软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(3):971−1005 [doi: 10.13328/j.cnki.jos.007516] [CSTR: 32375.14.jos.007516]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

向量数据库中近似最近邻搜索关键技术综述*

宋子文, 王 斌, 张喜瑞, 赵世豪, 杨晓春

(东北大学 计算机科学与工程学院, 辽宁 沈阳 110819)

通信作者: 王斌, E-mail: binwang@mail.neu.edu.cn

摘 要: 高维向量近似最近邻搜索 (approximate nearest neighbor search, ANNS) 是向量数据库的基础和核心之一. 随着人工智能的发展, 向量数据库发挥了日益关键的作用, 获得了广泛的关注, 高效的 ANNS 方法对向量数据库的性能优化十分关键. 在几十年的发展中, ANNS 取得了一系列成果. 近些年随着该领域的快速发展, 涌现出来的新方法和研究成果亟须系统性梳理. 首先介绍了 ANNS 的基本概念; 其次在已有的综述框架的基础上, 根据向量数据组织方式将当前的内容进一步归纳为基于图、层次、量化、哈希和混合数据组织这 5 类, 并结合代表性和最新的成果进行介绍; 然后从向量数据搜索优化方法的角度提出面向硬件加速、面向学习增强、面向距离比较操作、面向磁盘内存混合场景、面向数据访问优化、面向分布式场景、面向混合查询和理论分析这 8 个方面的分类体系对最近的搜索方法进行综述; 最后基于当前的研究成果和趋势, 展望未来的研究方向.

关键词: 近似最近邻搜索; 向量数据库; 索引; 高维数据; 向量搜索; 查询优化

中图法分类号: TP311

中文引用格式: 宋子文, 王斌, 张喜瑞, 赵世豪, 杨晓春. 向量数据库中近似最近邻搜索关键技术综述. 软件学报, 2026, 37(3): 971–1005. http://www.jos.org.cn/1000-9825/7516.htm

英文引用格式: Song ZW, Wang B, Zhang XR, Zhao SH, Yang XC. Survey on Key Techniques of Approximate Nearest Neighbor Search in Vector Databases. Ruan Jian Xue Bao/Journal of Software, 2026, 37(3): 971–1005 (in Chinese). http://www.jos.org.cn/1000-9825/7516.htm

Survey on Key Techniques of Approximate Nearest Neighbor Search in Vector Databases

SONG Zi-Wen, WANG Bin, ZHANG Xi-Rui, ZHAO Shi-Hao, YANG Xiao-Chun

(School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China)

Abstract: High-dimensional approximate nearest neighbor search (ANNS) is one of the fundamental and core components of vector databases. With the advancement of artificial intelligence, vector databases have played an increasingly critical role and have gained widespread attention. ANNS methods are essential for optimizing the performance of vector databases. Over decades of development, ANNS has achieved a series of milestones. Rapid advancements in this field in recent years have led to a surge of novel methods and findings, necessitating systematic organization. In this study, the basic concepts of ANNS are first introduced. Next, building upon existing survey frameworks, current approaches are further categorized into five groups based on vector data organization methods: graph-based, hierarchical, quantization-based, hashing-based, and hybrid data organization. Representative works and the latest research advances in the field are systematically discussed. Then, from the perspective of vector search optimization methods, recent advancements are reviewed and categorized into eight types. These categories include hardware acceleration oriented, learning enhanced, distance comparison operation oriented, disk-memory hybrid oriented, data access optimization oriented, distributed oriented, hybrid query oriented, and theoretical analysis. Finally, based on current research achievements and trends, potential future research directions are outlined.

* 基金项目: 国家重点研发计划 (2024YFF0617702); 国家自然科学基金 (U22A2025, 62232007, U23A20309); 高等学校学科创新引智计划 (B16009)
本文由“向量数据库及 DB4LLM 技术”专题特约编辑高宏教授、李国良教授、张蓉教授推荐.
收稿时间: 2025-05-06; 修改时间: 2025-06-30; 采用时间: 2025-08-20; jos 在线出版时间: 2025-09-02
CNKI 网络首发时间: 2026-01-15

Key words: approximate nearest neighbor search (ANNS); vector database; index; high-dimensional data; vector search; query optimization

向量数据库[1–4]是专门用于高效存储和检索高维向量数据的数据库系统. 随着机器学习、计算机视觉和自然语言处理等领域的快速发展, 向量化表征已经成为图像、文本、用户行为等多模态数据的通用抽象表示方法. 随着大语言模型的兴起, 向量数据库的应用场景日益扩展. 面对大语言模型[5]存在的幻觉问题和时效性局限[6], 检索增强生成 (retrieval-augmented generation, RAG) 技术[7]通过向量搜索获取相关上下文作为模型输入, 有效提升了模型性能, 在优化大模型推理性能的 KV cache 中也发挥着重要作用[8], 这使得向量数据库成为 AI 基础设施的关键组成部分. 向量数据库的核心任务是高效地进行向量相似性搜索, 即在给定查询向量的情况下, 快速找到与之相似的向量. 由于维度灾难的问题[9], 在高维向量上搜索难以返回准确的近邻点. 近似最近邻搜索 (approximate nearest neighbor search, ANNS) 旨在放宽对精确近邻的要求, 以提高搜索速度和效率.

在人工智能和大数据驱动的技术场景下, 向量检索面临着更大的挑战, 当前向量数据维度进一步攀高, 能达到成百上千维, 向量数据规模也进一步增长, 攀升至 10 亿级规模, 对于维度为 d 的 N 条向量采用暴力搜索的时间复杂度为 O(Nd), 而且存储原始浮点向量所需的空间随着数据规模和维度的增加线性增加, 内存开销也随之增加. 计算开销、存储成本开销和内存访问开销制约着向量搜索效率的提升. 向量数据库设计高效的索引结构和搜索算法, 以支持快速的相似性搜索和高效的数据存储, 其主要从多个目标进行优化, 包括但不限于: (1) 降低数据访问代价, 通过优化数据的访问方式来降低访问代价, 提升搜索性能; (2) 降低计算代价, 通过减少距离计算等来提升搜索性能; (3) 降低内存开销, 通过量化编码等方式减少内存中的数据量; (4) 提升算法扩展性, 充分利用新硬件和分布式提升处理能力. 在设计算法时需综合考虑查询效率、精度、存储开销和硬件适配之间的权衡, 并与向量数据库有机配合, 以实现高效的向量检索.

向量近似最近邻搜索是一个经典的问题, 已有几十年的历史, 学术界进行了广泛而深入的研究. 过去已有不少优秀的综述对近似最近邻搜索方法进行了系统的总结和分析, 主要集中在基于图、树、哈希、量化等方法上[10]. 近年来, 大批新方法相继被提出, 包括设计新的图构建方法、量化方法等以及从如何利用硬件加速、面向分布式环境的设计、如何优化距离比较操作等角度来进行优化, 已有的综述工作并没有对这些新工作进行充分反映, 亟须对这些研究成果进行梳理. 文献 [10] 从基于树、哈希和图的方法上系统性地评估了各个算法的效果. 其成文时间较早, 当前有更多新的研究成果被提出来并展示了良好的效果. 文献 [11,12] 对基于图的近似最近邻搜索方法进行了综述, 没有涉及近年来提出的其他优化方法. 文献 [13,14] 对基于哈希的近似最近邻搜索方法进行了系统的总结, 主要介绍了多种哈希函数和索引结构. 其成文时间较早, 后续有更多新的工作被提出. 文献 [15] 则对基于乘积量化的近似最近邻搜索方法进行了深入的分析, 探讨了量化技术在高维向量搜索中的应用, 其成文时间同样较早, 当前有更多的量化方法被提出, 尤其将量化和其他形式索引结合的方式取得了很好的效果, 需要进一步总结. 文献 [4] 从向量数据库系统的角度进行了综述, 其总结的算法主要是针对向量数据库系统实现的相关方法. 文献 [16] 核心展示了对 ANNS 算法进行性能评测的工具. 这些综述为研究人员提供了宝贵的参考和指导, 更好地理解和应用近似最近邻搜索方法.

近年来涌现出一批新的研究成果. 一方面, 在向量数据索引组织方面, 一些新的优化策略被提出, 如更多基于图的优化方案, 同时, 研究不再局限于单一的组织方式, 而是研究多种组织方式的协作, 如结合图和量化的方法对索引结构进行优化以提升搜索性能, 这要求我们针对索引组织方法进行更系统的总结分类. 另一方面, 很多方法不再局限于索引组织方式的优化, 而是扩展到更多技术路径, 面向更多场景, 采用更多技术手段来对向量搜索进行优化, 如利用硬件加速进行优化, 利用学习方法来增强索引能力、面向分布式场景等. 这些方法不同于以往的分类框架, 需要新的总结分类, 进行综述梳理.

本文的主要目的是对最新的近似最近邻搜索方法研究成果进行全面的综述. 以往的综述将近似最近邻搜索分为基于图、基于树、基于哈希和基于量化的方法, 本文在现有基础上将向量数据组织方法分为 5 类: 基于图的索引组织方法、基于层次的索引组织方法、基于哈希的索引组织方法、基于量化的索引组织方法和混合索引组织方法. 我们结合相关成果对每种方法进行介绍和分析. 在此基础上, 我们进一步总结最新的向量搜索优化方法成

果, 将其分为面向硬件加速的方法、面向学习增强的方法、面向距离比较操作的方法、面向磁盘内存混合的方法、面向数据访问优化的方法、面向分布式的方法、面向混合搜索的优化方法以及理论分析研究.

本文介绍了向量近似最近邻的背景和基本概念, 包括近似最近邻搜索的定义、数据类型和相似度函数; 根据向量数据索引组织方式并结合最新的工作来介绍不同的向量搜索方案; 进一步总结最新的向量搜索优化方法成果; 基于当前近似最近邻搜索的研究, 展望未来的发展方向.

1 向量近似最近邻搜索的基本概念

本节主要介绍向量近似最近邻搜索的基本概念, 主要从 3 个方面进行: 问题定义、数据类型和相似度度量方法.

1.1 问题定义

向量 K 近邻搜索是在给定向量集合中, 根据特定的相似度度量标准, 获取与查询向量最相似的 k 个向量的过程, 其具体的问题定义如下.

定义 1 (K 近邻搜索). 给定一个数据集 P, 其中的每个点 $\mathbf{p}\in\mathbb{R}^d$ 为 d 维实数向量, 查询点 $\mathbf{q}\in\mathbb{R}^d$ 为同维向量, 定义距离函数 $\delta(\mathbf{p},\mathbf{q})$ 衡量两个点之间的距离, 从数据集 P 中找出一个子集 $K\subset P$, 满足 $|K|=k$, 并且 $\forall\mathbf{p}'\in P\backslash K$, 有 $\delta(\mathbf{p}',\mathbf{q})\geqslant\max_{\mathbf{p}\in K}\delta(\mathbf{p},\mathbf{q})$.

由于维度灾难问题[9], 寻找 K 近邻十分困难, 采用近似最近邻搜索的方式以放宽对结果的要求, 从而提高搜索效率.

定义 2 ((ε,k)-近似最近邻搜索). 给定 $\varepsilon>0$, 给定数据集 P 和查询点 $\mathbf{q}$, 让 $\mathbf{r}_i^*$ 是数据集 P 中距离查询 $\mathbf{q}$ 第 i 近的点, 从数据集 P 中返回 k 个点 $\mathbf{r}_i\in P$, 对每个点 $\mathbf{r}_i$, 满足 $\delta(\mathbf{r}_i,\mathbf{q})\leqslant(1+\varepsilon)\delta(\mathbf{r}_i^*,\mathbf{q})$.

在实践中向量数据库进行搜索时, 并不要求满足 ε 的要求, 而是利用召回率来衡量检索结果的准确度[17], 其定义如下:

$$Recall(R_o)=\frac{|R_o\cap R_k|}{|R_k|},$$

其中, $|\cdot|$ 表示集合中元素的数量, R_o 表示搜索结果, R_k 表示最近的 k 个邻居节点 (ground truth). 同时, 通过 *Recall*-QPS 曲线来衡量不同方法的搜索性能, 不同方法之间比较相同召回率下 QPS (query per second) 的高低, 或者比较同样的 QPS 下召回率的高低.

1.2 数据类型

依据不同的检索模型以及应用场景, 会产生不同种类的向量类型, 主要包括稠密向量、稀疏向量以及二值向量, 向量数据库支持多种数据类型以适应不同的检索需求, 表 1 总结了主要数据类型的特征与应用场景.

表 1 不同的数据类型比较

数据类型	特征	来源	典型应用	优点	缺点
稠密向量	连续浮点数, 高维	768维BERT[18]、1 536维OpenAI[19]	自然语言处理[20]、推荐系统	语义表达能力强, 捕捉细微差别	空间占用大
稀疏向量	高维但大部分为0	词袋模型[21]、SPLADE[22]	文本检索	可解释性好, 存储效率高	语义表达相对较弱
二值向量	每维度0或1	图片哈希[23]	跨模态检索[24]	计算快速, 极省空间 (1 bit/维)	信息损失大, 召回率低
属性数据	关键词、标量等结构化数据[1]	对目标的结构化数据描述等	混合查询、精细化检索	补充向量表达, 提高检索精度	需要额外维护

(1) 稠密向量通过连续的浮点数表示, 能够编码丰富的语义信息, 在深度学习模型中广泛应用, 但高维度和浮点表示导致存储开销较大. (2) 稀疏向量虽然维度可达上万, 但通过仅存储非零值实现高效存储, 它具有良好的可解释性, 常见于词袋模型, 基于神经网络的 SPLADE 模型[22]结合了传统的稀疏检索 (BM25[25]) 以及稠密检索的优点, 在保持可解释性的同时增强了语义表示. (3) 二值向量的每个维度是一个比特位, 取值 0 或 1, 来源于神经网络

的二值化编码等, 在牺牲一定精度的情况下获得了极高的存储和计算效率, 通过异或和 popcnt 指令可以实现硬件级别的加速, 特别适合大规模快速检索场景, 为了弥补信息损失, 通常采用更高的维度来保持足够的区分度. (4) 属性数据作为向量的补充, 提供了额外的结构化信息维度. 在实际检索中, 通过属性过滤与向量相似度的联合查询, 可以实现更精准的检索需求, 弥补单一向量表达在某些特定维度上的不足.

1.3 相似度度量

相似度函数被用于衡量两个向量之间的相似程度. 不同的相似度函数适用于不同的数据类型和应用场景, 下面介绍向量数据库中使用的基本相似度函数. 表 2 是相关内容总览. 更多函数可以参考文献 [26].

表 2 不同的相似度比较

相似度量函数	值域	常用数据类型	复杂度	优点	缺点
欧氏距离	$[0,+\infty)$	稠密向量	O(d)	几何意义直观	计算效率低
余弦相似度	$[-1,+1]$	稠密向量	O(d)	关注向量方向	异常值不敏感
内积	$(-\infty,+\infty)$	稠密、稀疏向量	O(d)	计算速度高	不是度量空间
汉明距离	$[0,d]$	二值向量	O(d/64)	位运算高效	信息损失大
Jaccard相似度	$[0,1]$	稀疏向量	O(m+n)	可解释性强	计算速度慢

(1) 欧氏距离

欧氏距离是最常用的相似度函数之一. 它定义为两个向量之间的平方差之和的平方根. 在实际中, 一般用平方欧氏距离来避免开方运算. 对于两个向量 $\mathbf{p}$ 和 $\mathbf{q}$, 其平方欧氏距离计算公式如下:

$$\delta(\mathbf{p},\mathbf{q})=\sum_{i}^{d}(p_i-q_i)^2.$$

欧氏距离几何意义明确, 具有平移不变性, 广泛用于向量之间的相似性度量.

(2) 余弦相似度

余弦相似度是计算两个归一化后的向量的内积, 代表两个向量之间的角度, 对向量的幅度不敏感. 余弦相似度计算公式如下:

$$sim(\mathbf{p},\mathbf{q})=\frac{\mathbf{pq}}{\|\mathbf{p}\|\|\mathbf{q}\|}.$$

余弦相似度在文本向量中应用广泛. 它与欧氏距离可在归一化后进行转换, 公式如下: $\delta(\mathbf{p},\mathbf{q})=2\cdot(1-sim(\mathbf{p},\mathbf{q}))$.

(3) 内积

内积是向量空间中最基本的相似度函数, 计算公式为 $\delta(\mathbf{p},\mathbf{q})=\mathbf{p}\cdot\mathbf{q}$. 内积的几何意义为两个向量之间的夹角余弦值与向量模长的乘积, 不满足度量空间要求.

(4) 汉明距离

汉明距离是用于计算两个二进制向量之间的相似度函数, 对于两个二值向量 $\mathbf{p}$ 和 $\mathbf{q}$, 汉明距离计算公式为:

$$H(\mathbf{p},\mathbf{q})=\sum_{i=1}^{d}p_i\oplus q_i,$$

其中, $\oplus$ 表示异或操作. 汉明距离定义为两个等长向量中对应位置元素不同的个数. 其几何意义可以直观地理解为两点在超立方体坐标上相异维度的数量. 两个向量之间的汉明距离可以通过异或和 popcnt 实现快速计算.

(5) Jaccard 相似度

Jaccard 相似度是用于计算两个集合之间相似度的函数. 对于两个集合 $\mathbf{p}$ 和 $\mathbf{q}$, Jaccard 相似度计算公式为:

$$J(\mathbf{p},\mathbf{q})=\frac{|\mathbf{p}\cap\mathbf{q}|}{|\mathbf{p}\cup\mathbf{q}|}.$$

Jaccard 相似度的几何意义为两个集合的交集与并集的比值. 两个向量的相似度可以直接通过分别计算集合

的交集与并集来完成. 在实际中, Jaccard 相似度用 minhashing[27]来加快计算速度, 实现对文档的重复度计算等.

2 向量数据索引组织方法

为了实现高效的近似最近邻搜索, 研究工作主要聚焦于通过设计索引对高维向量数据进行组织, 以空间换时间为原则, 将无序的向量数据系统性地、结构化地组织成高效的索引以提升搜索性能. 面对低延迟高精度的近似最近邻搜索的需求以及维度灾难带来的挑战, 需要对高维向量的内在结构和分布规律进行深度的挖掘, 优化索引组织以提升性能.

向量索引组织的经典范式一般将高维向量索引的组织分为树、图、哈希、量化、倒排等方法, 如在文献 [4] 中将索引组织分为 3 种类型: 基于表格 (包含哈希、量化等)、基于树以及基于图的方法, 在文献 [9] 中分别从图、哈希和树这 3 个分类上进行描述. 本文在向量数据索引组织的经典分类范式的基础上结合经典的研究工作和最新的研究成果来展示内在的设计原理和优化思路. 本文在已有研究工作和综述分类的基础上, 进一步地将组织方法分为 5 大类: 基于图的数据组织方法、基于层次的数据组织方法、基于哈希的数据组织方法、基于量化的数据组织方法和混合数据组织方法, 并在这一分类的基础上进一步详细介绍各个方法, 表 3 是对所有组织方法的总体对比展示.

表 3 向量数据索引组织方法总览

数组组织方法	关键方案	代表性方法举例	原始数据存储	维护难度	构建速度	数据访问方式	内存占用	磁盘支持
基于图的数据组织	多层图	HNSW[28]	是	高	慢	随机	高	否
	单层图	NSG[17]	是	高	慢	随机	高	否
基于层次的数据组织	倒排结构	IVF[29]	是	低	快	顺序	中	否
	基于树的结构	k-d tree[30]	是	低	中	顺序	中	否
基于哈希的数据组织	局部敏感哈希	QALSH[31]	是	中	快	混合	低	是
	学习 (二进制) 哈希	GPH[32]	是	中	快	顺序	低	否
基于量化的数据组织	乘积量化	PQ[33]	否	低	快	顺序	低	否
	标量量化	RaBitQ[34]	否	低	快	顺序	低	否
混合数据组织	图+量化	OG-LVQ[35]	否	高	中	随机	低	否
		DiskANN[36]	是	高	中	随机	低	是
	图+树	SPTAG[37]	是	高	慢	随机	高	否
	图+哈希	LSH-APG[38]	是	高	中	随机	高	否
	倒排+量化	IVFPQ[39]	否	低	快	顺序	低	否
	哈希+树	SRS[40]	是	中	快	混合	低	是

表 3 主要从 6 个角度进行总体对比, 并包含其关键方案和代表方法举例, 并结合代表性方法对关键因素进行分析. 当向量数据库中的数据被更改时, 需要维护的索引进行对应的更新, 维护的难易程度是一个需要考虑的关键因素. 当增量索引维护出现性能退化时, 往往需要执行索引重建. 在大规模向量数据场景下, 索引构建速度成为不可忽视的关键因素. 许多向量索引是基于内存的索引, 当针对的是大规模数据构建索引时会占据大量内存资源, 索引的内存占用开销就是一个关键的因素. 一些索引组织方法仅存储量化后的数据而并非原始数据, 一方面, 这种做法可以减少空间消耗, 另一方面也会影响搜索精度, 考虑索引是否存储原始数据对分析其空间占用和搜索性能十分重要. 面向大规模数据和内存资源受限场景, 索引是否支持磁盘是一个十分重要的衡量因素. 磁盘的随机访问性能往往低于顺序访问性能, 同样, 内存的顺序访问因为能够有效地利用缓存预取而能获得相较于随机访问更快的速度, 分析不同方法的数据访问方式有助于评价并优化索引.

