 RaBitQ 需要 ~4 秒。即使在这个结构良好的数据集上，在相近的召回率水平下，CRISP 的构建速度也比 HNSW 快 4–7×，比 RaBitQ 快 ~3×。图4中仅展示了 OPQ 在 MNIST 上的结果，因为在相近的召回率下，它耗时约 24 分钟，比 CRISP 长 1、100×。在其余所有数据集上，OPQ 要么超时，要么在任何测试配置下都未能达到 80% 的召回率阈值。在所有数据集上，CRISP 构建耗时曲线平坦，源于其设计：构建代价主要来自对数据进行一次遍历以完成子空间编码；当无需旋转时，谱检查带来的额外开销可以忽略不计（约占总构建成本的 2.5%）。SuCo 的构建同样轻量，但在谱能量高度集中的数据集（高 CEV）上无法实现较高召回率，例如**Gist** 和**Fashion-MNIST**；这是因为独立的子空间划分无法捕捉数据真实的底层分布。HNSW 的图构建成本随连通性（M）和候选池大小（efConstruction）超线性增长，而 RaBitQ 的 k-means 训练成本则随簇数量增加。

内存效率。 表 3 报告了各方法在其最高召回率帕累托配置下搜索阶段的常驻集大小（RSS）。虽然 CRISP 和 SuCo 的渐近内存占用均为 $O(ND + NM)$，但在实际应用中，CRISP 的内存需求始终低 ≈1.85×

表 3：每种方法在各数据集上的搜索阶段峰值常驻集大小（GB）。

数据集	CRISP	SuCo [34]	RaBitQ [10]	HNSW [22]
Agnews-mxbAI	**3.57**	6.6	6.41	6.47
Ccnews-nomic	**1.73**	3.2	3.11	3.24
Gist	**4.35**	8.03	7.80	7.78
Gooaq-distilroberta	**5.13**	9.18	–	9.44
ImageNet	**3.79**	6.78	6.67	7.08
Simplewiki-OpenAI	**3.34**	6.4	6.59	6.38
Trevi	**1.50**	2.95	3.19	2.96
MNIST	**0.24**	0.46	0.47	0.47
Fashion-MNIST	**0.25**	0.46	0.48	0.45
空间复杂度	$O(ND+NM)$	$O(ND+NM)$	$O(ND+NB+K_qB)$	$O(ND+NM_{HNSW}\log N)$

于 SuCo。RAM。这种效率主要归功于我们采用了缓存一致的压缩稀疏行（CSR）倒排索引，从而消除了传统基于哈希的倒排列表所带来的元数据开销（即碎片和指针）。HNSW 的内存复杂度为 $O(ND + NM_{HNSW}\log N)$，其中 M_{HNSW}表示每个节点的双向链接数；因此，为了在具有挑战性的数据集上实现高召回率而提高图连通度，会使图存储和构建期内存开销都线性增加。RaBitQ 除存储维度为 B 的二进制量化码和 K_q 个聚类质心外，还存储全精度数据集，其内存占用为 $O(ND + NB + K_qB)$；此外，在聚类训练阶段还会产生显著的峰值内存使用。如前所述，RaBitQ 在 **Gooaq-distilroberta** 上构建时内存耗尽。

总之，CRISP 在所有测试数据集上实现了最低的内存占用——在渐近复杂度上与 HNSW 和 RaBitQ 相当，并且尽管底层算法结构相似，仍远低于 SuCo——这得益于精细的内存管理。

6.3 检索性能

近似最近邻搜索索引设计中的主要权衡在于查询吞吐量（以每秒查询数（QPS）衡量）与检索准确率（Recall@100）之间。图5表明，CRISP 在各类数据集上取得了具有竞争力或更优的性能。我们评估了 CRISP 的两种模式，分别记为 CRISP-Optimized 和 CRISP-Guarantee。OPQ 被省略，因为它在任何数据集上要么超时，要么未能达到 85% 的召回率阈值。