2.1 基于图的数据组织方法

基于图的方法在高维向量近似最近邻搜索中取得了优异的性能[28]. 其主要思想是将数据点视为图中的节点,

通过边连接相似的数据点, 从而形成一个图结构. 通过图的遍历和搜索, 可以快速找到与查询点相似的数据点. 图的基本搜索策略是基于 best-first-search[11]的贪心搜索策略, 如算法 1 所描述, 在搜索中通过维护一个队列实现在图上的搜索, 在 HNSW[28]的实现中使用两个队列, 此处一个队列参考了在 NSG[17]图中的实现, 其搜索方式等价. 通过不断地检查搜索队列中的点的邻居节点来更新搜索队列, 实现在图上的遍历操作. 通过控制队列的大小可以控制在图上进行遍历的程度. 搜索终止之后会将队列中的前 k 个结果返回, 作为最终的结果.

算法 1. 贪心搜索算法 (best-first-search).

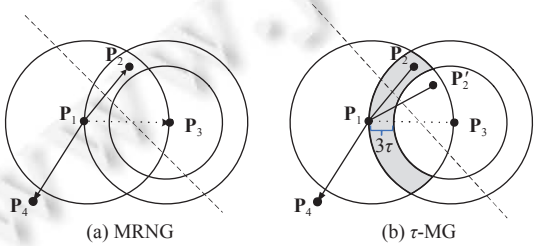
输入: 图 G=(V, E), 查询点 **q**, 返回近邻数量 k, 搜索队列 Q, 搜索队列大小 W, 初始点 ep;
输出: 查询 **q** 的 k 个近似最近邻.

1. 用 ep 初始化 Q, 初始化访问标记 V
2. **while** Q 中有邻居没有被检查的点 **do**
3. **x** = Q 中离 **q** 最近, 同时邻居节点还没有被检查的点;
4. 标记 **x** 的邻居被检查;
5. **for** **x** 的每个邻居 **y** **do**
6. **if** (**y** 在 V 中) **continue**;
7. 计算 **y** 和 **q** 的距离, 更新 Q, 保留最近的 W 个点;
8. 在 V 中标记 **y** 被访问;
9. **end for**
10. **end while**
11. 返回 Q 中距离查询最近的 k 个点作为搜索结果

2.1.1 构建策略

HNSW 算法[28]是基于图的索引组织方法中的一种, 被广泛应用于各大数据库 (以支持近似最近邻搜索). 它通过构建多层次的图结构来加速搜索过程. HNSW 利用导航小世界图的思想, 在 NSW 基础上将其扩展为具有多层导航结构的图, 每一层都是一个图结构, 底层图包含所有数据点, 而上层图则是对底层图的摘要表示. HNSW 取得了相较于 NSW[41]以及 KNN 图[42]更快的搜索速度.

HNSW 采用增量式的构建方法, 将点逐个地插入到图中. 每个点在插入时会随机选择一个层数, 从此层开始逐层插入该点直至最底层. 在每一层中通过贪婪搜索算法搜索到相似的 m 个点, 作为当前点的候选邻居节点, 然后根据选点策略选择若干点作为当前点的邻居节点. 具体策略为从距离要插入的点最近的候选点开始检查, 当且仅当候选点与任何已经成为邻居节点的对象的距离比与要插入对象的距离更近时, 才将该候选点作为邻居节点. 如图 1 所示, $\mathbf{P}_2$ 是 $\mathbf{P}_1$ 的邻居节点, 但是 $\mathbf{P}_3$ 距离 $\mathbf{P}_2$ 更近, 因此 $\mathbf{P}_3$ 不能成为 $\mathbf{P}_1$ 的邻居节点. HNSW 的搜索方法是逐层进行搜索, 从最高层开始, 每层通过贪婪搜索定位到下一层的入口点 (搜索获得的最近的邻居), 当搜索最底层时, 将队列大小设置为 ef, 执行贪心搜索策略, 返回找到的最近的 k 个点作为搜索结果, 通过设置不同 ef 参数值可以平衡效率和精度.



(a) MRNG (b) τ-MG

图 1 MRNG[17]和 τ-MG[43]邻居节点构建策略

HNSW 的设计理念为后续图方法研究和改进提供了重要启示. FANNG[44]是和 HNSW 同期提出来的图构建算法, 除了与 HNSW 相同的邻居构建策略外, 还进一步引入了阈值 τ 来选择更多的候选邻居节点. NSG 图是一个单层图结构. 它形式化地描述了单调搜索网络[17], 并依此提出了 MRNG (monotonic relative neighborhood graph), 在构建图的过程中, 其邻居节点选择策略与 HNSW 一致. SSG[45]在 NSG 的基础上进一步改进, 使每个节点的邻居节点在不同方向上尽可能均匀地分布. HSSG[46]进一步提出了多层结构以加快搜索速度. DPG[10]在 KNN 图的基础上, 在节点选择过程中引入邻居节点之间角度的信息以使节点更分散, 从而提升搜索性能. 针对 NSG 等图的构建策略会导致搜索路径太长的问题, Vamana 图[36]提出在构建阶段引入一个参数 α 来调节裁边选点力度的 α-RNG 规则, 将所有的候选邻居节点按照与将要建立邻居的点 $\mathbf{p}$ 的距离升序排列, 对于当前准备要与 $\mathbf{p}$ 建立邻居的 $\mathbf{p}^*$, 如果满足 $\alpha\|\mathbf{p}^*-\mathbf{p}'\|\leqslant\|\mathbf{p}^*-\mathbf{p}\|$, 则不作为邻居, 其中 $\mathbf{p}'$ 为已经建立的邻居节点. Vamana 采用两阶段迭代的图构建策略: 首先生成随机图结构, 随后对每个节点执行贪婪搜索以确定潜在邻居集合, 并通过鲁棒剪枝 (RobustPrune) 筛选出满足距离约束 α 的邻居. 具体而言, 剪枝过程会保留对搜索方向贡献最大的邻居, 同时剔除冗余连接, 从而降低图直径并减少搜索时的跳数. τ-MG 指出[43], MRNG 的方法都基于查询点 $\mathbf{q}$ 是数据集中的一点这一假设来设计, 但实际中往往并非如此, 对于查询点不存在于数据集中的情况, 在 MRNG 图中搜索会出现无法发现最近邻点的情况, 基于此观察提出了 τ-MG 的选点策略, 使查询 $\mathbf{q}$ 和近邻点 $\mathbf{p}$ 在满足 $\delta(\mathbf{q},\mathbf{p})<\tau$ 的情况下依然能够搜索最近邻. 图 1 基于文献 [43] 的描述进一步调整优化, 展示了这两个方法的选点策略, 可以看到, τ-MG 降低了某个点成为邻居节点的要求. HNSW 通过将数据点不断插入图中最终完成索引构建的增量方式可以支持动态数据的插入, 避免面对增量数据时进行索引的全局重建. NSG 等方法主要针对的是静态数据集, 在完整的数据集上完成最终的图构建. 它们依赖于先构建一个基础的近邻图, 然后在此图上针对每个节点运行贪心搜索策略来确定候选集, 并利用相应的选边策略来确定最终的邻居节点, 如 NSG 依赖于先构建一个近似 K 近邻图 (KNN graph), 而 τ-MG 依赖于先构建一个 NSG 或者 HNSW 图.

在 DEEP、SIFT、MSong、GIST、CRAWL、GloVe 这 6 个经典数据集的对比上, 根据文献 [43], 在相同召回率下, τ-MG 可以取得比 NSG 更好的搜索性能, 而 NSG 取得了优于 HNSW 的搜索性能. 表 4 展示了这 6 个数据集的统计信息.

表 4　6 个主要数据集的统计信息

数据集	维度	数据量	查询数量	类型
DEEP	256	1 000 000	200	图片
SIFT	128	1 000 000	10 000	图片
MSong	420	990 000	200	音频
GIST	960	1 000 000	1 000	图片
CRAWL	300	1 980 000	10 000	文本
GloVe	100	1 180 000	10 000	文本

此外, HCNNG[47]基于分治的思想来构建图. 它将数据集递归地依据每次随机选择两个数据点进行二分类, 直到每个子集中的数据点个数小于预定义的数量 N 时则停止划分, 然后分别针对每个子集分别构建最小生成树, 最后将每个子集进行合并. 这个递归划分并合并的过程会进行多次, 最后将多次形成的图进行合并以形成最终结果. HCNNG 不依赖贪心搜索策略来获取候选集, 而是依赖于将数据集划分为多个子集, 同时其连边策略依赖于在这些子集中构建最小生成树. 实验展示其在 GIST、SIFT 等经典数据集上取得了优于 HNSW 的搜索性能. RoarGraph[48]研究跨模态的近似最近邻查询. 由于不同模态的数据通过嵌入生成的向量属于不同分布, 使得查询向量相较于数据集中的向量出现分布外 (out-of-distribution) 的情况. 针对这类问题, RoarGraph 提出了一种查询导向的图索引方法, 通过二分图来建模维护跨模态数据间的距离度量.

2.1.2　更新策略

传统基于图的近似最近邻索引 (如 HNSW、NSG 等) 虽然在静态数据集上表现出色, 但是其静态特性无法应

对实时数据更新场景, 导致工业界依赖定期全量重建索引[49]的高成本方案. 现有动态索引方案存在内存消耗大、查询延迟高等缺陷, 而量化方法[15]虽然降低了内存需求却牺牲了召回精度. 针对这一挑战, FreshDiskANN[49]通过设计更新图的算法与分层存储架构, 在单机上实现 10 亿级数据集的毫秒级实时更新与搜索, 同时保持 95% 以上的召回率.

FreshDiskANN[49]是面向流式相似性搜索的实时图索引系统, 支持动态更新. FreshDiskANN 的核心创新包含 3 部分. 首先, 提出了首个支持增删操作的图索引算法 FreshVamana, 通过 α-RNG 邻域剪枝规则[36]维持图结构的搜索性能. 该算法采用两阶段更新策略: 插入时基于贪婪搜索定位新节点的拓扑位置, 采用鲁棒剪枝保留满足 α-RNG 规则的最优邻接边; 删除时延迟处理失效节点, 通过合并邻域路径避免图稀疏化, 配合周期性批量剪枝消除冗余边. 其次, 设计了分层存储架构, 将长期数据存储在 SSD 构建的长期索引 (long-term index, LTI) 中, 而内存中的临时索引 (TempIndex) 负责聚合实时更新. LTI 采用乘积量化压缩数据以降低空间占用, 而 TempIndex 保留原始精度向量确保更新质量. 最后, 提出了流式合并算法 StreamingMerge, 通过删除阶段块扫描、插入阶段双路搜索、修补阶段批量剪枝的 3 步策略, 将增量更新以线性时间复杂度合并至 LTI. 该算法仅需两次全量扫描即可完成索引更新, 相比于全量重建节省 90% 计算资源, 同时通过距离近似计算减少 SSD 随机访问. 实验结果表明, FreshDiskANN 在 10 亿级规模数据集上可实现 1 800 次/s 的稳态更新吞吐, 搜索延迟稳定在 20 ms 内且召回率超过 95%. 相比于现有方案, 其硬件成本降低到 1/5–1/10, 为流式相似性搜索提供了首个可扩展的工业级解决方案.

2.2 基于层次的数据组织方法

2.2.1 基于树的方法

基于树的索引方法是经典的索引组织方法, 其被广泛用于索引的设计, 但是由于维度灾难的问题, 在对高维向量进行索引时难以取得较好的效果. 基于树的方法的核心思路是将向量数据集合递归地划分为若干个子集, 从而形成层次化的树状组织结构.

k-d tree[30]是经典的多维数据索引结构, 它选择超平面作为划分数据的依据. k-d tree 是一棵二叉树, 每个节点对应一个 k 维向量数据点. 每个非叶节点同时对应一个超平面, 该超平面将空间划分成两个半空间, 分别交由其左右子节点处理. k-d tree 在非叶节点中考察方差最大的维度, 对于某一个非叶节点, 选择当前空间内在考察维度上取值最接近平均值或者中位数的向量数据点所在的与考察维度的方向向量垂直的超平面以进一步划分空间. k-d tree 的搜索过程是一个递归过程, 从根节点开始, 根据查询点 **q** 在当前节点考察的维度上的取值与当前节点对应的向量数据点的取值的大小关系判断下一步向左子树或右子树前进, 直到到达叶节点, 将其标记为当前的最近点并回溯, 对于每个回溯到的节点, 首先检查其本身与查询点 **q** 的距离是否较当前最近点更近, 再计算查询点 **q** 与当前节点的分隔超平面的距离, 如果该距离小于当前的最近距离, 则说明在超平面另一侧的空间内可能存在距离查询点 **q** 更近的节点, 需要递归搜索另一棵子树.

除了 k-d tree 之外, 还有很多经典的基于树的方法. Randomized k-d tree[50]与传统的 k-d tree 相比, 在划分维度的选取上有所不同. 传统的 k-d tree 在每个非叶节点中选择当前空间内数据方差最大的方向, 而 randomized k-d tree 只从所有数据方差最大的 D 个方向中随机选取其一. R-tree[51]使用最小边界矩形组织数据, PCA tree[52]与 PKD tree[50]根据主成分分析确定划分超平面, Random Projection tree[53,54]不使用复杂度较高的主成分分析方法, 通过随机投影的方法确定划分超平面. 而 M-tree[55] (通过中心点与覆盖半径构建嵌套超球体结构) 和 VP-tree[56] (对每个节点选择单一观测点, 根据数据与参考点的距离将数据划分为两个区域) 等方法则是基于距离度量而非坐标来组织数据. K-means tree[57]和 Hierarchical K-means tree[58], 基于 K-means 聚类算法将数据划分为 K 个簇, 并在簇内递归划分直到满足终止条件 (如树的深度、簇内数据数量等). ANNOY[59]是一个基于树的方案, 其不同版本中维护了 Random Projection trees 森林与 Hierarchical K-means trees 森林. FLANN[60]是一个包含一系列近似最近邻搜索方法的库. 它可以根据不同数据集考虑索引构建、搜索耗时以及内存占用的消耗, 从不同的包括 Hierarchical K-means tree 与 Randomized k-d trees 森林以及线性扫描等算法中选择效果最好的方法. LRUS-CoverTree[61]设计了一种树状索引, 并根据其性质设计了分支限界算法以支持近似的最大内积查询.

2.2.2 基于倒排的方法

(1) 构建策略

倒排文件索引 (inverted file (IVF) index) 通过聚类缩减了搜索范围从而提升搜索速度, 图 2 展示了一个基本的 IVF 索引结构. 高维空间被划分成 Voronoi 图[62], 向量数据点根据聚类算法被划分为若干簇 (Voronoi 单元), 每个簇被一个质心代表, 索引会记录质心及其代表的簇中包含的向量数据点的信息. 对于一个查询点 **q**, 首先计算得到所有质心中与 **q** 距离最近的质心, 再在该质心对应的簇中通过线性扫描簇中的向量数据点来得到该簇中离 **q** 最近的 (若干) 数据点作为搜索结果. 受边际效应的影响, 当查询点距离簇分界线较近时, 搜索可能无法得到正确的结果. 可以通过搜索多个簇来提升搜索结果的召回率, 但由于遍历了更多数据, 搜索的速度会减慢. 算法库 Faiss (Facebook AI similarity search)[29]与向量数据库 Milvus[63]实现了 IVF_FLAT (FLAT 代表簇中记录的是未被压缩或量化的原始数据) 索引.

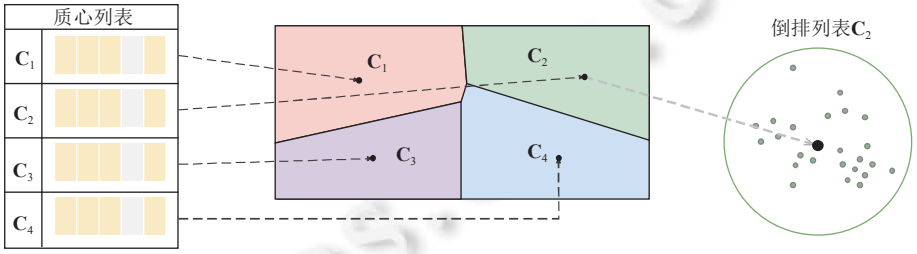


图 2 IVF 索引结构图

一些工作进一步优化了倒排方法. Tribase[64]细粒度地划分了聚类, 根据子聚类的不同特征选择不同的搜索方法, 并结合了距离与角度的三角不等式以提升剪枝效果. SC[65]用子空间碰撞度量 SC-score 来近似欧氏距离, 给出了误差的理论分析, 并进一步为搜索框架 SC 设计了一种基于聚类的轻量化索引与搜索策略 Suco.

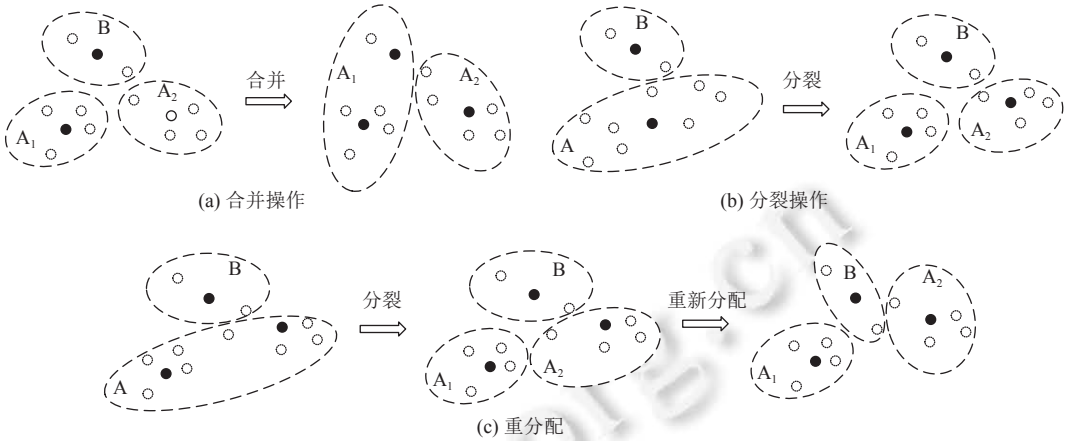
稀疏向量由于其良好的可解释性, 在信息检索中发挥越来越重要的作用, 基于倒排方法来索引稀疏向量取得了很好的效果. 文献 [66] 提出基于倒排的索引方法 SEISMIC, 取得了比基于图的方法更高的搜索效率. 文中指出, 稀疏向量的 L1 范数主要由少部分的高权重坐标贡献, 两个向量的内积也可以由少部分的坐标有效地近似. 基于这一发现, SEISMIC 在构建倒排索引时对每个倒排列表中的数据按照权重排序, 进行静态剪枝, 减少检索时需要评估的数据数量, 并通过 K-means 聚类将倒排列表进行分块, 构建摘要向量, 在搜索时以整体跳过整个子块来提高搜索速度. 实验结果显示其拥有较低构建时间, 占用较小存储空间, 在保持较高召回率的同时具有更快的检索速度.

(2) 更新策略

SPFresh[67]是一个支持 10 亿级向量搜索和更新的系统, 其核心是轻量级增量再平衡协议 LIRE. 该研究针对高维向量索引更新存在的全局重建成本高、查询性能波动剧烈等问题, 创新地将传统索引的全局重建过程解耦为局部动态调整机制. SPFresh 基于平衡聚类索引框架 SPANN 构建基本索引结构, 通过维护向量分区的最近邻分配属性 (neighbor posting assignment, NPA), 在数据分布发生偏移时仅需对边界区域的少量向量进行再分配. LIRE 协议包含分裂、合并、重新分配这 3 类核心操作, 图 3 是这 3 类操作的示意图, 当分区向量数超过阈值时, 通过多约束平衡聚类算法将其拆分为两个新分区, 并基于必要条件检查触发相邻分区的向量再分配; 对于过小分区则采用最近邻合并策略, 删除冗余质心后重新校准向量归属. 与传统全局重建方法相比, LIRE 通过限定影响范围 (仅检查分裂质心最近的 64 个邻居分区) 和版本控制机制 (维护向量复制版本号), 将再平衡操作的计算复杂度从 $O(N)$ 降至 $O(1)$, 有效地解决了数据持续更新导致的分区失衡和查询准确率衰减问题.

SPFresh 在更新方面进行了系统架构的设计: 更新器采用尾部追加策略将新向量写入最近邻分区, 并维护版本映射表记录逻辑删除; 本地重建器通过多线程并发执行分区分裂、合并及再分配操作, 采用细粒度分区锁保证原子性. 在存储层设计了用户态块控制器, 绕过传统文件系统直接管理 NVMe SSD 块, 通过预分配空闲块池和批量异步 I/O 请求队列实现高吞吐量访问. 为了实现崩溃恢复机制, 系统结合周期性快照与预写日志机制, 利用快照及

写时复制技术实现秒级数据回滚. 实验结果表明, 在处理 10 亿级 SIFT 数据集时, SPFresh 在 15 核单 NVMe SSD 环境下同时支撑 4k QPS 搜索吞吐和 2k QPS 更新吞吐. 相较于仅支持追加更新的 SPANN+, 其查询准确率提升超过 15 个百分点且尾部延迟稳定在 4 ms 左右, 验证了动态再平衡机制对索引质量的持续维护能力.



(a) 合并操作　(b) 分裂操作　(c) 重分配

图 3　SPFresh 的基本更新操作[67]

2.3　基于哈希的数据组织方法

基于哈希的方法是一种高效的索引组织方法. 它将数据点映射到一个低维空间中或者由一系列比特组成的二进制编码中成为哈希码, 从而可以通过哈希值来快速定位数据点. 基于哈希的索引组织方法主要分为两类: 局部敏感哈希 (locality sensitive hashing, LSH) 和学习型哈希 (learning to hash). LSH 通过随机投影等方式将相似的数据点映射到相同的桶中来实现近似最近邻搜索. LSH 一般是与数据无关的. 学习型哈希则是将数据点映射到一个二进制编码中, 从而可以通过汉明距离来计算相似度. 它考虑了数据分布的特点. 本文依据其哈希后的结果是二进制形式而称学习型哈希为二进制哈希. 关于哈希的相关工作在综述文献 [13,14,68] 中有详细的介绍. 在此, 我们从局部敏感哈希和学习型哈希两个方面进行简要介绍, 主要介绍在利用哈希函数对数据进行哈希之后如何进行组织检索, 更多更详细的介绍可以参考相关的综述.

2.3.1　局部敏感哈希

局部敏感哈希在文献 [9] 中被首次提出, 此后一系列以此为基础的研究工作取得了更好的搜索性能. 在 LSH 中, (r_1, r_2, p_1, p_2)-敏感哈希函数是其中的关键. 要找到一个哈希函数族满足如下的性质.

定义 3 (局部敏感哈希函数)[69]. 一个哈希函数族 $\mathcal{H}$ 被称为 (r_1, r_2, p_1, p_2)-敏感 (其中, $r_1<r_2$, $p_1>p_2$), 如果对于任意两个数据点 $\mathbf{x}$ 和 $\mathbf{y}$, 满足以下条件.

(1) 如果 $d(\mathbf{x},\mathbf{y}) \leqslant r_1$ (即两点距离不超过 r_1), 则它们被哈希到同一桶的概率至少为 p_1:

$$\Pr_{h\in\mathcal{H}}[h(\mathbf{x}) = h(\mathbf{y})] \geqslant p_1.$$

(2) 如果 $d(\mathbf{x},\mathbf{y}) \geqslant r_2$ (即两点距离至少为 r_2), 则它们被哈希到同一桶的概率最多为 p_2:

$$\Pr_{h\in\mathcal{H}}[h(\mathbf{x}) = h(\mathbf{y})] \leqslant p_2.$$

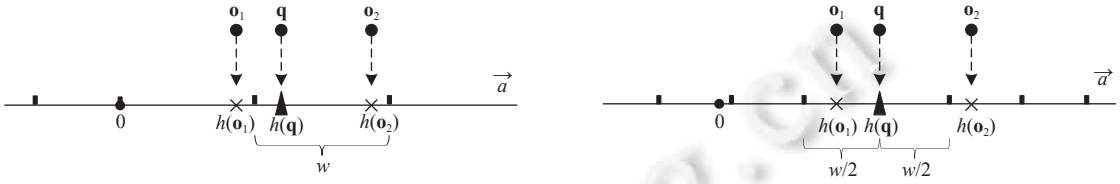
一个经典的方法是基于 p-stable 分布来设计 LSH 函数族. 其定义如下.

定义 4 (p-stable 分布)[69]. 一个分布 D 被称为 p-stable (其中 $p \geqslant 0$), 如果对于任意 n 个实数 $v_1, v_2, \ldots, v_n$ 以及独立同分布 (independent and identically distributed, i.i.d.) 的随机变量 $X_1, X_2, \ldots, X_n \sim D$, 随机变量 $\sum_{i=1}^{n} v_i X_i$ 的分布等同于 $\left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p} \cdot X$, 其中 $X \sim D$.

高斯分布是一个 p-stable 分布. 文献 [69] 中提出利用高斯分布设计随机投影向量, 将高维数据映射到低维空

间中, 实现欧氏空间的 LSH 函数的设计. 后续有一系列工作提出更多的优化方案, 如经典的 QALSH 方案[31].

QALSH 的目标是解决哈希桶划分的问题[31]. 在此前的工作如 C2LSH[70]中, 需要预先划分哈希桶, 这种方式可能导致距离查询较近的数据点被划分到不同的桶中, 影响搜索效率. QALSH 提出以查询对象为锚点, 根据到查询点的距离信息来划分哈希桶. 图 4 展示了两种划分策略的区别, QALSH 可以更加高效而灵活地划分数据. QALSH 进一步提出了虚拟重哈希 (virtual rehashing) 技术, 以动态地设置哈希桶的范围, 避免重建哈希表. 具体来说, QALSH 是将数据点通过多个哈希函数映射为多个一维数据点并存储于若干 B+树中, 然后在 B+树的叶子节点中进行线性搜索, 以扫描更多的哈希桶, 从而找到近似最近邻点, 当检查的点的数量超过预设值时停止搜索.



(a) C2LSH (b) QALSH

图 4 C2LSH[69]和 QALSH[70]投影示例

此外, SOSIA[71]研究利用哈希来加速稀疏向量的近似最大内积搜索问题. SOSIA 首先将稀疏向量数据转化为二进制向量并视为集合, 随后通过多个不同的局部敏感哈希函数将其映射到哈希桶中. 对于查询向量, 首先将其转化为二进制向量, 再根据 Jaccard 距离进行搜索. FARGO[72]通过设计一种全局多次探测 (global multi-probing, GMP) 策略, 利用内积的性质筛选高质量的候选项, 通过将点随机映射到两个不同方向的 RXT 变换来减少数据不均衡分布, 结合自适应的终止条件以进一步提高搜索效率. 文献 [68] 对更多局部敏感哈希的相关工作进行了综述.

2.3.2 学习型哈希

学习型哈希通过学习一个映射函数, 将数据点编码为二进制形式, 从而能够在汉明空间中进行高效近邻搜索, 在保证检索性能的同时尽可能接近真实查询结果. 学习型哈希的关键在于如何设计映射函数, 使得相似的数据点在映射后具有相似的哈希值, 更多的哈希函数族可以参考文献 [13,14] 中的介绍. 在学习哈希函数之后, 会对数据进行哈希编码, 编码完成之后需要在学习到的哈希编码上进行搜索操作, 主要分为两类: 基于线性扫描的搜索方法 (Hamming ranking) 和基于哈希码索引的搜索方法[13,14]. 前者通过计算查询点的哈希码和所有数据点的哈希码之间的汉明距离, 然后保留距离最小的前 L 个数据点作为近似最近邻点. 后者则是通过索引结构来加速搜索过程, 由于是二进制编码, 因此可以直接进行哈希表查找, 这也是一种倒排表. 但是实际中编码往往很长, 导致哈希表的存储开销很大, 因此需要进一步设计基于倒排的方式对哈希码进行索引, 搜索过程是将查询点的哈希码按照倒排建立的情况进行分割, 然后在每个分割的哈希码上进行倒排表的查找, 最后将所有的结果进行合并来得到最终的近似最近邻点.

经典的方案是文献 [73] 提出的基于鸽巢原理的方案, 将两个二进制编码 $\mathbf{h}$ 和 $\mathbf{g}$ 分成 m 个子串, 如果两个点之间的汉明距离 $\|\mathbf{h}-\mathbf{g}\|_H < r$, 那么必然有一个子串的汉明距离小于等于 $\lfloor r/m \rfloor$, 基于此, 只需要在所有的子串上执行半径小于等于 $\lfloor r/m \rfloor$ 的搜索操作即可. 该方案将二进制编码分为 m 个不相交的子串, 针对每个子串建立一个哈希表, 查询时, 同样将查询点的哈希码分为 m 个子串, 然后在每个子串上进行哈希表的查找, 最后将所有结果进行合并, 计算精确的汉明距离, 最终返回汉明距离满足要求的数据. 在采用前述方式之前需要检查的桶的数量为:

$$L(b,r)=\sum_{k=0}^{r} \mathrm{C}_b^k,$$

优化之后变为 $m\cdot L(b/m,\lfloor r/m \rfloor)$. 这种方式减少了空桶的数量, 也就是减少了需要维护的哈希桶的数量, 从而降低了内存需求.

GPH 方法[32]进一步优化了利用鸽巢原理在汉明空间进行检索的性能. 针对当前方法基于等长的空间划分和固定阈值分配导致生成不必要候选集的问题, GPH 提出了一种新形式的鸽巢原理. 它允许变长的子空间划分和不

同的阈值分配策略, 基于新的原理, 能够减少候选点的数量从而提高性能. 文献 [32] 设计了基于成本感知的空间划分和阈值分配策略来优化搜索性能. 文献 [74] 进一步讨论了支持汉明距离的连接查询处理方法. 文献 [75] 提出了允许重合的划分策略增强的鸽巢原理 (augmented pigeonhole principle, APP), 能够捕捉查询负载和数据集的关系来优化查询, 并基于 APP 提出了一个高效的汉明空间索引框架以支持范围查询和 KNN 查询.

2.4 基于量化的数据组织方法

基于量化的方法主要目的是降低数据占用空间, 获得处理大规模数据的能力. 它将高精度的原始数据用低精度的方式进行表示, 然后用整数进行编码, 在检索时根据存储的整数值查询码表将向量数据还原为低精度的表示. 下面介绍具体技术.

2.4.1 乘积量化

乘积量化 (product quantization)[15,33]是将高维空间中的点用一个有限子集进行编码的过程. 其核心思想是将高维向量分解为多个低维子空间, 并分别量化, 减少了码本的占用, 显著减少计算和存储开销. 乘积量化大概分为子空间分解、独立量化、压缩表示和距离计算这 4 个方面. 图 5 展示了乘积量化的基本过程.

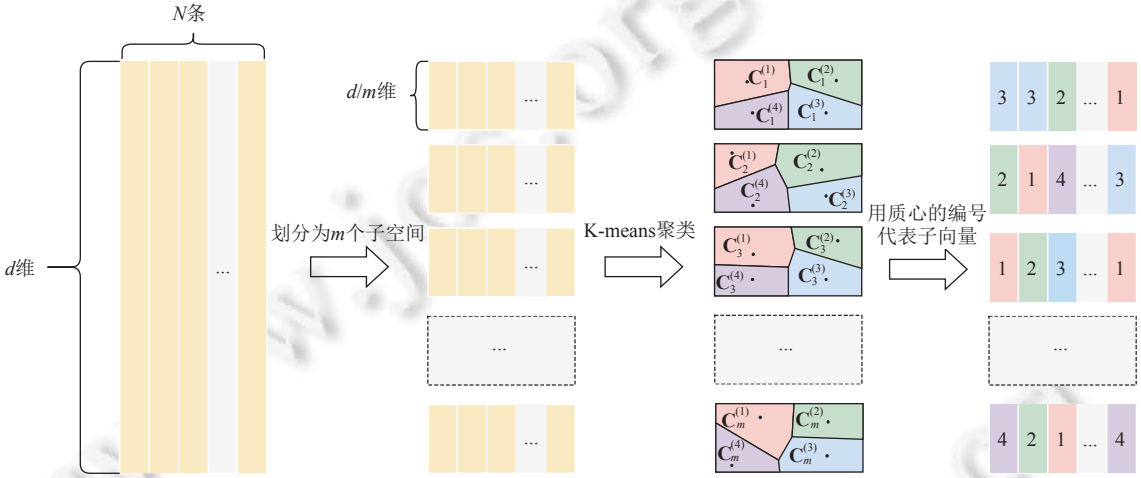


图 5 乘积量化示意图

子空间分解: 将原始高维向量 (维度 d) 均匀切分为 m 个子空间, 每个子空间维度为 d/m. 例如, 128 维向量分成 8 个 16 维子空间. 独立量化: 对每个子空间单独进行 K-means 聚类, 生成 K 个质心 (如 $K=256$). 每个子向量用最近的质心编号 (1–K) 表示, 相当于用 8 位整数编码. 压缩表示: 原始向量被压缩为 m 个质心编号的组合 (如 8 个子空间编号拼接), 存储空间从 d 个浮点数降至 m 个整数, 极大地节省了内存. 距离计算: 距离计算分为对称距离和非对称距离. 对于前者, 查询向量和数据库中的向量都用质心来近似, 通过预计算的子空间质心距离表查表快速求和, 需要为每个子空间维护一个表格, 空间复杂度为 $O(mK^2)$; 对于后者, 仅数据库中的点用质心表示, 查询向量保留原始值, 因此在查询到来时需要先计算查询和质心的距离形成一个距离表, 额外维护 m 个表格的空间复杂度为 $O(mK)$, 后续搜索通过查表求和快速求解距离, 通常精度更好. 两种方法求与 N 个点的距离的复杂度都为 $O(mN)$. 通过量化能够显著降低存储开销, 如 128 维向量内存占用可降至原来的 1/64, 适用于大规模向量检索的场景.

在基本的乘积量化的基础上, 有更多优化方法被提出来. 文献 [15] 讨论了许多基于乘积量化的工作, 包括但不限于 Optimized PQ[76]与 Cartesian K-means[77]通过正交矩阵旋转空间以获取更好的码本; Additive quantization[78,79]与 Composite quantization[80]用子空间质心的和拓展了乘积量化的表示方法; Optimized CK-means[81]、Tree quantization[82]与 Sparse PQ[83]使用不同的编码策略以得到更准确的查询结果. 此外, DPQ[84]额外量化了数据到其最近聚类质心的残差距离, DCPQ[85]通过基于学习的方法维护码本, RVPQ[86]为每个子空间构建由多个有序残差码本组成的残差层次结构, CHPQ[87]根据数据特征局部优化码本, 文献 [88] 提出 Fast-Scan 策略来充分利用 SIMD (single instruction

multiple data) 指令加速距离计算. 更多关于量化的内容可以参考文献 [15].

2.4.2 标量量化

标量量化是一种广泛应用于向量压缩的技术[89], 旨在通过降低数据精度来减小存储和计算开销, 其核心是将高维向量的每个分量的连续浮点数映射到有限个离散值, 从而用更少的比特来表示数据. 例如, 将 32 位的浮点数量化为 8 位的整数, 可以将存储需求减少为原来的 1/4. 标量量化的优势不仅体现在存储压缩上, 还体现在加速相似性搜索过程, 由于数据量变小, 搜索过程中能够更快地访问内容, 进而提高搜索速度.

标量量化的基本步骤如下. (1) 确定量化范围. 分析数据的分布情况, 如浮点向量的最大值、最小值, 以确定量化区间的上下界. (2) 划分量化区间. 根据目标精度和存储需求, 划分为若干个离散值区间. (3) 映射. 将每个浮点数映射到对应的离散值, 通常采用线性映射或非线性映射. (4) 存储. 保存量化所需要的元数据, 包括上下界、步长、码本等内容, 以便在搜索时能够还原为近似的值. 标量量化的关键在于如何选择离散值和映射方式, 以最小化量化误差和存储开销. LVQ (locally-adaptive vector quantization)[35]是一个局部自适应的量化方法. 它考虑每个向量的特点来有针对性地优化, 并提出两层量化机制来对残差部分进一步量化从而提高精度. 后续将详细介绍 LVQ.

RaBitQ[34]是一种新型的量化方法. 它利用乘积量化和标量量化的优点, 取得了显著的效果. 不同于传统的乘积量化方法, 它提供了误差界限的理论保证. 它将 d 维的向量量化为 d 比特的二进制编码, 并设计无偏估计方法来计算向量间的内积. 为了能够高效地进行计算, RaBitQ 引入了标量量化, 将查询点进行标量量化之后参与计算, 通过这一操作, 在计算中引入了位运算, 从而能够加快计算速度. 文献 [34] 中提供了两种方法, 一种针对的是单条量化数据的计算, 另一种是针对批量量化数据的计算. 前者通过比特级别的操作 bitwise-and 结合 popcount 实现快速计算, 而后者则是通过结合 SIMD 指令集的查表操作来实现快速计算. 文献 [34] 将 RaBitQ 应用在 IVF 索引结构中以优化近似最近邻搜索的速度, 实验结果表明其能够获得比传统量化更快的查询速度. 文献 [90] 针对 RaBitQ 只支持 1 比特量化的问题, 进一步扩展了 RaBitQ 的支持范围, 使其能够选择较小的压缩率来实现增加空间占用, 从而获得更高的估计精度, 同时保留了对距离的无偏估计能力.

2.5 混合数据组织方法

结合各类数据组织方案的优点形成混合组织方案的方法受到越来越广泛的关注, 有很多研究成果展示出通过有机的组合形成新的组织方法, 相较于单一方案能够取得更好的效果. 单一的组织方案各具优势和劣势, 通过组合的方式可以充分发挥各个方案的优势, 同时弥补单一方案的劣势, 进而提高近似最近邻搜索效率. 下面我们介绍相关的技术.

2.5.1 基于图和量化的组织方法

针对图能够实现高性能查询和量化可以降低内存占用的优势, Aguerrebere 等人[35]提出了基于图和量化的 OG-LVQ 方法, 取得了突破性的性能提升, 为处理 10 亿级高维向量相似性搜索提供了高性能解决方案. 该方法从软件算法和硬件架构两方面进行设计.

在软件算法方面,提出了局部自适应量化的方法. (1) 该方法基于对深度学习时代高维向量分布特性的深度洞察, 通过全局均值中心化消除维度间分布偏移后, 采用向量级动态范围标量量化策略, 使每个向量的数值范围独立适配其统计特性, 最大化利用有限比特位的表达能力. 与传统的全局量化或分块量化 (如 PQ) 相比, 该策略在几乎不损失精度的前提下, 将向量存储带宽降低至原始数据的 1/8. (2) 该方法设计了二级残差量化结构, 第 1 级量化结果直接用于图索引的快速搜索, 第 2 级残差编码仅在高精度重排序时激活, 形成“粗筛-精排”的协同机制. (3) 在索引构建层面, LVQ 实现了压缩向量直接构造近邻图, 其理论证明量化误差对图拓扑稳定性的影响呈正态分布, 通过 4–8 比特量化即可保持剪枝规则的等效性, 使索引构建内存需求降低到 1/6 而不影响搜索质量.