CRISP 的优势在最高维度的设置中最为显著。在**Trevi** ($D = 4096$)上，CRISP-Optimized 远超所有方法：在 95% 召回率下，速度是排名第二方法的 2.95× 倍（2,463 QPS，而 HNSW 为 834 QPS）；在 99% 召回率下，速度是 6.6× 倍（1,751 QPS，而 HNSW 为 267 QPS）。值得注意的是，即使是 CRISP-Guarantee，在 Trevi 上的表现也优于 HNSW，在 95% 召回率下达到 1,206 QPS，在 99% 召回率下达到 746 QPS。在 **Simplewiki-OpenAI** ($D = 3072$)上，CRISP-Optimized 在 95% 召回率下达到 2,137 QPS，而 HNSW 为 1,080 QPS，RaBitQ 为 559 QPS。在 99% 召回率下，CRISP-Optimized 仍达到 1,319 QPS，高于 HNSW 的 458 QPS 和 RaBitQ 的 245 QPS；CRISP-Guarantee 的表现也颇具竞争力，达到 673 QPS。在这两种情况下，CRISP 连续的 CSR内存布局和顺序扫描模式，都避免了当向量数据超出 L2缓存时，图遍历所受的指针追逐开销影响。

在**ImageNet** ($D = 640$)上，性能会随召回率阈值而变化。在较低的召回率阈值（≤95%）下，HNSW 的吞吐量最高。然而，随着召回率目标提高，CRISP-Optimized 在召回率达到 99% 时可实现 243 QPS，比 RaBitQ 的 91 QPS 高出，而 HNSW 则无法达到这一阈值。值得注意的是，CRISP-Guarantee 的性能与之相当，在召回率达到 99% 时可实现 241 QPS，这表明在高召回率目标下，严格的界限并不会牺牲吞吐量。CRISP 也是该数据集上唯一能达到 ≥99.5% 召回率的方法。

在 **Agnews-mxbAI** ($D = 768$) 上，在大多数阈值下，CRISP 仅次于 HNSW 位居第二；在 99.5% 召回率时，CRISP-Optimized 达到 989 QPS，而 HNSW 为 1,021 QPS，CRISP-Guarantee 则为 745 QPS。CRISP 的两种模式始终优于 RaBitQ，最高可达 2×。在 **Ccnews-nomic** ($D = 768$) 和 **Gooaq-distilroberta** ($D = 768$) 上，HNSW 在所有召回率水平上都保持吞吐量优势，而 CRISP 分别在低于 95% 和 99% 的召回率水平上位居第二。

在结构良好的数据集 **MNIST** ($D = 784$) 和 **FashionM-NIST** 上，CRISP 在所有召回率水平上都领先，或紧随 HNSW 位居第二。CRISP-Guarantee 仍保持很高的性能，在大多数召回率水平上优于 RaBitQ 和 SuCo。

在**Gist** （$D = 960$）上，这是一个维度间相关性很强的数据集，大多数方法都难以实现较高的召回率。无论是 HNSW 还是 SuCo，在任何实际可行的吞吐量水平下都无法达到 95% 的召回率（图 5）。CRISP-Optimized 在 799 QPS 时达到 95% 的召回率，而 RaBitQ 为 718 QPS；在 97% 的召回率下，CRISP-Optimized 的吞吐量为 707 QPS，RaBitQ 则为 571 QPS。CRISP-Guarantee 也颇具竞争力，在 95% 和 97% 召回率下分别达到 499 QPS 和 450 QPS。

6.4 消融实验

为了评估 CRISP 中设计选择和参数选择的影响，我们在四个数据集上进行消融实验，这些数据集涵盖了多种内在维度、谱结构和语义领域。具体而言，**Fashion-MNIST** 和**Gist** 均具有较高的谱能量集中度（CEV 分别为 $= 0.94$ 和 0.91），但两者的内在维度差异显著（15.3 对比 44.8），因此可作为互补的压力测试，用于评估自适应旋转机制；**Simplewiki-OpenAI**涵盖了维度极高的嵌入空间（$D = 3{,}072$，LID 为 ≈ 27.5）；而 **Ccnews-nomic** 代表近似各向同性的分布，其方差均匀分布于各维度（CEV 为 $= 0.77$，LID 为 ≈ 22.9）。