在硬件加速层面: (1) 指令集优化, 设计了基于 AVX 指令集的向量解码与相似度计算的融合内核, 将压缩数据流转化为 SIMD 友好的计算模式, 使单指令处理速度较 float16 提升 2.1 倍; (2) 访存能力增强, 针对图搜索的随机访存特征, 提出基于预取偏移量与步长的自适应预取策略, 配合大页内存布局消除 TLB 失效, 实现 90% 的峰值内存带宽利用率; (3) 并行架构适配, 通过扁平化数据结构和无锁队列设计, 在 40 核服务器上相交单线程达成 33 倍

加速比. 实验结果表明, 该方法在保持 99% 搜索精度的同时, 10 亿级数据集查询吞吐量相比于现有最优方案提升 20.7 倍, 内存消耗减少至 1/3. 其创新之处在于, 首次实现压缩编码与图索引的端到端协同优化, 显著超越传统量化方法与图搜索框架的组合方案, 为基于大语言模型的检索增强系统提供了可扩展的底层支持. 此外, DiskANN 同样是一个基于图和量化的索引结构. 它是一个基于磁盘和内存混合的索引方式, 充分利用 SSD 的能力来提高单机处理能力, 在第 3 节会进行更多的介绍.

2.5.2 基于图和树的组织方法

针对基于树的方法在搜索上效率不足以及图方法会陷入局部最优的问题, 文献 [37] 提出了基于树和图结合的搜索策略, 将经典的 K-NN 图[42]和分区树[30]相结合, 采用查询驱动的迭代搜索策略, 不断地扩展搜索范围, 最终获得搜索结果. 其搜索主要分为 4 个步骤: (1) 通过搜索树确定初始的搜索结果; (2) 在上一步基础上对局部的图做进一步搜索; (3) 根据搜索历史在树上搜索, 确定在图上搜索的新位置; (4) 在图上进一步进行搜索. 其核心思想是通过迭代的邻域扩展, 逐步逼近真实最近邻. 该方法结合了图结构与搜索策略的优势, 为基于图和树混合的组织方法提供了重要参考. 实验展示出该方法相对于图方法取得了非常好的性能提升, 为后续基于图的方法的发展提供了重要的依据. ELPIS[91]是一个基于图和树的方法. 它首先通过 Hercules[92]将数据集划分为若干个聚类, 每个叶子节点对应一个聚类; 随后并行地在叶子节点上构建 HNSW 图索引. Hercules 通过 EAPCA[93]摘要进行裁剪来加速树上搜索过程.

2.5.3 基于图和哈希的组织方法

针对图方法构建成本高和局部敏感哈希查询效率低的问题, 文献 [38] 提出了结合 LSH 和图方法的 LSH-APG 混合索引方法. 在构建时对数据集中的每个点, 通过 LSH 索引快速找到邻居候选节点, 然后根据邻居节点的选择规则构建邻居, 如果邻居数量超过预先定义的上限, 则删除最远的点. 通过 LSH 框架加速邻居的搜索, 避免了传统图方法的高计算成本. 在搜索时, 给定查询点 q 后通过 LSH 找到入口点, 从而降低搜索的半径, 并在检查邻居节点时通过访问 LSH 结构过滤距离 q 较远的点, 检索距离计算. LSH-APG 通过 LSH 加速构建和动态剪枝优化, 在保证查询质量的同时, 与 HNSW 等图方法相比, 显著降低了构建时间.

2.5.4 基于倒排和量化的组织方法

IVFADC[39]是一种结合倒排与非对称距离计算 (asymmetric distance computation, ADC) 的方法. 在预处理阶段, IVFADC 将 N 条向量数据 $\mathbf{X}=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N]$ 划分为 J 组 $\mathbf{X}=\cup_{j=1}^{J}\mathbf{X}_j$. 每一组向量 $\mathbf{X}_j$ 记录一个代表向量 $\mathbf{C}_j$. 对每一组向量 $\mathbf{X}_j$, 计算其中的每一条向量 $\mathbf{x}\in\mathbf{X}_j$ 与该组代表向量的残差向量 $\mathbf{x}-\mathbf{C}_j$, 并记录残差向量的乘积量化编码. 对于一条查询向量 $\mathbf{y}$, IVFADC 会先对其进行粗糙量化 (coarse-quantization), 选择与 $\mathbf{y}$ 距离最近的 $\mathbf{C}_j$ 并计算残差向量 $\mathbf{y}-\mathbf{C}_j$. 随后 IVF 进行距离估计, 通过计算残差向量 $\mathbf{y}-\mathbf{C}_j$ 与乘积量化后的 $\mathbf{x}-\mathbf{C}_j$ 的非对称距离更新搜索结果. IVFOADC+G+P 希望能将每个倒排索引管理的区域划分成更小的子区域, 但存储全部子区域对应的子质心会导致码本占用过多的空间. 基于这个目标, IVFOADC+G+P 通过分组方法, 由区域质心和其邻近质心的凸组合来构造子质心以减少内存开销. 此外, 文献 [34] 提出了 RaBitQ, 并将其应用到 IVF 索引结构上, 取得了比倒排加乘积量化高的搜索性能.

IMI (inverted multi-index)[94]将空间划分为两个子空间的笛卡尔积 (等同于在乘积量化中将空间划分为 m=2 个子空间), 并为两个子空间分别维护码本, 根据两个大小为 K 的码本, 将空间实质划分为了 K^2 个区域并在每个区域内使用乘积量化编码残差向量. 由于 IMI 对空间进行了更细粒度的划分, 每个区域内含有的向量数量较少, 缩减了搜索空间以提升效率. IMI 方法在大规模数据检索上取得了很好的效果, 但是文献 [95] 指出 IMI 方法会导致很多空区域导致性能问题, 其提出一个内存高效的数据分组方法来设计基于倒排的高维向量检索系统, 在 10 亿级的 SIFT 和 DEEP 数据集上取得了更优的效果.

2.5.5 基于哈希和树的组织方法

此类方法用树结构组织 LSH 投影后的数据. 如 C2LSH[69]、QALSH[70]、R2LSH[96]利用 B+树来组织数据点哈希后的结果; SRS[40]可将数据投影到多维度空间, 因此采用 R 树来组织数据. 文献 [97] 提出 LSB-tree 以实现快速

而准确的检索, 其基本思想是相近的点其 Z-order 值也相近, 最终的索引结构是由多棵 LSB-tree 组成的 LSB-forest. 文中指出, 两棵树就可以取得很好的搜索效果. HD-index[98]是一个基于磁盘的索引方法. 文献 [98] 提出基于 Hilbert[99]空间填充曲线的 RDB-tree 来对数据进行组织. PM-LSH[100]利用 PM-tree[101]来组织投影后的数据. DET-LSH[102]通过动态编码树 (DE-Tree) 来编码根据数据分布投影后的向量, 以提升索引的构建效率. DET-LSH 将动态编码树与基于局部敏感哈希的方法结合, 通过两阶段的搜索方式提升召回率.

2.6 小　结

在实际中要基于数据量、业务场景以及成本等选择不同索引方案[63], 并在搜索性能、空间占用开销和维护难度等方面做出取舍. 针对只有几千条数据的小数据量场景, 无须维护复杂的索引结构, 采用直接搜索全部数据的方案就能够满足要求; 对于一些维度相对较低的数据, 采用基于树的方法, 如 Annoy[59]也可以很好地完成相关查询任务[63]; 面对搜索大量向量数据的情况, 基于图的方法, 如 HNSW、NSG 等方案的优秀搜索性能支持完成大部分任务. 但是, 仅使用基于图的方案并非适用于每个场景, 如表 3 所示, 其具有空间占用高、更新维护困难等问题, 当面对 10 亿级别大规模向量搜索以及数据动态更新频繁的业务时, 基于图的方法将面临巨大挑战, 此时就需要考虑更多的方法来找到解决方案, 并在搜索性能、维护代价、价格成本等之间取得平衡, 支撑上层业务平稳运行.

当面对的数据规模很大, 如超过 10 亿规模的情景, 内存资源有限导致无法在内存中存储全部数据的情况时, 使用 PQ、RaBitQ 等量化方法存储有损数据可以适配有限的内存空间. OG-LVQ 方案提供了将图和量化进行结合的方案, 并且不要求在索引中存储原始的向量数据, 有效地降低了内存的占用, 在获得较高的查询速度的同时保证较高的召回率. 除了面向内存的方案, 还可以选择面向硬盘的方案以解决内存空间限制导致超出了单机处理能力的问题, 此时 DiskANN 等基于磁盘内存混合的索引方法可以有效地完成任务; 向量数据库在数据动态变化的场景中面临着处理更新的挑战, 基于图的方法面临着更新困难的问题, 选择基于倒排文件 (IVF) 等构建速度快、维护难度低的索引可以很好地平衡维护成本与查询性能的关系, SPFresh 设计基于倒排的更新方案取得了良好的效果. 此外, 将倒排和量化方法进行结合使用是一个经常被采用的搜索方案. 局部敏感哈希方法具有理论保证, 可以用于需要精度质量承诺的场景中, 对搜索结果提供可量化的性能边界和可靠性支撑.

3 向量数据搜索优化方法

随着人工智能的发展, 向量近似最近邻搜索发挥了日益关键的作用, 在获得越来越广泛关注的同时, 应用场景也更加复杂多样, 在效率、精度和可扩展性等层面面临严峻的挑战. 为了应对挑战, 实现更高效的搜索以支持越来越广泛的应用场景, 近年来很多研究工作从多个维度探索, 包括硬件加速、距离比较操作优化、数据访问优化等研究方向, 并取得了一系列的成果, 共同推进构建了高维向量搜索的解决方案.

本节系统性地梳理近年来的研究成果, 依据不同技术路径的差异, 从面向硬件加速、面向学习增强、面向距离比较操作、面向磁盘内存混合场景、面向数据访问优化、面向分布式场景、面向混合查询场景和理论分析这 8 个方面对相关工作进行梳理. 表 5 从核心思想、典型方法、优势和局限性这 4 个方面概括展示了整体的优化方法.

表 5　向量搜索优化方法总览和对比展示

分类维度	核心思想	典型方法	优势	局限性
面向硬件加速的优化	利用硬件特性加速搜索	AVX指令集、GPU加速、多线程等	能够大幅提升吞吐量, 降低延迟	硬件成本高, 移植复杂度高
面向学习增强的优化	利用机器学习方法来挖掘数据特性	Neural LSH[103]、BLISS[104]等	适应数据分布, 减少冗余计算	训练复杂度高, 模型泛化性要求高
面向距离比较操作的优化	定位算法高开销部分, 设计优化算法降低距离计算	ADSampling[105]、PEOs[106]等	降低计算代价, 通用性强	对算法设计要求高, 并且收益受到数据的维度、分布特性等影响

表 5 向量搜索优化方法总览和对比展示 (续)

分类维度	核心思想	典型方法	优势	局限性
面向内存硬盘混合场景的优化	优化磁盘访问的I/O效率	DiskANN[36]、SPANN[107]等	利用硬盘降低成本, 提升单机处理能力	较慢的磁盘IO影响搜索性能
面向数据访问的优化	重组数据结构提升局部性, 充分利用缓存	QG-LVQ[35]、SymphonyQG[108]等	提升缓存命中率, 减少传输带宽压力	对算法设计要求高, 空间换时间带来额外的内存占用
面向分布式场景的优化	分布式并行处理与负载均衡	Pyramid[109]、Auncel[110]等	横向扩展能力强, 处理超大规模数据	网络通信成为瓶颈, 任务调度复杂性高, 增加算法设计和维护的复杂性
面向混合查询场景的优化	支持高效向量+结构化数据混合查询	Filtered-DiskANN[111]、ACORN[112]、SeRF[113]等	满足负载业务逻辑, 为应用提供更多样支撑	支持场景复杂性增加导致查询优化器设计复杂, 高效索引设计困难
理论分析	用数学模型证明复杂度边界和性能极限	最坏情况分析[114]、LID[115]、Steiner-hardness[116]	指导算法设计, 提供性能保证	理论假设偏离系统数据实际情况

3.1 面向硬件加速的优化

3.1.1 利用 SIMD 加速距离计算

单指令多数据流 (single instruction multiple data, SIMD) 技术通过一条指令同时处理多个数, 显著提升计算密集型任务的性能. 现代 CPU 往往支持 SIMD 操作, 比如 x86 架构的 AVX/AVX2/AVX512 指令集[117]和 ARM 架构的 NEON[118].

利用 128 位 SIMD 寄存器加速计算向量 $\mathbf{x}$ 与向量 $\mathbf{y}$ 的欧氏距离的平方的一种方案如图 6 所示. 其中, 第①与第②步分别将 4 个元素从内存加载到 SIMD 寄存器中. 第③步通过 sub 指令计算得到 $\mathbf{z}=\mathbf{x}-\mathbf{y}$. 第④步利用 fmadd 指令实现 $res_i = res_i + z_i \cdot z_i$. 第⑤步与第⑥步通过执行两次 hadd_ps 指令将 result 寄存器中的值求和并保存在首个元素中. 第 7 步通过 cvtss 指令提取寄存器中的首个元素, 得到计算结果.

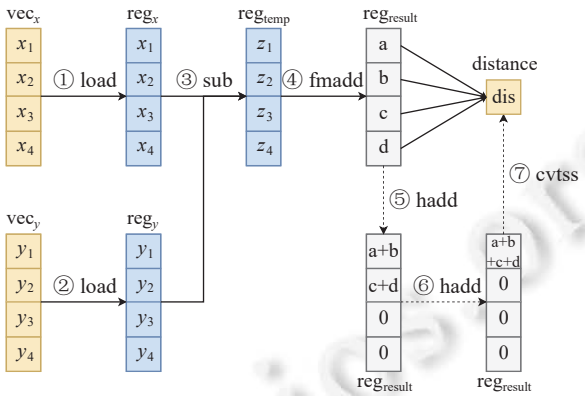


图 6 SIMD 计算距离示例图

通过 SIMD 来加速向量计算已经广泛地用于向量搜索中, Milvus[1]等向量数据库以及 Faiss[29]等算法库利用了 SIMD 优化近似最近邻查询. Faiss 从 3 个层次利用了 SIMD: (1) 对于较为简单的操作 (比如对两个向量求和), 在代码实现时会通过较为紧凑的循环或使用某些关键字使得编译器能够自行实现向量化 (auto-vectorization); (2) 利用通过 C++编译器扩展实现的 SIMD 变量和指令; (3) 通过优化数据布局与算法设计来更好地利用 SIMD. Milvus 针对 SIMD 主要做了两个工程优化: (1) 支持 AVX512 指令集; (2) 为不同架构的 CPU 自动识别并选择对应的 SIMD 指令.

表 6 展示了进行 Top20 查询获得 96% 和 98% 召回率时, 在 Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz

CPU 下, HNSW 索引在 GIST 和 DEEP 数据集上不使用 SIMD 加速和使用 SIMD512 指令集进行加速的查询性能 (QPS), 可以看出, 通过开启 SIMD 来加速距离计算, 可以获得最多 2.6 倍的性能提升.

表 6　HNSW 在开启 SIMD 和不开启 SIMD 时查询性能 (QPS) 对比

SIMD加速状态	96%召回率			98%召回率		
	DEEP	GIST	MSong	DEEP	GIST	MSong
NonSIMD	542	87	1 375	407	54	1 067
SIMD	1 291 (2.3×)	227 (2.6×)	2 518 (1.8×)	886 (2.1×)	141 (2.6×)	1814 (1.7×)

3.1.2　利用多线程的优化方法

并行设计已经普遍应用在计算机系统中, 通过利用 CPU 的并行能力, 同时执行多个操作来提升性能. 当前多线程的加速可以分为两类: 多线程构建和多线程查询加速. 构建的多线程是将数据集划分为多个子集, 然后在多个线程中并行执行插入操作, 最后将所有的结果进行合并来得到最终的结果. 查询的多线程是将查询操作分为多个子任务, 然后在多个线程中并行执行, 最后将所有的结果进行合并来得到最终的结果.

(1) 构建的多线程优化

基于图的方法已经广泛应用在向量数据库中. 图的构建是一项十分耗时的任务, 通过并行技术来加速图的构建是一个可行的方案. 当前 HNSW 图的构建是将所有数据点按照一定顺序插入图中, 当插入点 **p** 时, 算法在 **p** 和图中的现有点之间添加新的边, 其依赖算法 1 在当前图中搜索获取相应的候选点, 然后执行对应的选点操作完成邻居节点的构建. 文献 [119] 提出了一种基于多线程的图构建方法, 利用多线程来加速图的构建过程. 其主要思想是将数据集分为多个批次, 在每个批次上执行多线程插入操作, 同时确定多个点的邻居节点.

具体来说, 采用指数级批量增加方法来逐渐增加每个批次的点的数量. 初始状态, 每个批次的点的数量较少, 这更类似于顺序版本, 从而允许更高质量的图, 后续随着图变得更大, 允许一批中有更多的数据点, 从而实现更高的并行度, 同时, 对批的大小设置一个上限避免无限增大. 通过这种方法, 实现了在性能和图的质量之间的平衡. 在对每个点构建邻居节点的过程中, 第 1 步是确定其邻居节点 (出边), 第 2 步是确定 **p** 是否可以作为这些节点的邻居 (入边). 对于前者, 批处理中的所有数据点在不可变快照上独立构建自己的邻域, 因此不会相互影响. 对于后者, 会收集当前批次在第 1 步添加的出边, 然后统一对出边指向的节点执行选点操作来确定入边, 从而实现无锁的并行入边构建.

(2) 搜索的多线程优化

多线程同样可以用来加速搜索过程. 利用多线程的一个基本方案是, 对于多个查询同时到来的情况, 并发地执行多个查询. iQAN[120]是一个利用多线程来加速单个查询的性能的方法. 基于图的方法是在图上不断地遍历数据点的过程, iQAN 的核心思想是并行地在图上检查数据点, 通过同时搜索多个路径上的点, 实现同时对多个范围内的数据进行检查, 然后将检查的结果合并, 在合并结果的基础上进行新的多个范围的检查. 主要搜索策略包括: (1) 路径并行, 通过多个线程同时搜索多个不同的路径实现并行搜索; (2) 分阶段扩展, 随着搜索的进行, 逐步扩展线程数量, 用更多的路径来实现并行搜索, 在减少起步阶段的开销的同时提高后期的遍历效率; (3) 减少同步开销, 允许不同的工作线程检查重复的点, 避免了线程之间的同步开销, 在合并时统一进行结果去重操作. 在这些设计的基础上, iQAN 最终实现了利用多线程来加速搜索过程, 提升了搜索性能, 在 SIFT1B 和 DEEP1B 数据集上实现了最多 16 倍的性能提升.

3.1.3　利用 GPU 进行加速

基于 CPU 的近似最近邻搜索方案面临着高维向量密集计算代价高、实时性能要求高、维护开销大的问题. GPU 可以凭借大规模并行架构和高内存带宽的优势突破性能瓶颈, 如其可以批量计算向量之间的距离, 实现高效的图构建等, 当前有很多工作研究如何利用 GPU 来优化近似最近邻搜索.

SONG[121]是一个基于 GPU 加速的近似最近邻搜索系统. 该系统针对基于图的方法进行了深度优化, 首次实现了基于图的 GPU 向量搜索框架. SONG 将搜索过程解耦为 3 个阶段: 候选定位 (candidates locating)、批量距离计算

(bulk distance computation) 和数据结构维护 (data structure maintenance). 候选定位阶段从图索引中提取当前节点的邻居顶点; 批量距离计算阶段利用 GPU 的并行计算能力高效完成高维向量的距离计算; 数据结构维护阶段则更新优先级队列和哈希表以准备下一轮迭代. 通过这一解耦设计, SONG 将原本串行的距离计算转化为批量并行任务, 显著提升了 GPU 计算资源的利用率. 此外, SONG 针对 GPU 内存特性设计了多项优化, 包括采用固定度图 (fixed degree graph) 存储, 通过全局内存的连续布局减少索引访问开销; 利用共享内存等方案; 使用布隆过滤器 (bloom filter) 或 Cuckoo 过滤器替代传统哈希表, 在允许可控误报率的前提下减少显存占用; 提出选择性插入 (selected insertion) 和访问删除 (visited deletion) 策略, 仅保留与当前最优候选相关的节点信息, 将哈希表内存消耗限制为 $2K$ (K 为搜索参数). 为了进一步应对 GPU 的显存限制, 引入了随机投影来降维数据. 实验结果表明, SONG 在 GPU 上实现了基于图 ANNS 的突破性加速, 相比于单线程 HNSW 和 Faiss 分别提升 50–180 倍和 4.8–20.2 倍吞吐量.