自适应旋转阈值的敏感性分析。我们的子空间划分效果取决于数据集与主坐标轴的对齐程度。为了确定触发全局旋转的最佳阈值，我们评估了一系列 τ_{cev} 值。如图6所示，$\tau_{cev} = 0.85$ 带来了最出色的整体性能，为区分哪些数据集受益于正交旋转、哪些数据集不受益提供了有效的决策边界。对于内在结构在高维空间中形成紧密局部流形的数据集，该阈值能够正确触发变换。提高阈值以抑制这些数据集上的旋转，会显著降低检索质量。相反，对于方差分散在多个维度上的数据集，正交旋转带来的准确率提升微乎其微，却会产生不容忽视的查询时间开销。

耐心因子的敏感性分析。CRISP-Optimized 的提前终止机制以搜索准确率换取查询吞吐量。我们评估了此处使用的耐心因子 P

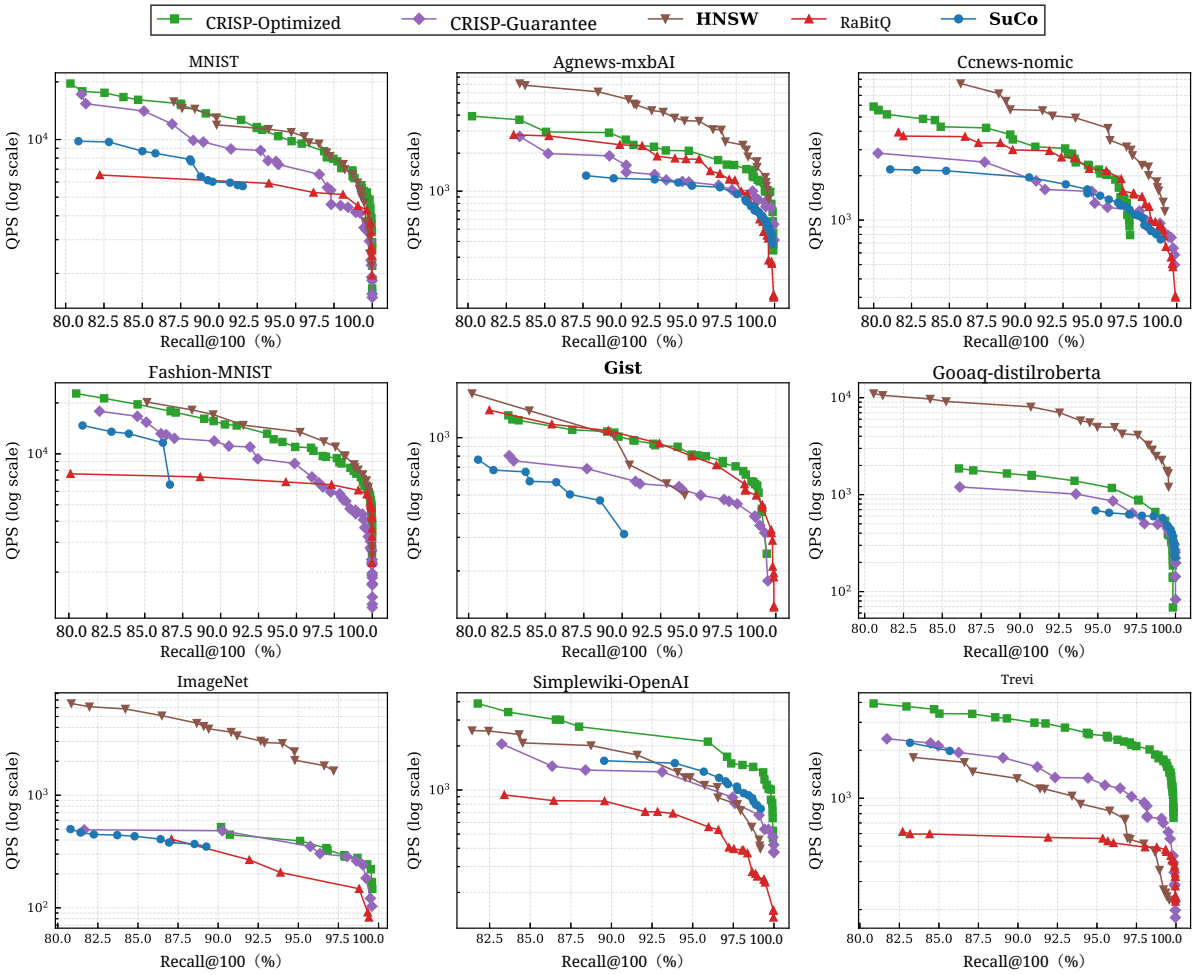


图5：九个基准数据集上的 Recall@100 与 QPS 帕累托前沿（y 轴为对数刻度）。每个子图展示了各方法的最优吞吐量-准确率权衡。越高、越靠右越好。

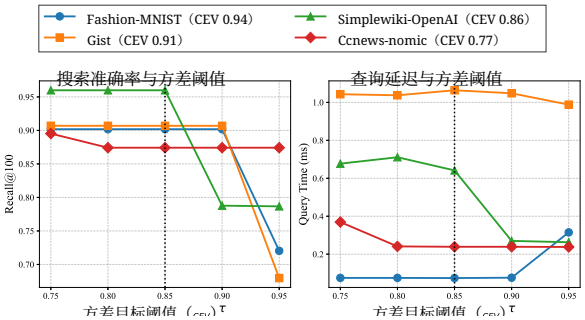


图6：方差目标阈值（τ_{cev}）对代表性数据集上（a）搜索准确率（Recall@100）和（b）查询延迟（毫秒）的影响。

权衡参数，即在更新前连续检查的候选项数量，更新内容为前k项结果，并测试提前终止。如图8所示，我们的结果表明，40 是最有效的该参数在测试数据集上的取值。请注意，将 P 从 20 增加到 40，几乎能显著提高所有数据集的召回率；然而，进一步将该因子增加到 60 则不会带来额外收益。另一方面，正如预期，查询延迟会随 P 线性增加，因此 $P = 40$ 是实测的最佳值。

多阶段过滤流水线的影响。CRISP 查询流水线采用两个组件来加速最终验证阶段：用于候选项优先级排序的二进制汉明重排序，以及用于近似距离计算的 ADSampling。为评估这两个组件，我们将完整流水线与两种配置进行比较：一种使用汉明重排序和精确 L2 距离（禁用 ADSampling），另一种使用 ADSampling，但不进行预先的汉明重排序。如图7所示，这两个组件对于实现最佳召回率–每秒查询数帕累托前沿都至关重要。

结果显示，ADSampling 是查询吞吐量的主要驱动因素；例如，对于 **Gist** 在 90% 召回率下，启用 ADSampling 相对于精确 L2 距离计算可将吞吐量提高 2× 以上。这些收益对召回率的代价可忽略不计，因为 ADSampling 的自适应剪枝能有效跳过冗余维度

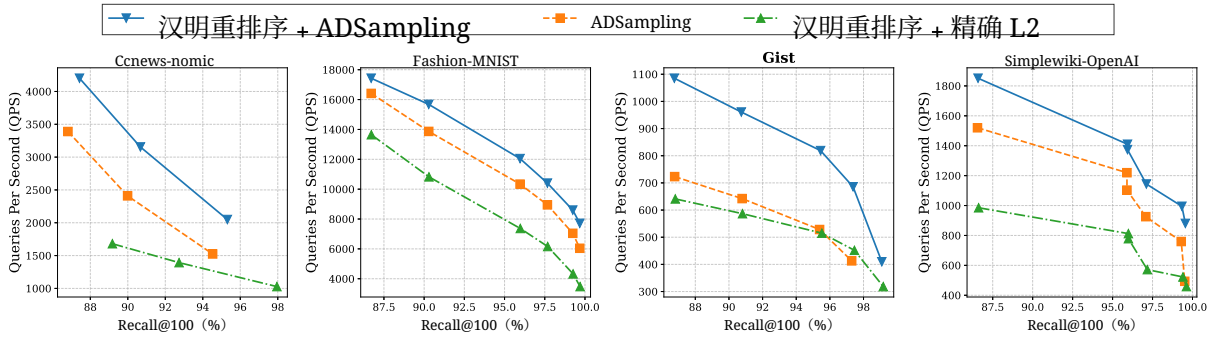


图7：过滤子组件的消融实验，展示了 (a) Ccnews-nomic、(b) Fashion-MNIST、(c) Gist 和 (d) Simplewiki-OpenAI 的召回率-QPS帕累托前沿。