GANNS[122]是一个基于 GPU 加速的邻近图近似最近邻搜索与构建的框架. 该框架针对现有 GPU 图搜索方法 (如 SONG) 在数据结构操作上的瓶颈, 设计了惰性更新 (lazy update) 与惰性检查 (lazy check) 策略, 以充分释放 GPU 的并行计算潜力. GANNS 将传统串行搜索流程重构为如图 7 所示的 6 个并行化阶段: 候选定位 (candidate locating)、邻域扩展 (neighborhood exploration)、批量距离计算 (bulk distance computation)、惰性检查 (lazy check)、排序 (sorting) 和候选更新 (candidate update). 其中, 候选定位阶段通过线程束 (warp) 级别的位掩码操作快速定位待探索节点; 邻域扩展阶段将当前节点的邻居批量加载至共享内存; 惰性检查阶段则通过并行二分搜索过滤已处理节点, 避免冗余计算. 与传统方法不同, GANNS 采用固定长度数组 (而非动态优先级队列) 维护候选集, 并利用 GPU 友好的排序算法 (如 Bitonic Sort) 批量更新候选节点顺序. 这一设计不仅消除了动态内存分配的同步开销, 还通过线程块 (thread block) 内的协作机制实现数据结构操作的并行化. 此外, 文献 [122] 中提出分治策略 GGraphCon, 将数据点划分为多个子集, 并行构建局部 NSW 图后逐步合并, 首次实现 GPU 端高效的 NSW 图与 HNSW 图构建. 实验结果表明, GGraphCon 在构建 NSW 图时相比于单线程 CPU 方法加速 40–50 倍.

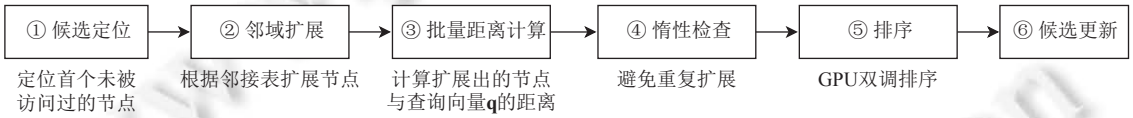


图 7 GANNS 的工作流程

根据 GANNS 的实验结果[122], 其性能与 SONG 相比, 主要结论如下. (1) 在查询性能上, 相同召回率下, GANNS 的 QPS 比 SONG 高 1.5–5 倍. 例如, 在 SIFT1M 数据集上, 当召回率为 0.795 时, GANNS 达到 458.5k QPS, 而 SONG 仅为 88.5k QPS. (2) 在召回率上, 两者在相同数据集上达到相似的召回范围, 表明 GANNS 的并行化未牺牲结果精度. (3) 在性能瓶颈上, SONG 的瓶颈是数据操作 (占 50%–90% 时间), 因其依赖单线程处理优先级队列和哈希表; GANNS 通过懒策略 (lazy update 和 lazy check) 减少数据操作开销, 使距离计算主导时间, 优化了 GPU 利用率. (4) 在查询性能鲁棒性上, 当返回邻居数 k 从 1 增至 100 时, GANNS 的速度提升稳定 (在 SIFT1M 上提升 5–5.3 倍, 在 GIST 上提升 1.5–2 倍), 而 SONG 因数据操作瓶颈在高 k 值时效率下降更明显. (5) 数据维度的影响: GANNS 在低维数据集上性能提升更明显, 如将 GIST 从 960 维降到 60 维时, 相较于 SONG 的性能提升从 1.5 倍提升至 6 倍, 因其能充分利用线程并行处理.

此外还有很多工作利用 GPU 优化性能. GENIE[123]通过倒排索引将查询分解成子任务以充分利用 GPU 的并行计算能力, 并提出了一种基于 GPU 实现的哈希表 c-PQ, 用于从候选集中选出 TopK 作为结果. PQT[124]对乘积量化进行拓展, 提出了一种双层次的乘积量化树以减少精确距离测试的次数, 通过设计遍历顺序与重排序算法提升搜索性能, 并给出了基于 GPU 的实现. 文献 [125] 提出了在 GPU 上实现 IVFADC 的方法. RobustiQ[126]提出了结合基于量化的层次化倒排索引提升搜索效率与系统的鲁棒性的 GPU 上的搜索方法. GGNN[127]设计了一种 GPU 友好的线程块级别的搜索方式, 通过全并行多用途缓存与对节点近邻数量的固定提升了片上的资源利用率. GGNN 还设计了一种自下而上的图索引构建方法, 提升了索引构建的速度. GTS[128]提出了一种基于 GPU 的索引. 该索引

具有树状的层次结构并通过基于表的方法进行存储, 实现了对树上非连续节点的并行计算. 此外, GTS 设计了一种并发搜索策略以预防内存死锁, 并引入一个成本模型以平衡并发与剪枝效率. BANG[129]研究如何在显存受限的情况下通过单 GPU 处理大量数据. BANG 利用 CPU 与内存处理索引与数据, 并利用 GPU 加速距离计算, 还通过阶段性执行策略优化了 GPU-CPU 的负载均衡. CAGRA[130]面向英伟达 GPU 设计了基于图的近似最近邻搜索算法. 实验结果显示, 相较于 CPU 上的方法, CAGRA 性能获得了大幅提升.

3.2 面向学习增强的优化

人工智能技术已经被广泛用于优化数据库性能, 其中学习型索引近年来取得了很大的进展. 学习型索引的核心思想是通过学习数据的分布来优化索引结构和查询过程, 从而提高系统的性能. 当前有很多工作利用学习的方法来优化高维向量近似最近邻搜索的性能, 通过采用学习的方法, 能够充分挖掘高维向量的分布特征, 从而有针对性地优化算法. 当前利用学习的方法来优化高维向量索引主要是为了能够基于数据的分布来更好地划分数据, 从而能够使查询更高效、准确地定位到邻近点所在区域, 减少数据的访问开销. 下面我们介绍相关的技术方案.

(1) Neural LSH 索引

文献 [103] 提出的 Neural LSH 方法利用神经网络将数据更好地划分到多个桶中. 传统的利用聚类、LSH 以及树的方法难以捕捉到数据的分布特征. Neural LSH 通过神经网络来学习数据的分布特征. 它利用图划分技术和有监督学习的方法来优化数据的划分, 图 8 展示了其划分示意图. 该方法主要分为 3 个重要的步骤: (1) 构建一个 KNN 图, 利用 KNN 图来捕捉数据的分布特征; (2) 利用平衡图划分技术将数据均匀地划分到多个桶中; (3) 训练一个基于神经网络的分类器, 将其应用到整个数据空间, 对所有的数据进行划分. 为了提升划分的效果, 该方法采用层次化的划分结构, 逐层递归地划分子空间, 先划分为大的区域, 然后对每个区间做进一步的划分. 实验结果表明, 在通过 Neural LSH 划分的数据上进行搜索能够取得优于传统的聚类方法和 LSH 方法的性能.

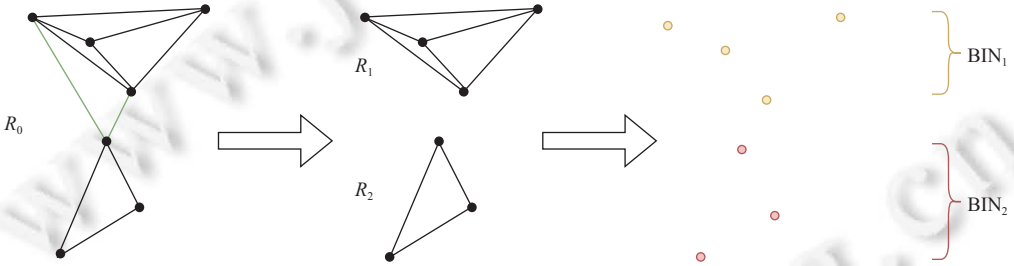


图 8 Neural LSH 划分示意图

(2) BLISS 索引

BLISS[104]采用迭代的方式来优化数据的划分, 通过交替进行以下两个步骤来优化分区: (1) 通过学习将数据映射到对应的桶; (2) 以每个桶中数据均衡为目标重新分配数据点. 在训练映射函数阶段针对每个数据点会学习到针对桶的评分函数, 并每次在评分最高的 K 个桶中选择负载最小的桶来进行分配, 总共训练 R 个映射函数. 在推理阶段, BLISS 通过 R 个已训练好的映射函数对数据点进行映射, 每个映射函数分别选择 5 个桶来获取候选结果, 在这些候选结果中进一步筛选以获取最后的查询结果. 实验结果表明, BLISS 在多个数据集上都取得了优于 Neural LSH 的性能.

(3) BATL 索引

BATL[131]是一个基于平衡 K 叉树的学习型层次划分索引, 同一个桶中的数据被一条从根节点到叶子节点的路径表示, 查询过程是从根节点不断路由到叶子节点的过程. 其将路由任务转化为分支序列生成任务, 利用 Transformer 模型[132]设计了一个解码器编码器框架, 在搜索时利用这个框架和束搜索 (beam search) 来生成分支序列, 动态地将查询从根节点路由到叶子节点. BATL 的训练过程首先随机初始化一棵平衡树, 然后随机选择部分数据点作为查询点, 根据当前的树结构生成 (查询, 路径) 对作为训练数据, 将任务建模为自回归分类问题, 采用序列到序列的学习范式进行训练. 其整个训练过程是交替迭代的, 固定树结构, 训练路由模型; 固定路由模型, 更新树结

构, 逐渐完成整个训练过程. BATL 训练多个树模型以提高召回率, 实验展示出其在时延、准确性和内存占用之间取得了良好的平衡.

(4) LIDER 索引

LIDER[133]是一个基于聚类的双层层次结构的学习型索引. 它包括两个重要的组成部分, 第 1 层维护了聚类的中心点, 第 2 层维护了聚簇数据. 在这两层中, 都包含一个称为 core model 的组件, 其包含两个部分: (1) 降维模块; (2) 一个位置预测模块 RMI. 降维模块又分为两个部分, EK-LSH 用来将数据映射成哈希键, 键重缩放组件用来将字符类型的哈希键映射到数值型, 用于 RMI 模块的训练. LIDER 的整体工作流程是: 首先将数据利用 K-means 聚类, 然后对聚类中心点和每个聚类都训练对应的 core model, 最后将所有的 core model 组合成一个索引. 在搜索阶段, 查询首先搜索到对应的聚类中心点确定要搜索的聚类, 然后在对应的聚类中进行搜索, 多个聚类采用并行的方式进行搜索, 最后将来自每个聚类的结果合并, 取前 k 个结果返回. 实验结果表明, LIDER 与已有的索引相比, 在取得更高搜索速度的同时具有更高的召回率.

表 7 对比展示了上述 4 个方法. 此外, 文献 [134] 利用梯度下降树学习到提前终止搜索策略, 以用于在向量搜索的过程中提前终止从而减少冗余搜索. 文献 [135] 提出一个基于编码解码器的方案 PCE-Net 来确定需要参与搜索的倒排列表的数量, RPQ[136]为基于图的方法设计了一种路由指导的端到端的学习型乘积量化方法, 其通过采样并提取出路由特征与近邻特征用于训练多个可微的量化器, 使得量化结果更好地适应基于图的近似最近邻搜索. 文献 [137] 提出利用学习的方法来设计面向磁盘的 I/O 优化的方法. 它通过学习的方法来使投影后点的相对顺序和原始高维空间尽可能一致从而减少随机访问. SmartANNS[138]利用学习的方法来确定需要搜索的数据分片所在的 SmartSSD 来实现更高的设备利用率.

表 7　4 个学习增强方法的对比

索引技术	核心思想	组织形式	技术特点	优点	缺点
Neural LSH	利用神经网络学习数据分布, 结合图划分技术优化数据分层划分	递归划分数据形成多层桶结构	有监督学习、图划分技术结合	优于传统LSH和聚类方法的搜索性能	训练开销大, 层级划分可能引入误差
BLISS	迭代优化数据划分: 交替训练映射函数与数据重分配	多映射函数+负载均衡桶	迭代式负载均衡策略	优于Neural LSH的性能, 索引的空间占用比HNSW低	映射函数数量影响性能, 候选桶筛选计算复杂
BATL	将路由任务建模为序列生成问题, 基于Transformer的编解码框架动态构建平衡K叉树	平衡K叉树	树结构+自回归序列学习	树与Transformer进行结合, 技术新颖	训练需反复迭代树结构, 模型复杂度较高
LIDER	基于聚类的双层结构 (中心点+聚簇), 结合降维与位置预测模块	聚类中心层+聚簇数据层的双层结构	降维技术与聚类结合	并行友好	聚类中心质量依赖初始K-means

3.3 面向距离比较操作的优化

在向量搜索中, 计算代价在性能开销中占据了很大比重, 其主要是由在搜索的过程中计算两个点的距离产生的, 通过优化计算代价可以提升系统的性能. 文献 [105] 中提出, 在搜索过程中涉及共同的一个关键步骤, 比较两个点之间的距离是否小于一个阈值 τ, 如果小于 τ, 则返回两个点之间的距离, 这个过程被称为距离比较操作 (distance comparison operation, DCO), 距离计算操作主要发生在这个过程中. 为了完成这个操作, 基本方法是计算两个点之间的距离, 然后判断是否小于 τ. 文献 [105] 指出, 对于大部分的点, 两个点之间的距离都是大于 τ 的, 不需要计算其真实距离, 只需要判断是否小于 τ 即可, 这带来了优化距离计算的机会. 如图 9 所示, 仅计算部分距离并与阈值比较, 在不满足要求的情况下及时终止距离计算, 从而优化计算代价.

ADSampling[105]提出利用假设检验的方法来估计距离是否大于 τ, 其核心是采用随机投影的方式将数据投影到子空间, 估计两个点的距离, 然后通过假设检验判断两个点的距离是否大于 τ. 这是一个迭代的过程, 通过不断采样更多的维度来优化估计的精度直到点被过滤或者真实距离被计算出来. 实验结果显示, 将这一方法应用于 HNSW 和 IVF 索引后减少了距离计算, 进而提高了搜索性能. DDC[139]在 ADSampling 的基础上, 将随机投影改成

利用 PCA (principal component analysis) 投影, 以最小化近似距离和真实距离的误差, 然后利用基于标准差的方式进行距离校正, 最后提出一个利用线性模型从数据中学习到的更通用的距离校正方法. 实验结果显示, 与 ADSampling 相比, DDC 实现了 1.6–2.1 倍的性能提升. DADE[140]提出了一个基于数据分布的距离估计方法, 从理论上证明了 DADE 的距离估计在数据分布上是无偏的. DADE 利用主成分分析获得的矩阵对数据进行正交变换, 实验结果显示其效果优于 ADSampling.

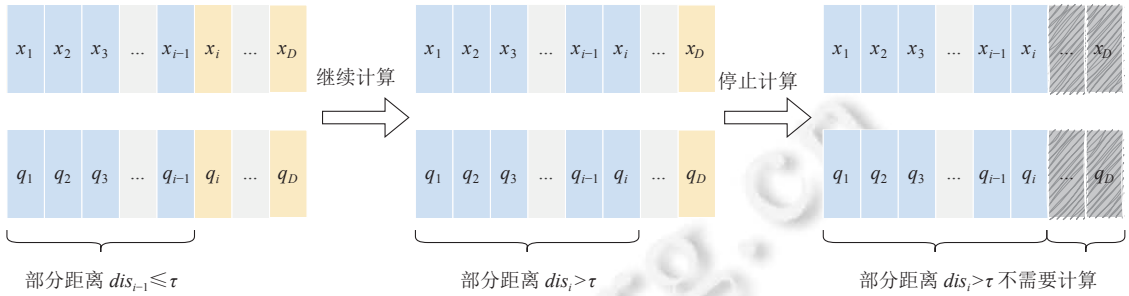


图 9 DCO 时提前终止距离计算

表 8 展示了 ADSampling、DADE 和 DDC 在 3 个数据集下进行 Top20 查询时分别在 96% 和 98% 召回率下的搜索性能 (QPS) 对比, “—”代表默认参数下无法达到对应的召回率, 从结果中可以看到, 不同方法在不同数据集上的性能表现不同, 比如, 在 DEEP 数据集上, DDC 方法能够取得更好的效果, 而在 MSong 数据集上, ADSampling 效果更好. 在实际使用中需要根据数据集来选择合适的方法, 以取得更好的效果.

表 8 采用提前终止距离计算的 ADSampling、DADE 和 DDC 的搜索性能 (QPS) 对比

优化方法	96%召回率			98%召回率		
	DEEP	GIST	MSong	DEEP	GIST	MSong
ADSampling	728	209	1 972	520	138	1 492
DDC	1 329	281	1 183	915	192	—
DADE	1 215	208	1 650	844	185	931

FINGER[141]同样是一个优化距离比较操作的方法. 它不是通过提前终止距离计算, 而是通过估计残差向量之间的角度来近似距离函数, 从而跳过对一些点的距离计算. 在此基础上, 它利用改进的局部敏感哈希方法来降低计算成本, 相较于 ADSampling 方法, 需要更多的存储代价. 实验结果显示, FINGER 在搜索上相较于 HNSW 提升了最多 60% 的性能. PEOs[106]提出了概率路由的概念, 为搜索中探索邻居节点提供概率保证, 从而高效选择需要精确距离计算的邻居节点. 它利用空间划分和随机投影技术来估计邻居节点和查询向量的角度, 然后根据估计的角度来选择需要精确计算距离的邻居节点. 与 FINGER 相比, PEOs 显著减少了额外的存储代价, 在 GIST 数据集上由 6.12 GB 降低到 4.64 GB, 实验结果显示其性能相比于 FINGER 提升了最多 1.4 倍.

3.4 面向磁盘内存混合场景的优化

磁盘内存混合场景是指系统可用物理内存容量不足以直接存储全部待处理数据, 或内存使用成本过高迫使必须采用混合存储策略的计算环境. 此类场景常见于超大规模数据应用 (如 10 亿级向量检索), 其核心矛盾在于内存容量与数据规模的不匹配性. 高维向量数据每一条数据包含成百上千标量数据, 存储 1 000 万条 128 维的 float 类型向量需要 4.7 GB 的空间, 建立的索引还会引入额外的空间占用, 在实际中需要处理更大规模的数据, 全部依赖内存将带来极大的成本开销, 此时无法采用传统内存索引. 因此, 基于磁盘内存混合的近似最近邻算法被提出来, 以降低对内存容量的需求.

3.4.1 基于倒排的方法

SPANN[107]是面向内存受限场景的高效近似最近邻搜索系统. 该系统基于倒排索引框架构建了一种内存-磁盘

混合索引结构. SPANN 在内存中存储倒排列表的质心点作为粗粒度索引, 并将大规模原始数据划分到磁盘中的倒排列表, 图 10 展示了其基本的结构. 索引构建阶段采用分层平衡聚类算法, 通过多约束优化将数据递归划分为均匀的子集, 控制单个倒排列表的最大长度, 从而降低单次磁盘访问开销. 针对边界向量易丢失的问题, 提出闭包多簇分配策略: 对靠近多个簇质心的边界点进行复制存储 (如最多 8 个副本), 提升其召回概率. 该方法在保持倒排列表长度平衡的同时, 仅增加 20% 存储开销即可有效缓解近邻搜索的边界效应问题.