在距离计算过程中。与此同时，基于耐心的提前终止的有效性关键取决于候选项的评估顺序。关闭汉明重排序会导致可测得的性能下降。在 **Gist** 上，移除汉明重排序阶段而保留 ADSampling，会在同等召回率水平下使吞吐量下降 22%。这证实了二进制汉明码作为欧几里得邻近度的低代价有效代理的有效性。

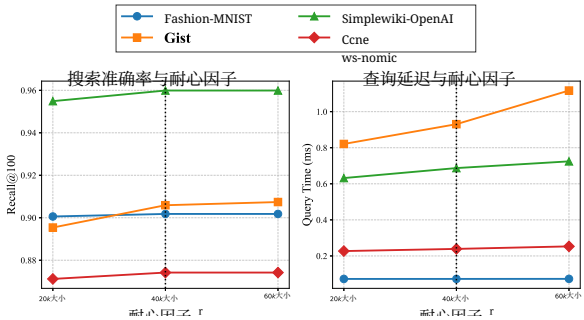


图8：耐心因子对代表性数据集上 (a) 搜索准确率（Recall@100）和 (b) 查询延迟（毫秒）的影响。

6.5 实验分析总结

在所有基准测试中，CRISP 在极高维场景下展现出最为显著的优势。在维度为 $D \geq 3072$的数据集（**Trevi**、**Simplewiki-OpenAI**）上，CRISP-Optimized 在 99% 召回率下的吞吐量最高可比 HNSW 高出 6.6×，同时构建时间低一个数量级。在 $D \approx 960$ （**Gist**）上，强烈的维间相关性导致 HNSW 和 SuCo 都完全无法达到高召回率，而 CRISP 仍是唯一能在实用吞吐量水平下超过 97% 召回率的方法。除检索性能外，CRISP 在所有测试数据集上始终保持最低的搜索阶段内存占用，尽管算法结构相似，所需 RAM 仍比 SuCo 少 ≈1.85×，并且比 RaBitQ 和 HNSW 都更为紧凑。在构建方面，CRISP 相较所有方法均表现出更优或具有竞争力的性能，但在 **Simplewiki-OpenAI** 和 **Trevi** 上除外，其中 RaBitQ 和 SuCo 的构建速度更快，但与 CRISP 相比，其 QPS 较低。

在低维场景（$D \leq 768$）中，CRISP 的局限性开始显现，此时基于图的方法仍具有很强的竞争力。尽管 CRISP 在这些数据集上的表现始终优于 RaBitQ 和 SuCo，且在 **MNIST** 和 **Fashion-MNIST** 上与 HNSW 不相上下，但在 **Ccnews-nomic** 和 **Gooaq-distilroberta** 上则落后于 HNSW；在大多数召回率水平下，HNSW 都保持着明显的吞吐量优势。具体而言，在 **Ccnews-nomic** 上，CRISP-Optimized 的召回率最高可达 97%。类似地，在 **ImageNet** （$D = 640$）上，HNSW 在中等召回率阈值下领先；但当召回率超过 97.5% 时，CRISP-Optimized 成为表现最佳的方法，也是唯一能在该数据集上达到 99.5 % 召回率的方法。

7 结论与未来工作

在本工作中，我们提出了 CRISP，一种面向极高维数据的高性能索引框架，它将相关性感知预处理与缓存高效数据结构和向量化查询处理相结合。尽管我们的结果在多种基准测试中展现出强劲性能，以下方向仍有待探索，以扩展该框架：

自适应子空间分解。CRISP 目前将特征空间划分为大小相等的子空间（例如，将 128 个维度划分为 8 个各含 16 个维度的子空间）。这种均匀划分忽略了某些维度比其他维度携带更多信息这一事实。一个尚未探索的思路是根据方差调整子空间大小：在需要更精细量化的高方差区域分配更少的维度，而在低方差区域将更多维度组合在一起。这将在不增加内存使用量的情况下降低偏斜数据集上的量化误差。

部分方差重分配。目前，当检测到相关性时，CRISP会对整个特征空间应用全局旋转。一种有前景的优化是分块方差重分布，即仅对表现出高谱能量集中度的特定维度子集应用旋转。该方法在目标子空间集合内局部重分配方差，而非在整个向量范围内重分配，从而将预处理复杂度从 $O(ND^2)$降至 $O(ND \cdot d_{\text{sub}})$，其中 $d_{\text{sub}} \ll D$。因此，在仅特定特征子空间相关的混合分布数据集中，可以实现更细粒度的适应性。

参考文献

[1] Cecilia Aguerrebere、Ishwar Singh Bhati、Mark Hildebrand、Mariano Tepper与Theodore Willke。2023。基于压缩索引的转瞬相似性搜索。《*VLDB*基金会会刊》16, 11 （2023 年 7 月），3433–3446。https: //doi.org/ 10 14778 3611479 3611537 . / . Laurent Amsaleg与Hervé Jégou。2026。近似最近邻搜索数据集。http://corpus-texmex.irisa.fr/。（访问时间：2026年2月）。[3] Martin Aumüller、Erik Bernhardsson与Alexander John Faithfull。2017。ANN-Benchmarks：面向近似最近邻算法的基准测试工具。载于第*10*届国际相似性搜索与应用会议（*SISAP*）论文集。施普林格，34–49。https://doi.org/10.1007/978-3-319- 68474-13[4] Martin Aumüller与Matteo Ceccarello。2023。近似最近邻搜索的最新方法与趋势及基准测试评述。*IEEE*数据工程通报 47, 3（2023），89–105。http://sites.computer.org/debull/A23sept/ p89.pdf [5] Artem Babenko与Victor Lempitsky。2012。倒排多重索引。载于*IEEE*计算机视觉与模式识别会议（*CVPR*）论文集。IEEE, 3069 3076 10 1109 2012 6248038 – 。https://doi.org/ . / CVPR. . [6] Kevin S. Beyer、Jonathan Goldstein、Ramakrishnan Ramakrishnan与Uri Shaft。1999*99, 71012, 1999*, 。“最近邻”何时有意义？。载于数据库理论——*ICDT*第届国际会议，以色列耶路撒冷，*1*月——论文集（计算机科学讲义），第1540卷。施普林格，217–235。https://doi. org/ 10 1007 3 540 49257 7_15 . / - - - Meng Chen、Kai Zhang、Zhenying He、Yinan Jing与X. Sean Wang。2024。RoarGraph：用于高效跨模态近似最近邻搜索的投影二部图。《*VLDB*基金会会刊》 17, 11（2024），2735–2749。https: //doi.org/10.14778/3681954.3681959[8] Qi Chen、Bing Zhao、Haidong Wang、Mingqin Li、Chuanjie Liu、Zengzhong Li、Mao Yang与Jingdong Wang。2021。SPANN：高效的十亿规模近似最近邻搜索。载于神经信息处理系统进展*34*：*2021*年神经信息处理系统年度会议，*NeurIPS 2021*，*2021*年*12*月6日至*14*日，线上，Marc'Aurelio Ranzato、Alina Beygelzimer、Yann N. Dauphin、Percy Liang与Jennifer Wortman Vaughan（编）。5199–5212。https://proceedings.neurips.cc/paper/2021/hash/299dc35e747eb77177d9cea10a802da2-Abstract.html[9] Jianyang Gao与Cheng Long。