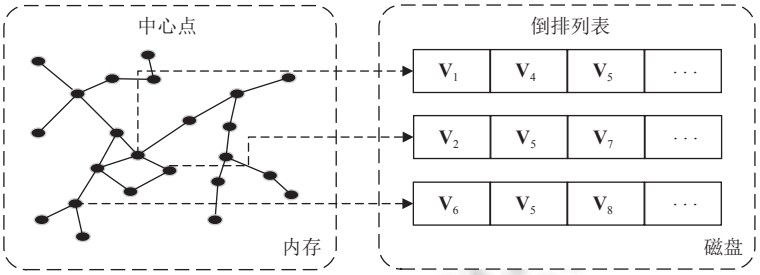


图 10 基于文献 [67] 展示的 SPANN 的索引结构

在查询优化方面, SPANN 设计了查询感知的动态剪枝机制. 搜索时优先从内存中的 SPTAG 索引 (基于空间划分树与近邻图) 快速定位 K 个最近质心, 然后根据查询与质心的距离动态调整候选列表: 仅访问与最近质心距离相差在阈值范围内的倒排列表 (如设定相对误差阈值 $\varepsilon = 0.6$). 这种自适应策略使得不同难度的查询获得差异化的搜索深度, 相比于固定候选数策略减少 40% 磁盘访问. 与 DiskANN 对比, SPANN 在相同 32 GB 内存配置下实现 2 倍加速, 召回率超过 90%. 在分布式场景中, 通过多约束平衡分片与查询负载预测算法, 将 10 亿级数据均匀分布到 32 台机器, 使单查询平均访问机器数降至 6.3 台, 比随机分片降低 80% 计算开销.

3.4.2 基于图和量化的方法

DiskANN[36]是基于 SSD 的图索引方案, 其核心是新型图索引算法 Vamana. 该算法通过引入可调节参数 α ($\alpha \geqslant 1$) 优化图结构直径, 在保证高召回率的同时减少搜索路径长度. 与 HNSW 和 NSG 相比, Vamana 的图结构在 10 亿级数据规模下可将搜索路径的磁盘随机访问次数降低到 1/3–1/2, 同时支持通过参数 α 灵活平衡图密度与搜索效率. 与 HNSW 等传统图索引相比, Vamana 通过调节 α 参数实现了更灵活的图密度与路径长度平衡. 此外, Vamana 在构建过程中采用反向边插入策略, 增强图的连通性, 避免局部最优陷阱.

DiskANN 通过分片和内存-磁盘混合索引存储的策略, 将 10 亿级索引部署于单节点的 64 GB 内存与 SSD 组合环境中. 首先, DiskANN 采用分治的策略, 通过 K-means 聚类将数据集划分为 40 个子集, 每个数据点分配至最近的 2 个子集, 并在各子集上独立构建 Vamana 子图索引; 最终合并所有子图形成全局索引, 确保跨子集的查询路径连通性. 其次, DiskANN 结合量化压缩与全精度缓存降低内存占用; 使用乘积量化 (PQ) 将向量压缩至 32 字节存储于内存, 同时在 SSD 中保存全精度向量与图结构. 搜索时, 通过束搜索批量预取多个节点的邻居信息, 减少 SSD 随机访问次数; 并利用 SSD 的 4 KB 对齐读取特性, 将全精度向量与邻接列表存储于同一磁盘扇区, 实现隐式重排序 (implicit re-ranking), 以压缩向量引导搜索方向, 最终通过全精度距离计算提升召回率. 此外, DiskANN 采用热点缓存策略, 将距离搜索起点 3–4 跳内的节点数据预加载至内存, 进一步减少磁盘 I/O. 实验结果表明, DiskANN 在 SIFT1B 数据集上仅需 348 GB SSD 空间与 64 GB 内存, 即可支持每秒 5 000+次 *recall*@1 达 95% 以上的查询. 这种“内存压缩+SSD 原始向量”的混合存储模式为 10 亿级 ANNS 任务提供了高精度、低延迟与低成本的单机解决方案. 此外, Starling[142]是一个基于图的磁盘内存索引方案. 它分为磁盘和内存索引两部分, 内存部分通过采样来建立摘要图以实现快速定位到查询区域附近, 而磁盘部分采用数据重排序来提升数据的局部性访问以降低磁盘 I/O.

SPANN 和 DiskANN 均通过粗粒度内存索引引导细粒度磁盘访问解决 I/O 瓶颈. SPANN 以倒排索引为核心, 通过数据均匀划分、边界点冗余和动态剪枝优化磁盘访问效率, 在低延迟和低内存场景下优势显著, 且构建速度更快, 但需要存储点的冗余副本数据, 因此比较消耗磁盘空间. DiskANN 以图索引为基石, 结合量化压缩与硬件特

性减少 I/O, 同样可以在大规模数据下达到高召回率, 但其基于图的算法, 面临着构建时间很长的问题, 且难以应对索引更新的场景.

3.5 面向数据访问的优化策略

向量搜索的性能同时受到计算代价和数据访问代价的影响, 尤其是在基于图的方法中, 数据的随机内存访问方式导致 cache 的预取操作失效, 带来了高昂的访存代价, 通过优化数据的访问可以提升搜索的性能. 当前提出了很多方法来优化数据的访问方式, 主要包括: (1) 数据重排序优化, 通过改变数据的存储顺序来提升数据的访问性能; (2) 数据预取优化, 数据预取是通过提前加载数据到缓存中来提升数据的访问性能; (3) 数据压缩优化, 通过压缩数据来降低数据大小从而 减少访问代价.

(1) 数据重排序优化

在图的搜索中, 需要不断地访问邻居节点的数据. 文献 [143] 指出, 40% 的时间消耗在内存访问上, 尤其是向量数据的获取, 如果能够优化相关时间, 将能极大地提升搜索性能, 基于此, 提出了数据布局优化的方法, 通过调整节点的内存布局, 使相邻节点的数据在内存中邻近存储, 利用硬件预取机制减少缓存缺失, 从而加速查询. 文章在理想化缓存模型下分析图遍历成本, 依据经典的图重排序方法[143]进一步优化, 提出了基于查询的加权重排序算法 Porder 来优先优化重要的邻居节点, 实验结果表明, Porder 实现了最多 40% 的延迟减少, 为后面的图索引优化提供了新方向.

Starling[142]是一个面向磁盘图索引方案的内存布局优化方法. 它提出一个基于磁盘的重排序方案来增强数据访问的局部性, 减少磁盘访问的开销. 它首先采样部分向量数据, 在内存中构建导航图以实现在内存中快读定位接近查询点的邻居节点, 减少磁盘 I/O 开销. 对于磁盘中的数据, 采用块重排将相邻顶点尽可能存储在统一磁盘块中, 提升数据局部性, 使得单次磁盘 I/O 能加载更多相关数据. 文献 [142] 证明块重排序问题是 NP 难问题, 提出启发式策略来进行近似求解. 实验结果表明, Starling 方案实现了 43.9 倍的吞吐量提升. 此外, 通过优化布局还可以加速索引的构建速度, Flash[144]通过优化数据布局同时充分利用 SIMD 来加速图索引的构建速度. PDX[145]针对 IVF 索引方法设计垂直布局方案来优化搜索速度.

(2) 数据预取优化

数据的随机访问限制了数据的硬件预取, 但是可以通过软件预取的方式预判数据的访问模式, 将数据提前加载进缓存中以提升数据的访问性能. VSAG[146]提出了利用预取来进行数据的访问优化的方法. 它包括软件预取和硬件预取优化两个部分. 软件预取是通过分析数据的访问模式来提前加载数据到缓存中提升缓存命中率. VSAG 通过提前将还没有被访问过的点预取到缓存中来提升数据的访问性能, 针对可能发生的缓存驱逐问题, 进一步提出了基于步长的预取方案以根据计算速度确定预取时机, 避免了缓存的驱逐问题. 硬件预取是通过利用 CPU 的硬件预取机制来提升数据的访问性能, VSAG 通过根据服务器的负载状况将邻居节点的数据额外存储在连续内存位置的冗余向量存储方法来实现顺序内存访问.

(3) 数据量化优化

通过量化后降低要处理的数据, 可以减少内存中的数据量, 从而降低需要处理的数据总量, 具有处理更多数据点的能力. 通过量化的方法可以压缩数据量, 代表性方法如 NGT-QG[147]、OG-LVQ[35]和 SymphonyQG[108]. SymphonyQG 是一个利用最新的 RaBitQ[34]量化方法和图方法协同集成的索引方案. 它结合了量化和利用布局两种优化手段. 通过调整图拓扑结构, 将图的每个节点 **p** 的邻居节点数据的量化编码连续存储在 **p** 所在位置的邻近内存中, 避免了访问邻居节点时随机访问的开销, 由于存储的是量化数据, 因此减少了重复地存储节点带来的内存开销, 通过空间换取时间的方式提升了查询性能. 相较于 NGT-QG 方案, SymphonyQG 能够提供更准确的距离估计, 并且其避免了显式的重排序从而降低了内存访问开销. 此外, RaBitQ 的量化方法的高计算效率, 加速了图的构建. 实验结果显示, SymphonyQG 在内存中实现了最多 4.5 倍的查询性能提升.

3.6 面向分布式场景的优化

面对大规模数据, 单机难以进行处理, 采用分布式的处理方法将数据分片到多台机器上协同处理是一个可行

的方案, 当前有许多方法探索如何利用分布式场景来实现高效的近似最近邻搜索.

(1) 基于算法框架的分布式优化

Pyramid[109]是一个基于 HNSW 的分布式搜索框架. 它支持欧氏距离、余弦相似度和最大内积搜索. 它通过采样数据构建 meta-HNSW, 然后采用图分割算法将图分割为子图, 其分割策略是最小化不同子图之间边的数量. 将数据集中的每个点分配到距离最近的子图中, 从而实现对数据集的分片. 再对每一个分片下的数据集分别构建 HNSW 索引. 查询时首先通过 meta-HNSW 定位相关数据子集, 然后分派查询到部分子集来进行查询处理, 通过这种方法能够充分利用计算资源, 提升吞吐率. 在系统架构层面, 它包括 3 个主要部件: 协调器 (处理查询分派)、执行器 (子 HNSW 搜索) 和消息代理 (Kafka 实现可靠通信), 支持异步处理和容错机制.

Auncel[110]是一个基于 IVF 索引的相似性搜索引擎, 用于解决分布式场景下的无法提供理论性能保证的问题. 其核心思想是通过单个查询向量的局部几何特性, 为每个查询构建精确的错误延迟概要 (error-latency profile, ELP). 此概要使 Auncel 能够对适量的数据进行采样, 以处理给定的查询, 满足其错误或延迟要求. 其搜索大致过程和基本的 IVF 索引搜索方法一致, 在完成搜索每个聚簇时额外添加了如下步骤, 利用中间搜索结果和 ELP 来预测当前的错误率, 如果错误率或者时限满足要求, 就终止搜索返回结果. 其分布式部署方案是将数据分片, 每个工作节点处理本地分片, 领导者节点聚合结果. Auncel 随机选择领导者节点来处理查询. 实验结果显示, Auncel 在满足错误或延迟限制的同时显著降低了查询延迟.

(2) 基于硬件的分布式优化

SmartANNS[138]是一个利用 SmartSSD 来解决 10 亿级别数据相似性搜索的方法. 它将数据分配到多个 SmartSSD 上, 实现高效的查询. SmartSSD 是一种将处理能力集成到 SSD 中的计算存储驱动器, 可直接在设备上处理数据, 减少对 CPU/GPU 传输的需求. SmartANNS 协同架构中主机维护分片质心, 作为全局协调器筛选搜索空间, 每个 SmartSSD 对分片数据建立 HNSW 索引执行搜索. 它允许数据复制分配到多块 SmartSSD 中, 并通过离线采样分析分片热度, 优化数据布局, 结合数据局部性和设备负载均衡调度查询, 并通过训练 GBDT 模型来动态地确定搜索分片范围, 避免冗余计算.

CXL (compute express link)[148]旨在实现处理器和设备之间的低延迟、高带宽连接, 近年来取得了越来越广泛的关注. CXL-ANNS[149,150]是一个利用软硬件协作来支持 10 亿级别向量搜索的方案. 它利用 CXL 将 DRAM 从主机中分离, 将所有的必要数据集放入内存池中, 针对搜索中的低延迟问题, 将访问频率高的点预先缓存在本地内存中, 减少对远程内存池的访问延迟, 对于未缓存的节点, CXL-ANNS 通过预测图的遍历行为, 对即将要访问的节点提前预取, 从而隐藏访问延迟. 进一步地, 通过 CXL 互联的层次化结构将搜索任务分发到多个硬件进行并行处理. 通过 3 层优化, 系统实现了对大规模向量数据的高精度低延迟搜索.

通过将数据分片处理, 可以有效地降低单机的处理负担, 实现对大规模向量数据的高效检索, 充分利用新硬件的能力来设计搜索架构可以更好地优化向量搜索. 未来将继续探索如何利用分布式来处理大规模数据下的向量检索, 设计高效的分布式索引结构和分布式检索算法是一个重要的研究方向.

3.7 面向混合查询场景的优化

混合向量搜索 (hybrid vector search) 研究同时考虑向量近似最近邻搜索和属性关键词信息是否满足要求的搜索问题, 以适应多样的复杂查询需求, 提供更准确、更全面的搜索结果. 支持混合搜索是向量数据库的一个重要功能, 为了完成混合搜索, 从搜索的实现上可以分为 4 种[63]: 先向量搜索后关键词 (post-filtering, 后过滤)、先关键词后向量搜索 (pre-filtering, 先过滤) 、单独搜索后合并、联合搜索. 对于先过滤的方案, 如果具有相同关键词的对象很多, 就会检查大量的数据; 对于后过滤场景, 如果要查询的关键词选择度很低, 就难以搜到相应的结果; 对于合并方案, 其在信息检索重排序阶段可以更灵活, 但是前期的搜索阶段会有很大的性能开销; 对于联合方案, 如何设计高效的索引是很大的挑战. 除了根据实现方式上的分类外, 从场景上还可以分为关键词过滤和范围过滤两类. 下面介绍具体的技术.

(1) 关键词过滤

Filtered-DiskANN[111]是一个基于图的混合查询索引方案. 它建立在 Vamana 图的基础上, 包括 FilteredVamana 和 StitchedVamana 两种方案. 前者从一个空的图开始进行构建, 通过动态剪枝和标签感知的搜索方法来确定每个节点的邻居节点; 后者建立在前者的基础上, 通过合并不同过滤标签下的子图以形成统一的索引. 为了降低索引大小, 在合并之后对每个点的邻居进行裁剪以降低每个节点的出度.

ACORN[112]是在 HNSW 基础上设计的支持同时处理向量相似性查询和结构化谓词过滤的索引方案. 它将每个节点的邻居候选点数量从 M 扩展到 λM, 以提高搜索到满足要求的点的可能性, 但是这会导致索引大小和构建时间的增加. 为了解决这个问题, 对扩展后的邻居列表进行剪枝操作, 保留 $M\beta$ 条边, 其余的边在搜索时通过两跳邻居获取, 以平衡索引大小和搜索效率. 实验结果显示, ACORN 方案相较于 Filtered-DiskANN 等设计方案取得了 2 倍以上的性能提升. UNG[151]是一个用于支持带标签混合搜索的信息框架. 它将数据根据标签进行分组, 然后通过有向无环图来构建标签导航图 LNG 以刻画标签的包含关系, 在此基础上利用当前的图构建方法对每一组数据分别构建相应的图, 最后通过跨组边来连接不同标签组的向量, 实现高效的搜索.

此外, HQANN[152]同样是一个基于图的方法, 在邻近图的构建阶段优先链接属性相同或相似的数据点以保持图的连通性. DEG[153]是一个面向多向量搜索的方法. 它同样是一个基于图的方案, 通过逐点插入的方式来构建索引, 并通过贪心帕累托 (greedy Pareto) 搜索方法来获取候选邻居集合, 采用动态剪枝策略来选择邻居节点. VBase[154]是一个微软开发的基于 PostgreSQL 的面向向量搜索的数据引擎. 它提供了用火山模型执行查询优化的后过滤搜索方案. AnalyticDB-V[155]是阿里巴巴开发的向量搜索引擎. 它能够根据查询优化器的代价估计选择合适搜索方案执行混合查询. Milvus[1]等向量数据库都提供包括前后过滤等一系列相关方案来支持混合搜索.

UNG 保证了在搜索过程中不会访问不能通过标签过滤的向量, 而 Filtered-DiskANN 与 ACORN 不具有该性质. 文献 [112] 中的 ACORN-1 索引构建速度较快, 而 ACORN-γ、UNG 与 Filtered-DiskANN 的构建速度相仿, 其中 UNG 略慢于其余二者. 在索引的内存占用方面, UNG 索引占用较少内存, Filtered-DiskANN 可以结合内存与磁盘存储索引以减少内存占用. 在更新方面, UNG 给出了一种局部重构的更新手段, 而其余方法并不能有效地支持动态更新, 针对标签过滤查询索引的高效更新策略仍然亟须进一步研究.

(2) 范围过滤

范围过滤指的是要求属性数据落在一个范围内, 整体的查询目标是找出属性值在指定范围内与查询向量近似的向量. SeRF[113]是一个面向范围过滤的方法, 对于半有界查询, 提出段图 (segment graph) 来优化查询, 通过利用图构建邻居节点的性质在单个 HNSW 索引中动态分段, 避免为每个范围单独构建图; 对于一般范围查询, 提出二维段图 (2D segment graph) 来加快查询, 并通过压缩 n 个段图来实现降低存储空间的目标. 它适用于指出动态范围过滤的向量检索, 如查询某个时间范围内具有显著特征的产品、车辆等. 实验结果显示其能够在保持较高召回率的同时具有很高的查询效率.

WST[156]是一个结合线段树和图来实现范围混合查询的方案. 它递归地划分数据集构建索引, 能够根据查询的范围特点来选择相应的查询算法. iRangeGraph[157]同样是一个基于线段树的方案. 它在线段树的每个节点上建立对应的图索引, 在搜索时先根据线段树来确定满足范围的点, 然后再进一步搜索返回满足要求的向量; 在构建阶段构建少量基础图, 在查询时动态组合这些图生成与查询目标相符合的图来完成搜索. 实验结果显示其比 Milvus 等方案在相同召回率下具有更快的查询速度. RangePQ[158]针对 iRangeGraph 难以支持更新和空间占用高的问题, 提出 PQ 量化和二叉树结合的方法来设计优化策略. DIGRA[159]通过基于动态多叉树的结构来设计支持动态更新.

SeRF、iRangeGraph、DIGRA 与 RangePQ 算法的各项均摊复杂度如表 9 所示. DIGRA 与 RangePQ 的索引构建速度相对较快. iRangeGraph 与 DIGRA 的索引内存占用相仿, SeRF 略小于二者, 由于 RangePQ 结合了量化技术, 其内存空间占用是 4 个方法中最小的. DIGRA 与 RangePQ 提供了对动态更新的支持, 二者各自的删除速度都明显快于插入速度; SeRF 与 iRangeGraph 当遇到数据更新时只能进行重构, 不能很好地支持数据频繁更新的场景. iRangeGraph 可拓展以支持多属性范围查询, 而其余 3 个方法并未直接提供对该场景的支持. 在查询性能方面, 由于 RangePQ 引入了量化技术, 其搜索精度会稍逊于其余 3 个方法, DIGRA 与 iRangeGraph 在高维数据集上搜索精

度相仿且都优于 SeRF, DIGRA 在低维度数据集 (比如 SIFT) 上相较于 iRangeGraph 表现更好.

表 9　4 个面向范围与向量的混合查询方法对比

具体方法	构建复杂度	空间复杂度	更新支持	单次更新时间复杂度	优点	缺点
SeRF	$O(nm^2\log n)$	$O(nm\log n)$	全局重构	N/A	空间占用较低	不支持局部更新
iRangeGraph	$O(nm+n\log n)$	$O(nm\log n)$	全局重构	N/A	搜索精度高、可拓展支持多属性范围过滤	不支持局部更新
DIGRA	$O(n\log n)$	$O(nm\log n)$	局部更新	$O(\log n)$	搜索精度高、更新效率高	空间占用较高
RangePQ	$O(n\log n)$	$O(n)$	局部更新	$O(\log n)$	空间占用低	搜索精度略低

混合查询受到了越来越广泛的关注. 它能够提升信息检索的精确度和灵活性, 满足复杂多样化的搜索需求, 提升用户体验. 如何支持高效的混合查询是向量数据库的关键问题之一, 探索更高效的搜索方法以及支持更多种类的混合查询具有重要意义, 未来需要进行更多相关研究.

3.8　理论分析

通过对近似最近邻算法进行理论分析, 可以帮助我们理解算法性能的边界, 指导算法的设计和优化以及揭示高维空间的性能, 避免按经验调参的盲目性, 为系统的扩展提供坚实基础. 近年来有不少工作在这方面取得了进展, 下面我们进行介绍.