2023。高维近似最近邻搜索：可靠且高效的距离比较操作。*ACM*数据管理汇刊 1, 2 2023 137 1 137 27 10 1145 3589282 ( ), : – : . https://doi.org/ . / Jianyang Gao与Cheng Long。. RaBitQ：面向近似最近邻搜索的带理论误差界的高维向量量化。*ACM*数据管理汇刊 2, 3 2024 167 10 1145 3654970 ( ), . https://doi.org/ . / Tiezheng Ge、Kaiming He、Qifa Ke与Jian Sun。. 面向近似最近邻搜索的优化乘积量化。载于*2013*年*IEEE*计算机视觉与模式识别会议，美国俄勒冈州波特兰，*2013*年*6*月*23*日至*28*日。IEEE计算机学会, 2946 2953 10 1109 2013 379 – 。https://doi.org/ . /CVPR. . Yunchao Gong、Svetlana Lazebnik、Albert Gordo与Florent Perronnin。2011。迭代量化：面向大规模图像检索的学习二进制编码的普罗克鲁斯忒斯方法。载于*IEEE*计算机视觉与模式识别会议（*CVPR*）论文集。IEEE，817–824。https://doi.org/10.1109/ CVPR.2011.5995432[13] Ruiqi Guo、Philip Sun、Erik Lindgren、Quan Geng、David Simcha、Felix Chern与Sanjiv Kumar。2020。利用各向异性向量量化加速大规模推理。载于第*37*届国际机器学习会议论文集，*ICML 2020*（机器学习研究论文集），第119卷。PMLR，3887–3896。http://proceedings.mlr.press/v119/guo20h.html [14] Wassily Hoeffding。1963。有界随机变量之和的概率不等式。《美国统计协会杂志》58, 301（1963），13–30。https://doi.org/10.1080/ 01621459.1963.10500830[15] Qiang Huang、Jianlin Feng、Yunjun Zhang、Qiong Fang与Wilfred Ng。2015。面向近似最近邻搜索的查询感知局部敏感哈希。《*VLDB*基金会会刊》 9, 1（2015），1–12。[16] Herve Jegou、Matthijs Douze与Cordelia Schmid。2011。面向最近邻搜索的乘积量化。《*IEEE*模式分析与机器智能汇刊》 33, 1（1月 2011 117 128 10 1109 2010 57），– 。https://doi.org/ . /TPAMI. . Jeff Johnson、Matthijs Douze与Hervé Jégou。. 基于GPU的十亿规模相似性搜索。*IEEE*大数据汇刊 7, 3（2019），535–547。[18] Leonardo Kuffó、Elena Krippner与Peter Boncz。2025。PDX：面向向量相似性搜索的数据布局。*ACM*数据管理汇刊 3, 3（2025），196:1–196:26。https://doi.org/ 10 1145 3725333 . / Leonardo Kuffo、Elena Krippner、Peter Boncz。2026。PDX：公开数据。https://drive.google.com/drive/u/1/folders/1f76UCrU52N2wToGMFg9ir1MY8ZocrN34。（访问时间：2026年2月）。[20] Elizaveta Levina与Peter J. Bickel。2004。内在维度的最大似然估计。载于神经信息处理系统进展*17*（*NIPS 2004*）。麻省理工学院 P ress, – 777 784.