首先是面向图构建的理论分析研究. 文献 [160] 研究了基于图的近似最近邻搜索的理论和时间, 重点分析了高维数据在稠密情况下 ($d << \log n$) 的问题. 在图的构建中, 会通过添加长边 (long-range link) 来加速搜索进程. 该文献理论分析了在图中添加长边对搜索的影响, 指出当前均匀地添加长边的策略优于随机添加. 该文献还对当前以 best-first-search 为基础的束搜索方案进行了理论分析, 指出当前的搜索策略有助于减少图的度数, 进而降低图的复杂度, 提升搜索性能. 当前理论依赖均匀分布假设, 未来需扩展至非均匀分布的情况. 这篇文献的理论分析为基于图的近似最近邻搜索提供了理论基础, 帮助我们理解图的构建和搜索过程中的关键因素. 文献 [114] 系统分析了主流的基于图的方案的在最坏情况下的性能表现和理论局限性. 它从慢处理和快处理两个方面来分析当前基于图的方案的性能表现. 慢处理是指在图中为每个节点添加邻居时考虑全局的数据点, 快处理是指先随机初始化图, 然后搜索部分数据点作为候选节点从中选择相应的点作为邻居节点. 分析结果指出, 除了 DiskANN 的慢处理方法能够实现多对数时间复杂度之外, 所有方法在构建的数据实例上都需要线性的时间复杂度来进行搜索, 但是 DiskANN 的慢处理方法需要 $O(n^3)$ 复杂度, 在实际中并不适用. 虽然算法的实际性能依赖数据分布和参数设置, 但是本文通过理论证明与系统性实验揭示了 ANNS 算法的实际适用边界, 为算法选择与参数调优提供了重要参考.

其次是查询难度分析的研究. LID (local intrinsic dimensionality)[161,162]是用来衡量数据集特征的重要工具. 它结合数据点的距离分布来获得分析结果. 可以用这个指标来判断数据集进行近似最近邻搜索的难易程度. 文献 [10] 给出了一些常用数据集的 LID 值和 RC (relative contrast)[163]值. RC 值是用平均距离和最近邻的距离的比值来衡量难易程度. 文献 [115] 指出通过 LID 可以衡量一个不同 KNN 查询的难度, 这为我们从理论上对 ANNS 搜索的性能分析带来了很多的启发和思路. 文献 [116] 提出了一种针对基于图的方法的查询难度度量方法 Steiner-hardness, 旨在解决现有方法在衡量图的查询复杂性时的局限性. 当前图方法在处理某些困难查询时难以获得很高的召回率, 导致整体性能下降, 传统的 LID 方法没有考虑图的拓扑结构信息, 难以反映查询的复杂性[116]. 文献 [116] 证明了 Steiner-hardness 和有向斯坦纳树 (directed Steiner tree, DST) 高度关联, 因此将问题简化为 DST 问题, 然后利用它的求解结果以有效地计算 Steiner-hardness. 实验结果显示, 与 LID 方法相比, Steiner-hardness 更能有效地反映查询实际代价.

3.9　小　结

面对高维向量数据高性能的要求以及复杂业务场景的挑战, 学界与业界推动了向量搜索算法在多个维度的持续创新与演进. 硬件加速技术 (如 SIMD 指令集、GPU 并行计算和多线程) 为向量搜索与索引构建提供了显著的性能优化空间. 通过并行计算架构和指令集层面的优化, 这些技术有效缓解了高维向量计算中的性能瓶颈. 现代

CPU 普遍支持的 SIMD 和多线程机制, 能够为向量搜索提供灵活高效的基础算力支撑; GPU 凭借其大规模并行架构和高带宽显存, 则特别适合批量向量运算, 但同时也面临数据结构适配性要求高、显存容量制约大规模数据处理以及高成本等挑战. 未来, 如何协同利用 CPU 与 GPU 的异构计算优势, 构建高效且可扩展的向量检索体系, 将成为向量数据库发展的关键方向.

在算法层面, 面向学习增强的优化引入机器学习模型对数据分布主动建模以适应不同数据集的特性, 优化索引划分以及搜索路由等, 并与倒排和图方法进行深度融合, 以提升搜索精度与效率. 面向距离比较操作 (DCO) 的优化方法的核心思想是通过在距离计算过程中尽早判断两个点的距离是否超过阈值, 提前终止距离计算以实现计算性能优化. 这些方法通过随机投影、主成分分析 (PCA)、残差角度估计等手段, 对两个点之间的距离进行估计, 结合假设检验等手段, 快速过滤掉不满足条件的点. 它们是在基于图、倒排等基本结构的基础上进行设计, 作为插件进行使用, 降低系统的改动和适配难度.

在应对大规模数据与资源受限场景方面, 磁盘-内存混合索引 (如 DiskANN、SPANN) 和分布式优化 (如 Pyramid、Auncel、SmartANNS) 提供了横向与纵向的可扩展性解决方案. 它们通过内存压缩、分布式分片、异构硬件协同等机制, 实现了单机与集群层面的高效处理能力. 磁盘-内存混合索引通常与数据访问优化 (如重排序、缓存) 协同设计, 以最大化 I/O 利用率; 分布式优化则结合算法特性与系统调度, 提升全局负载均衡与资源利用.

在人工智能广泛应用、业务需求多样化背景下, 混合查询优化 (如 Filtered-DiskANN、ACORN、SeRF 等) 实现了向量与结构化属性的联合检索, 满足复杂的业务逻辑和多维查询需求. 通过整合向量检索方法与经典数据结构来构建的混合查询索引, 按其核心优化目标, 主要分为面向关键词查询和面向范围查询两大类, 用户可以根据应用需求, 选择合适的索引完成高性能的查询.

在理论研究方面, 研究者通过建立严谨的数学模型, 对算法的复杂度边界和性能极限进行了系统的刻画. 这些工作不仅揭示了高维空间下 ANNS 算法的性能瓶颈, 还深入分析了如图结构构建、长边添加策略、搜索机制等关键环节对搜索效率的影响, 提出了如 LID、RC、Steiner-hardness 等度量方法, 用以量化数据集和查询任务的内在难度. 这些研究通过最坏情况分析与复杂度证明, 为算法设计和参数调优提供了坚实的理论基础, 减少了经验调参的盲目性, 并为系统的可扩展性和性能保证提供了科学指导.

4 未来研究展望

在人工智能时代, 向量数据库作为关键的数据管理的基础设施, 会发挥越来越重要的作用. 向量近似最近邻搜索作为其中的关键环节, 将在未来的研究中继续发挥重要作用. 通过以上的综述研究我们可以发现, 近年来向量近似最近邻检索取得了一系列的进展, 但面对以下 3 个趋势: (1) 非结构化数据快速增长, 如每天有超过 2 000 万条视频被上传到 YouTube 平台[164], 越来越多不同类型的如文本、图像、音频等数据会进行嵌入表示以实现深层语义的表达, 向量数据发挥着日益重要的作用; (2) 实时动态需求越来越重要, 每天有大量的视频、购物信息等新数据产生[67], 要求应用能够及时对新内容完成反馈; (3) 混合查询越来越重要, Miluvs 等向量数据库将混合查询作为向量数据库的一个重要功能[63]来实现结合向量的深层语义表示以及结构化数据的精确语义, 从而更准确地提供给用户满意的结果, 当前仍存在若干关键挑战亟待解决. 本节从 5 个方向来对未来的研究方向进行展望.

(1) 面向磁盘的索引结构优化

向量数据维度高、数据量大、内存占用高, 当前基于图的方法, 如 HNSW[28]、NSG[17]等严重依赖内存的方案难以应对大规模数据的存储和检索需求. 当前已有一些工作在这方面取得了一些进展, 如 SPANN[107]、DiskANN[36]等, 未来研究应致力于设计更高效的结构, 充分利用磁盘的存储能力, 设计磁盘内存结合的索引结构, 降低磁盘随机 I/O 的开销以保证搜索性能, 避免因磁盘带来的性能瓶颈, 并探索利用新硬件来提升性能等.

(2) 动态索引更新机制

向量数据的动态更新是一个重要的研究方向. 当前的索引结构大部分是静态的, 难以支持动态的数据更新操作, 尤其是基于图的方法. 当前在向量数据库中采用定期重建索引的方案来应对动态变化的数据, 导致开销巨大. 因此, 研究如何设计增量式的图更新策略, 在支持高效的增删操作的同时, 保证维持图的质量, 保持高召回率、低

延迟十分关键.

(3) 高效的混合搜索处理方案

支持向量和关键词等结构化数据的复杂查询是向量数据库中十分关键的任务, 在实际中难以用向量表示一切, 为了满足用户的需求, 向量数据库需要支持复杂的混合查询操作. 当前已有一些工作在这方面取得了进展, 如基于图的方法结合了关键词搜索等, 未来需要进一步设计高效的混合搜索处理方案, 包括扩展查询优化器的能力, 设计统一的查询处理框架以原生支持各类查询算子, 构建高效的复合索引结构, 开发基于代价估计模型的查询优化器以实现高效的执行计划等.

(4) 面向压缩数据的检索技术

通过压缩数据, 我们可以大幅度降低数据量, 从而突破内存的限制, 提升搜索性能. 当前已有一些工作在这方面取得了一些进展, 如基于量化的方法等. 未来需要进一步研究高效的压缩技术, 针对不同的相似性度量函数构建具有理论误差保证的量化方法, 以及根据数据的特点来设计高效的量化方法等. 进一步地, 如何在压缩数据的基础上设计高效的索引结构是需要继续研究的关键问题.

(5) 近似最近邻搜索的理论研究

通过理论分析能够指导算法的设计. 当前的理论研究主要集中在基于 LSH 的方法上, 对于很多方法, 尤其是基于图的方法缺乏足够的理论指导. 未来需要进一步研究不同计算模型下的近似最近邻搜索的理论, 精确刻画不同方法的性能表现, 研究动态更新下的理论, 帮助评估连续更新下的图的搜索稳定性, 研究不同查询实例的难度评估手段, 为设计稳定而高效的索引结构提供理论指导等.

5 总 结

向量近似最近邻搜索作为向量数据库的核心技术组件, 历经数十年发展已形成相对完善的方法体系, 相关研究在理论与应用层面均取得显著突破. 尽管已有若干优秀综述系统梳理了该领域的早期进展, 但近年来随着人工智能的发展, 向量检索得到更加广泛的应用, ANNS 技术呈现出新的发展态势. 本文基于前沿研究成果, 对当前 ANNS 技术进行系统性综述: 首先介绍了基本概念, 包含问题定义、相似度度量、数据类型等; 继而从 5 大类索引组织方法来剖析主流索引结构; 进而从硬件加速、学习增强方法、距离比较操作优化、磁盘内存混合场景、数据访问优化、分布式场景、混合查询场景和理论分析视角分类阐述最新的搜索优化成果; 最后对未来的研究方向进行探讨和展望. 本综述旨在为向量搜索和向量数据库的研究和设计提供参考和借鉴.

References

[1] Wang JG, Yi XM, Guo RT, Jin H, Xu P, Li SJ, Wang XY, Guo XZ, Li CM, Xu XH, Yu K, Yuan YX, Zou YH, Long JQ, Cai YD, Li ZX, Zhang ZF, Mo YH, Gu J, Jiang RY, Wei Y, Xie C. Milvus: A purpose-built vector data management system. In: Proc. of the 2021 Int'l Conf. on Management of Data. New York: ACM, 2021. 2614–2627. [doi: 10.1145/3448016.3457550]

[2] Guo RT, Luan XF, Xiang L, Yan X, Yi XM, Luo JG, Cheng QY, Xu WZ, Luo JR, Liu F, Cao ZS, Qiao YL, Wang T, Tang B, Xie C. Manu: A cloud native vector database management system. Proc. of the VLDB Endowment, 2022, 15(12): 3548–3561. [doi: 10.14778/3554821.3554843]

[3] Chen C, Jin CZ, Zhang YN, Podolsky S, Wu C, Wang SP, Hanson E, Sun Z, Walzer R, Wang JG. SingleStore-V: An integrated vector database system in SingleStore. Proc. of the VLDB Endowment, 2024, 17(12): 3772–3785. [doi: 10.14778/3685800.3685805]

[4] Pan JJ, Wang JG, Li GL. Survey of vector database management systems. The VLDB Journal, 2024, 33(5): 1591–1615. [doi: 10.1007/s00778-024-00864-x]

[5] Zhao WX, Zhou K, Li JY, Tang TY, Wang XL, Hou YP, Min YQ, Zhang BC, Zhang JJ, Dong ZC, Du YF, Yang C, Chen YS, Chen ZP, Jiang JH, Ren RY, Li YF, Tang XY, Liu ZK, Liu PY, Nie JY, Wen JR. A survey of large language models. arXiv:2303.18223, 2025.

[6] Liu ZY, Wang PJ, Song XB, Zhang X, Jiang BB. Survey on hallucinations in large language models. Ruan Jian Xue Bao/Journal of Software, 2025, 36(3): 1152–1185 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7242.htm [doi: 10.13328/j.cnki.jos.007242]

[7] Fan WQ, Ding YJ, Ning LB, Wang SJ, Li HY, Yin DW, Chua TS, Li Q. A survey on RAG meeting LLMs: Towards retrieval-

augmented large language models. In: Proc. of the 30th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Barcelona: ACM, 2024. 6491–6501. [doi: 10.1145/3637528.3671470]

[8] Zhang HL, Ji XD, Chen YL, Fu FC, Miao XP, Nie XN, Chen WP, Cui B. PQCache: Product quantization-based KVCache for long context LLM inference. Proc. of the ACM on Management of Data, 2025, 3(3): 201. [doi: 10.1145/3725338]

[9] Indyk P, Motwani R. Approximate nearest neighbors: Towards removing the of dimensionality. In: Proc. of the 30th Annual ACM Symp. on Theory of Computing. Dallas: ACM, 1998. 604–613. [doi: 10.1145/276698.276876]

[10] Li W, Zhang Y, Sun YF, Wang W, Li MJ, Zhang WJ, Lin XM. Approximate nearest neighbor search on high dimensional data—Experiments, analyses, and improvement. IEEE Trans. on Knowledge and Data Engineering, 2020, 32(8): 1475–1488. [doi: 10.1109/TKDE.2019.2909204]

[11] Wang MZ, Xu XL, Yue Q, Wang YX. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. Proc. of the VLDB Endowment, 2021, 14(11): 1964–1978. [doi: 10.14778/3476249.3476255]

[12] Azizi I, Echihabi K, Palpanas T. Graph-based vector search: An experimental evaluation of the state-of-the-art. Proc. of the ACM on Management of Data, 2025, 3(1): 43. [doi: 10.1145/3709693]

[13] Wang JD, Zhang T, Song JK, Sebe N, Shen HT. A survey on learning to hash. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2018, 40(4): 769–790. [doi: 10.1109/TPAMI.2017.2699960]

[14] Cai D. A revisit of hashing algorithms for approximate nearest neighbor search. IEEE Trans. on Knowledge and Data Engineering, 2021, 33(6): 2337–2348. [doi: 10.1109/TKDE.2019.2953897]

[15] Matsui Y, Uchida Y, Jégou H, Satoh S. A survey of product quantization. ITE Trans. on Media Technology and Applications, 2018, 6(1): 2–10. [doi: 10.3169/mta.6.2]

[16] Aumüller M, Bernhardsson E, Faithfull A. ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems, 2020, 87: 101374. [doi: 10.1016/j.is.2019.02.006]

[17] Fu C, Xiang C, Wang CX, Cai D. Fast approximate nearest neighbor search with the navigating spreading-out graph. Proc. of the VLDB Endowment, 2019, 12(5): 461–474. [doi: 10.14778/3303753.3303754]

[18] Reimers N, Gurevych I. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: ACL, 2019. 3982–3992. [doi: 10.18653/v1/D19-1410]

[19] OpenAI Platform. Vector embeddings. 2025. https://platform.openai.com/docs/guides/embeddings

[20] Pilehvar MT, Camacho-Collados J. Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning. Cham: Springer, 2021. 1–8. [doi: 10.1007/978-3-031-02177-0]

[21] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing & Management, 1988, 24(5): 513–523. [doi: 10.1016/0306-4573(88)90021-0]

[22] Lassance C, Clinchant S. An efficiency study for SPLADE models. In: Proc. of the 45th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Madrid: ACM, 2022. 2220–2226. [doi: 10.1145/3477495.3531833]

[23] Zhang XN, Liu XB, Song JK, Nie XS, Wang SH, Yin YL. Survey on hash learning for large-scale image retrieval. Ruan Jian Xue Bao/Journal of Software, 2025, 36(1): 79–106 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7141.htm [doi: 10.13328/j.cnki.jos.007141]

[24] Li TY, Yang XC, Ke YP, Wang B, Liu YN, Xu JX. Alleviating the inconsistency of multimodal data in cross-modal retrieval. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering. Utrecht: IEEE, 2024. 4643–4656. [doi: 10.1109/ICDE60146.2024.00353]

[25] Robertson SE, Walker S, Jones S, Hancock-Beaulieu MM, Gatford M. Okapi at TREC-3. In: Proc. of the 3rd NIST Text Retrieval Conf. (TREC3). Washington: NIST, 1996. 109–126.

[26] Bhattacharya A. Fundamentals of Database Indexing and Searching. New York: Chapman and Hall/CRC, 2014. [doi: 10.1201/b17767]

[27] Leskovec J, Rajaraman A, Ullman JD. Mining of Massive Datasets. 3rd ed., Cambridge: Cambridge University Press, 2020. 89–99.

[28] Malkov YA, Yashunin DA. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2020, 42(4): 824–836. [doi: 10.1109/TPAMI.2018.2889473]

[29] facebookresearch/faiss: A library for efficient similarity search and clustering of dense vectors. 2025. https://github.com/facebookresearch/faiss

[30] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975, 18(9): 509–517. [doi: 10.1145/361002.361007]

[31] Huang Q, Feng JL, Zhang YK, Fang Q, Ng W. Query-aware locality-sensitive hashing for approximate nearest neighbor search. Proc. of the VLDB Endowment, 2015, 9(1): 1–12. [doi: 10.14778/2850469.2850470]

[32] Qin JB, Wang YS, Xiao C, Wang W, Lin XM, Ishikawa Y. GPH: Similarity search in Hamming space. In: Proc. of the 34th IEEE Int'l Conf. on Data Engineering (ICDE). Paris: IEEE, 2018. 29–40. [doi: 10.1109/ICDE.2018.00013]

[33] Jégou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2011, 33(1): 117–128. [doi: 10.1109/TPAMI.2010.57]

[34] Gao JY, Long C. RaBitQ: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search. Proc. of the ACM on Management of Data, 2024, 2(3): 167. [doi: 10.1145/3654970]

[35] Aguerrebere C, Bhati IS, Hildebrand M, Tepper M, Willke T. Similarity search in the blink of an eye with compressed indices. Proc. of the VLDB Endowment, 2023, 16(11): 3433–3446. [doi: 10.14778/3611479.3611537]

[36] Subramanya SJ, Devvrit, Kadekodi R, Krishaswamy R, Simhadri HV. DiskANN: Fast accurate billion-point nearest neighbor search on a single node. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 1233.