[21] Wen Li、Ying Zhang、Yifang Sun、Wei Wang、Mingjie Li、Wenjie Zhang 和 Xuemin Lin。2020。高维数据上的近似最近邻搜索——实验、分析与改进。*IEEE*知识与数据工程汇刊32, 8 (2020), 1475–1488。https://doi.org/10.1109/TKDE. 2019.2909204[22] Yury A. Malkov 和 Dmitry A. Yashunin。2020。使用分层可导航小世界图进行高效、稳健的近似最近邻搜索。*IEEE*模式分析与机器智能汇刊42, 4 (2020), 824–836。https://doi.org/10.1109/ TPAMI.2018.2889473[23] John Paparrizos、Ikraduya Edian、Chunwei Liu、Aaron J. Elmore 和 Michael J. Franklin。2022。通过方差感知量化实现快速自适应相似性搜索。载于第*38*届*IEEE*国际数据工程会议（*ICDE 2022*），马来西亚吉隆坡，*2022*年*5*月*9*日至*12*日。IEEE，2969–2983。https://doi.org/10. 1109/ICDE53745.2022.00268[24] Suhas Jayaram Subramanya、Fnu Devvrit、Harsha Vardhan Simhadri、Ravishankar Krishnaswamy 和 Rohan Kadekodi。2019。DiskANN：在单个节点上进行快速、精确的十亿点最近邻搜索。载于神经信息处理系统进展*32*：*2019*年神经信息处理系统年会（*NeurIPS 2019*）。13740–13750。[25] Ji Sun、Guoliang Li、James Pan、Jiang Wang、Yongqing Xie、Ruicheng Liu 和 Wen Nie。2025。GaussDB-Vector：面向大语言模型应用的大规模持久化实时向量数据库。*PVLDB*18, 12 (2025), 4951–4963。https://dblp.org/rec/journals/pvldb/SunL0XRLN25.bib [26] Toni Taipalus。2024。向量数据库管理系统：基本概念、用例与当前挑战。认知系统研究85, C (2024年6月), 8。https://doi.org/ 10 1016 2024 101216 . /j.cogsys. . Yao Tian、Xi Zhao 和 Xiaofang Zhou。2023。DB-LSH 2.0：采用基于查询的动态分桶的局部敏感哈希。*IEEE*知识与数据工程汇刊 35, 12 2023 13076 13088 10 1109 2023 3263728 ( ), – 。https://doi.org/ . /TKDE. . Jun Wang、Wei Liu、Sanjiv Kumar 和 Shih-Fu Chang。. 面向大数据索引的学习哈希——综述。*IEEE*会刊104, 1 (2016), 34–57。https://doi.org/ 10 1109 2015 248797 . /JPROC. . Jingdong Wang、Ting Zhang、Jingkuan Song、Nicu Sebe 和 Heng Tao Shen。2018。学习哈希综述。*IEEE*模式分析与机器智能汇刊 40, 4 2018 769 790 10 1109 2017 2699960 ( ), – 。https://doi.org/ . /TPAMI. . Mengzhao Wang、Weizhi Xu、Xiaomeng Yi、Songlin Wu、Zhangyang Peng、Xiangyu Ke、Yunjun Gao、Xiaoliang Xu、Rentong Guo 和 Charles Xie。2024。Starling：面向数据段上高维向量相似性搜索的高效I/O磁盘驻留图索引框架。*ACM*数据管理会刊 2, 1 2024 2 014 1 2 014 27 10 1145 3639269 ( ), V mod : –V mod : . https://doi.org/ . / Mengzhao Wang、Xiaoliang Xu、Qiang Yue 和 Yuxiang Wang。. 基于图的近似最近邻搜索的综合综述与实验比较。*PVLDB*14, 11 (2021), 1964–1978。https://doi.org/10.14778/3476249.3476255[32] Zeyu Wang、Qitong Wang、Xiaoxing Cheng、Peng Wang、Themis Palpanas 和 Wei Wang。2024。Steiner难度：基于图的近似最近邻索引的查询难度度量。*PVLDB*17, 13 (2024), 4668–4682。https://doi.org/10. 14778/3704965.3704974[33] Roger Weber、Hans-Jörg Schek 和 Stephen Blott。1998。高维空间中相似性搜索方法的定量分析与性能研究。载于*VLDB'98*：第*24*届超大型数据库国际会议论文集，*1998*年*8*月*24*日至*27*日，美国纽约州纽约市，Ashish Gupta、Oded Shmueli 和 Jennifer Widom（编）。Morgan Kaufmann，194–205。http: //www.vldb.org/conf/1998/p194.pdf [34] Jiuqi Wei、Xiaodong Lee、Zhenyu Liao、Themis Palpanas 和 Botao Peng。2025。子空间碰撞：一种高效、精确的高维近似最近邻搜索框架。*ACM*数据管理会刊3, 1 (2025), 79 1 79 29 10 1145 3709729 : – : . https://doi.org/ . / Jiuqi Wei、Botao Peng、Xiaodong Lee 和 Themis Palpanas。2024。DET-LSH：一种采用动态编码树的近似最近邻搜索局部敏感哈希方案。*PVLDB*17, 9 (2024), 2241–2254。https://doi.org/10.14778/3665844.3665854[36] Qianxi Zhang、Shuotao Xu、Qi Chen、Guoxin Sui、Jiadong Xie、Zhi Cai、Yaoqi Chen、Yinxuan He、Yuqing Yang、Fan Yang、Mao Yang 和 Lidong Zhou。2023。VBASE：通过松弛单调性统一在线向量相似性搜索与关系查询。载于*USENIX*操作系统设计与实现研讨会。https://api.semanticscholar.org/CorpusID:259859049[37] Bolong Zheng、Xi Zhao、Lianggui Weng、Nguyen Quoc Viet Hung、Hang Liu 和 Christian S. Jensen。2020。PM-LSH：一种快速、精确的高维近似近邻搜索局部敏感哈希框架。*PVLDB*13, 5 (2020), 643–655。https://doi.org/10.14778/3377369.3377374