[37] Wang JD, Li SP. Query-driven iterated neighborhood graph search for large scale indexing. In: Proc. of the 20th ACM Int'l Conf. on Multimedia. Nara: ACM, 2012. 179–188. [doi: 10.1145/2393347.2393378]

[38] Zhao X, Tian Y, Huang K, Zheng BL, Zhou XF. Towards efficient index construction and approximate nearest neighbor search in high-dimensional spaces. Proc. of the VLDB Endowment, 2023, 16(8): 1979–1991. [doi: 10.14778/3594512.3594527]

[39] Jégou H, Tavenard R, Douze M, Amsaleg L. Searching in one billion vectors: Re-rank with source coding. In: Proc. of the 2011 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP). Prague: IEEE, 2011. 861–864. [doi: 10.1109/ICASSP.2011.5946540]

[40] Sun YF, Wang W, Qin JB, Zhang Y, Lin XM. SRS: Solving *c*-approximate nearest neighbor queries in high dimensional Euclidean space with a tiny index. Proc. of the VLDB Endowment, 2014, 8(1): 1–12. [doi: 10.14778/2735461.2735462]

[41] Malkov Y, Ponomarenko A, Logvinov A, Krylov V. Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems, 2014, 45: 61–68. [doi: 10.1016/j.is.2013.10.006]

[42] Dong W, Moses C, Li K. Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proc. of the 20th Int'l Conf. on World Wide Web. Hyderabad: ACM, 2011. 577–586. [doi: 10.1145/1963405.1963487]

[43] Peng Y, Choi B, Chan TN, Yang JY, Xu JL. Efficient approximate nearest neighbor search in multi-dimensional databases. Proc. of the ACM on Management of Data, 2023, 1(1): 54. [doi: 10.1145/3588908]

[44] Harwood B, Drummond T. FANNG: Fast approximate nearest neighbour graphs. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE, 2016. 5713–5722. [doi: 10.1109/CVPR.2016.616]

[45] Fu C, Wang CX, Cai D. High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2022, 44(8): 4139–4150. [doi: 10.1109/TPAMI.2021.3067706]

[46] Zhang JR, Ma RH, Song T, Hua Y, Xue ZG, Guan CY, Guan HB. Hierarchical satellite system graph for approximate nearest neighbor search on big data. ACM/IMS Trans. on Data Science, 2021, 2(4): 32. [doi: 10.1145/3488377]

[47] Muñoz JV, Gonçalves MA, Dias Z, Torres RDS. Hierarchical clustering-based graphs for large scale approximate nearest neighbor search . Pattern Recognition, 2019, 96: 106970. [doi: 10.1016/j.patcog.2019.106970]

[48] Chen M, Zhang K, He ZY, Jing YN, Wang XS. RoarGraph: A projected bipartite graph for efficient cross-modal approximate nearest neighbor search. Proc. of the VLDB Endowment, 2024, 17(11): 2735–2749. [doi: 10.14778/3681954.3681959]

[49] Singh A, Subramanya SJ, Krishnaswamy R, Simhadri HV. FreshDiskANN: A fast and accurate graph-based ANN index for streaming similarity search. arXiv:2105.09613, 2021.

[50] Silpa-Anan C, Hartley R. Optimised KD-trees for fast image descriptor matching. In: Proc. of the 2008 IEEE Conf. on Computer Vision and Pattern Recognition. Anchorage: IEEE, 2008. 1–8. [doi: 10.1109/CVPR.2008.4587638]

[51] Guttman A. R-trees: A dynamic index structure for spatial searching. In: Proc. of the 1984 ACM SIGMOD Int'l Conf. on Management of Data. Boston: ACM, 1984. 47–57. [doi: 10.1145/602259.602266]

[52] Sproull RF. Refinements to nearest-neighbor searching ink-dimensional trees. Algorithmica, 1991, 6(1–6): 579–589. [doi: 10.1007/BF01759061]

[53] Dasgupta S, Freund Y. Random projection trees and low dimensional manifolds. In: Proc. of the 40th Annual ACM Symp. on Theory of Computing. Columbia: ACM, 2008. 537–546. [doi: 10.1145/1374376.1374452]

[54] Dasgupta S, Sinha K. Randomized partition trees for exact nearest neighbor search. In: Proc. of the 26th Annual Conf. on Learning Theory. 2013. 317–337.

[55] Ciaccia P, Patella M, Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In: Proc. of the 23rd Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers Inc., 1997. 426–435.

[56] Yianilos PN. Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proc. of the 4th Annual ACM-

SIAM Symp. on Discrete Algorithms. Austin: Society for Industrial and Applied Mathematics, 1993. 311–321.

[57] Tavallali P, Tavallali P, Singhal M. K-means tree: An optimal clustering tree for unsupervised learning. The Journal of Supercomputing, 2021, 77(5): 5239–5266. [doi: 10.1007/s11227-020-03436-2]

[58] Fukunaga K, Narendra PM. A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans. on Computers, 1975, C-24(7): 750–753. [doi: 10.1109/T-C.1975.224297]

[59] spotify/annoy. 2025. https://github.com/spotify/annoy

[60] FLANN—Fast library for approximate nearest neighbors. 2025. https://github.com/flann-lib/flann

[61] Ma HZ, Li JZ, Zhang Y. Reconsidering tree based methods for k-maximum inner-product search: The LRUS-covertree. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering. Utrecht: IEEE, 2024. 4671–4684. [doi: 10.1109/ICDE60146.2024.00355]

[62] Voronoi diagram. 2025. https://en.wikipedia.org/wiki/Voronoi_diagram

[63] Milvus. 2025. https://milvus.io/zh

[64] Xu Q, Yang J, Zhang F, Pan JD, Chen K, Shen YR, Zhou AC, Du XY. Tribase: A vector data query engine for reliable and lossless pruning compression using triangle inequalities. Proc. of the ACM on Management of Data, 2025, 3(1): 82. [doi: 10.1145/3709743]

[65] Wei JQ, Lee X, Liao ZY, Palpanas T, Peng BT. Subspace collision: An efficient and accurate framework for high-dimensional approximate nearest neighbor search. Proc. of the ACM on Management of Data, 2025, 3(1): 79. [doi: 10.1145/3709729]

[66] Bruch S, Nardini FM, Rulli C, Venturini R. Efficient inverted indexes for approximate retrieval over learned sparse representations. In: Proc. of the 47th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Washington: ACM, 2024. 152–162. [doi: 10.1145/3626772.3657769]

[67] Xu YM, Liang HY, Li J, Xu ST, Chen Q, Zhang QX, Li C, Yang ZY, Yang F, Yang YQ, Cheng P, Yang M. SPFresh: Incremental in-place update for billion-scale vector search. In: Proc. of the 29th Symp. on Operating Systems Principles. Koblenz: ACM, 2023. 545–561. [doi: 10.1145/3600006.3613166]

[68] Jafari O, Maurya P, Nagarkar P, Islam KM, Crushev C. A survey on locality sensitive hashing algorithms and their applications. arXiv:2102.08942, 2021.

[69] Datar M, Immorlica N, Indyk P, Mirrokni VS. Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. of the 20th Annual Symp. on Computational Geometry. New York: ACM, 2004. 253–262. [doi: 10.1145/997817.997857]

[70] Gan JH, Feng JL, Fang Q, Ng W. Locality-sensitive hashing scheme based on dynamic collision counting. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. Scottsdale: ACM, 2012. 541–552. [doi: 10.1145/2213836.2213898]

[71] Zhao X, Chen ZH, Huang K, Zhang RY, Zheng BL, Zhou XF. Efficient approximate maximum inner product search over sparse vectors. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering. Utrecht: IEEE, 2024. 3961–3974. [doi: 10.1109/ICDE60146.2024.00303]

[72] Zhao X, Zheng BL, Yi XM, Luan XF, Xie C, Zhou XF, Jensen CS. FARGO: Fast maximum inner product search via global multi-probing. Proc. of the VLDB Endowment, 2023, 16(5): 1100–1112. [doi: 10.14778/3579075.3579084]

[73] Norouzi M, Punjani A, Fleet DJ. Fast search in Hamming space with multi-index hashing. In: Proc. of the 2012 IEEE Conf. on Computer Vision and Pattern Recognition. Providence: IEEE, 2012. 3108–3115. [doi: 10.1109/CVPR.2012.6248043]

[74] Qin JB, Xiao C, Wang YS, Wang W, Lin XM, Ishikawa Y, Wang GR. Generalizing the pigeonhole principle for similarity search in Hamming space. IEEE Trans. on Knowledge and Data Engineering, 2021, 33(2): 489–505. [doi: 10.1109/TKDE.2019.2899597]

[75] Liu QY, Shen YY, Chen L. HAP: An efficient Hamming space index based on augmented pigeonhole principle. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 917–930. [doi: 10.1145/3514221.3517880]

[76] Ge TZ, He KM, Ke QF, Sun J. Optimized product quantization. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2014, 36(4): 744–755. [doi: 10.1109/TPAMI.2013.240]

[77] Norouzi M, Fleet DJ. Cartesian K-means. In: Proc. of the 2013 IEEE Conf. on Computer Vision and Pattern Recognition. Portland: IEEE, 2013. 3017–3024. [doi: 10.1109/CVPR.2013.388]

[78] Babenko A, Lempitsky V. Additive quantization for extreme vector compression. In: Proc. of the 2014 IEEE Conf. on Computer Vision and Pattern Recognition. Columbus: IEEE, 2014. 931–938. [doi: 10.1109/CVPR.2014.124]

[79] Martinez J, Clement J, Hoos HH, Little JJ. Revisiting additive quantization. In: Proc. of the 14th European Conf. on Computer Vision. Amsterdam: Springer, 2016. 137–153. [doi: 10.1007/978-3-319-46475-6_9]

[80] Zhang T, Du C, Wang JD. Composite quantization for approximate nearest neighbor search. In: Proc. of the 31st Int'l Conf. on Machine Learning. 2014. II-838–II-846.

[81] Wang JF, Wang JD, Song JK, Xu XS, Shen HT, Li SP. Optimized cartesian K-means. IEEE Trans. on Knowledge and Data Engineering, 2015, 27(1): 180–192. [doi: 10.1109/TKDE.2014.2324592]

[82] Babenko A, Lempitsky V. Tree quantization for large-scale similarity search and classification. In: Proc. of the 2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). Boston: IEEE, 2015. 4240–4248. [doi: 10.1109/CVPR.2015.7299052]

[83] Ning QQ, Zhu JK, Zhong ZY, Hoi SCH, Chen C. Scalable image retrieval by sparse product quantization. IEEE Trans. on Multimedia, 2017, 19(3): 586–597. [doi: 10.1109/TMM.2016.2625260]

[84] Heo JP, Lin Z, Yoon SE. Distance encoded product quantization for approximate K-nearest neighbor search in high-dimensional space. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2019, 41(9): 2084–2097. [doi: 10.1109/TPAMI.2018.2853161]

[85] Pan ZB, Wang LZ, Wang Y, Liu YC. Product quantization with dual codebooks for approximate nearest neighbor search. Neurocomputing, 2020, 401: 59–68. [doi: 10.1016/j.neucom.2020.03.016]

[86] Xu Z, Niu LS, Meng RM, Zhao LY, Ji JQ. Residual vector product quantization for approximate nearest neighbor search. In: Gama J, Li TR, Yu Y, Chen EH, Zheng Y, Teng F, eds. Advances in Knowledge Discovery and Data Mining (PAKDD 2022). Cham: Springer, 2022. 208–220. [doi: 10.1007/978-3-031-05933-9_17]

[87] Xu Z, Zhou MD, Liu YX, Zhao LY, Liu JJ. Approximate nearest neighbor search by cyclic hierarchical product quantization. Signal, Image and Video Processing, 2025, 19(6): 452. [doi: 10.1007/s11760-025-04030-w]

[88] André F, Kermarrec AM, Le Scouarnec N. Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan. Proc. of the VLDB Endowment, 2015, 9(4): 288–299. [doi: 10.14778/2856318.2856324]

[89] Veasey T. Understanding optimized scalar quantization. 2025. https://www.elastic.co/search-labs/blog/scalar-quantization-optimization

[90] Gao JY, Gou YT, Xu YX, Yang YY, C Long, Wong RCW. Practical and asymptotically optimal quantization of high-dimensional vectors in Euclidean space for approximate nearest neighbor search. Proc. of the ACM on Management of Data, 2025, 3(3): 202. [doi: 10.1145/3725413]

[91] Azizi I, Echihabi K, Palpanas T. ELPIS: Graph-based similarity search for scalable data science. Proc. of the VLDB Endowment, 2023, 16(6): 1548–1559. [doi: 10.14778/3583140.3583166]

[92] Echihabi K, Fatourou P, Zoumpatianos K, Palpanas T, Benbrahim H. Hercules against data series similarity search. Proc. of the VLDB Endowment, 2022, 15(10): 2005–2018. [doi: 10.14778/3547305.3547308]

[93] Wang Y, Wang P, Pei J, Wang W, Huang S. A data-adaptive and dynamic segmentation index for whole matching on time series. Proc. of the VLDB Endowment, 2013, 6(10): 793–804. [doi: 10.14778/2536206.2536208]

[94] Babenko A, Lempitsky V. The inverted multi-index. In: Proc. of the 2012 IEEE Conf. on Computer Vision and Pattern Recognition. Providence: IEEE, 2012. 3069–3076. [doi: 10.1109/CVPR.2012.6248038]

[95] Baranchuk D, Babenko A, Malkov Y. Revisiting the inverted indices for billion-scale approximate nearest neighbors. In: Proc. of the 15th European Conf. on Computer Vision. Munich: Springer, 2018. 209–224. [doi: 10.1007/978-3-030-01258-8_13]

[96] Lu KJ, Kudo M. R2LSH: A nearest neighbor search scheme based on two-dimensional projected spaces. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering (ICDE). Dallas: IEEE, 2020. 1045–1056. [doi: 10.1109/ICDE48307.2020.00095]

[97] Tao YF, Yi K, Sheng C, Kalnis P. Efficient and accurate nearest neighbor and closest pair search in high-dimensional space. ACM Trans. on Database Systems, 2010, 35(3): 20. [doi: 10.1145/1806907.1806912]

[98] Arora A, Sinha S, Kumar P, Bhattacharya A. HD-index: Pushing the scalability-accuracy boundary for approximate kNN search in high-dimensional spaces. Proc. of the VLDB Endowment, 2018, 11(8): 906–919. [doi: 10.14778/3204028.3204034]

[99] Butz AR. Alternative algorithm for Hilbert's space-filling curve. IEEE Trans. on Computers, 1971, C-20(4): 424–426. [doi: 10.1109/T-C.1971.223258]

[100] Zheng BL, Zhao X, Weng LG, Hung NQV, Liu H, Jensen CS. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search. Proc. of the VLDB Endowment, 2020, 13(5): 643–655. [doi: 10.14778/3377369.3377374]

[101] Skopal T, Pokorný J, Snášel V. Nearest neighbours search using the PM-tree. In: Proc. of the 10th Int'l Conf. on Database Systems for Advanced Applications. Beijing: Springer, 2005. 803–815. [doi: 10.1007/11408079_73]

[102] Wei JQ, Peng BT, Lee XD, Palpanas T. DET-LSH: A locality-sensitive hashing scheme with dynamic encoding tree for approximate nearest neighbor search. Proc. of the VLDB Endowment, 2024, 17(9): 2241–2254. [doi: 10.14778/3665844.3665854]

[103] Dong YH, Indyk P, Razenshteyn I, Wagner T. Learning space partitions for nearest neighbor search. In: Proc. of the 8th Int'l Conf. on Learning Representations. 2020.

[104] Gupta G, Medini T, Shrivastava A, Smola AJ. BLISS: A billion scale index using iterative re-partitioning. In: Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Washington: ACM, 2022. 486–495. [doi: 10.1145/3534678.3539414]

[105] Gao JY, Long C. High-dimensional approximate nearest neighbor search: With reliable and efficient distance comparison operations. Proc. of the ACM on Management of Data, 2023, 1(2): 137. [doi: 10.1145/3589282]

[106] Lu KJ, Xiao C, Ishikawa Y. Probabilistic routing for graph-based approximate nearest neighbor search. In: Proc. of the 41st Int'l Conf.

on Machine Learning. 2024. 33177–33195.

[107] Chen Q, Zhao B, Wang HD, Li MQ, Liu CJ, Li ZZ, Yang M, Wang JD. SPANN: Highly-efficient billion-scale approximate nearest neighbor search. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. New York: Curran Associates Inc., 2021. 398.

[108] Gou YT, Gao JY, Xu YX, Long C. SymphonyQG: Towards symphonious integration of quantization and graph for approximate nearest neighbor search. Proc. of the ACM on Management of Data, 2025, 3(1): 80. [doi: 10.1145/3709730]

[109] Deng SY, Yan X, Kelvin KWN, Jiang CY, Cheng J. Pyramid: A general framework for distributed similarity search on large-scale datasets. In: Proc. of the 2019 IEEE Int'l Conf. on Big Data. Angeles: IEEE, 2019. 1066–1071. [doi: 10.1109/BigData47090.2019.9006219]

[110] Zhang ZL, Jin C, Tang LP, Liu XZ, Jin X. Fast, approximate vector queries on very large unstructured datasets. In: Proc. of the 20th USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX, 2023. 995–1011.

[111] Gollapudi S, Karia N, Sivashankar V, Krishnaswamy R, Begwani N, Raz S, Lin YY, Zhang Y, Mahapatro N, Srinivasan P, Singh A, Simhadri HV. Filtered-DiskANN: Graph algorithms for approximate nearest neighbor search with filters. In: Proc. of the 2023 ACM Web Conf. Austin: ACM, 2023. 3406–3416. [doi: 10.1145/3543507.3583552]

[112] Patel L, Kraft P, Guestrin C, Zaharia M. ACORN: Performant and predicate-agnostic search over vector embeddings and structured data. Proc. of the ACM on Management of Data, 2024, 2(3): 120. [doi: 10.1145/3654923]

[113] Zuo CJ, Qiao M, Zhou WC, Li FF, Deng D. SeRF: Segment graph for range-filtering approximate nearest neighbor search. Proc. of the ACM on Management of Data, 2024, 2(1): 69. [doi: 10.1145/3639324]

[114] Indyk P, Xu HK. Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations. In: Proc. of the 37th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2023. 2891.

[115] Aumüller M, Ceccarello M. The role of local dimensionality measures in benchmarking nearest neighbor search. Information Systems, 2021, 101: 101807. [doi: 10.1016/j.is.2021.101807]

[116] Wang ZY, Wang QT, Cheng XX, Wang P, Palpanas T, Wang W. Steiner-hardness: A query hardness measure for graph-based ANN indexes. Proc. of the VLDB Endowment, 2024, 17(13): 4668–4682. [doi: 10.14778/3704965.3704974]

[117] Intel advanced vector extensions. 2025. https://www.intel.cn/content/www/cn/zh/support/articles/000005779/processors.html

[118] Introducing NEON Development Article. 2025. https://developer.arm.com/documentation/dht0002/a/Introducing-NEON

[119] Manohar MD, Shen ZQ, Blelloch G, Dhulipala L, Gu Y, Simhadri HV, Sun YH. ParlayANN: Scalable and deterministic parallel graph-based approximate nearest neighbor search algorithms. In: Proc. of the 29th ACM SIGPLAN Annual Symp. on Principles and Practice of Parallel Programming. Edinburgh: ACM, 2024. 270–285. [doi: 10.1145/3627535.3638475]

[120] Peng Z, Zhang MJ, Li K, Jin RM, Ren B. iQAN: Fast and accurate vector search with efficient intra-query parallelism on multi-core architectures. In: Proc. of the 28th ACM SIGPLAN Annual Symp. on Principles and Practice of Parallel Programming. Montreal: ACM, 2023. 313–328. [doi: 10.1145/3572848.3577527]

[121] Zhao WJ, Tan SL, Li P. SONG: Approximate nearest neighbor search on GPU. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering (ICDE). Dallas: IEEE, 2020. 1033–1044. [doi: 10.1109/ICDE48307.2020.00094]

[122] Yu YH, Wen D, Zhang Y, Qin L, Zhang WJ, Lin XM. GPU-accelerated proximity graph approximate nearest neighbor search and construction. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering (ICDE). Kuala Lumpur: IEEE, 2022. 552–564. [doi: 10.1109/ICDE53745.2022.00046]

[123] Zhou JB, Guo Q, Jagadish HV, Krcal L, Liu SY, Luan WH, Tung AKH, Yang YJ, Zheng YX. A generic inverted index framework for similarity search on the GPU. In: Proc. of the 34th IEEE Int'l Conf. on Data Engineering (ICDE). Paris: IEEE, 2018. 893–904. [doi: 10.1109/ICDE.2018.00085]

[124] Wieschollek P, Wang O, Sorkine-Hornung A, Lensch HPA. Efficient large-scale approximate nearest neighbor search on the GPU. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016. 2027–2035. [doi: 10.1109/CVPR.2016.223]

[125] Johnson J, Douze M, Jégou H. Billion-scale similarity search with GPUs. IEEE Trans. on Big Data, 2021, 7(3): 535–547. [doi: 10.1109/TBDATA.2019.2921572]

[126] Chen W, Chen JC, Zou FH, Li YF, Lu P, Zhao W. RobustiQ: A robust ANN search method for billion-scale similarity search on GPUs. In: Proc. of the 2019 Int'l Conf. on Multimedia Retrieval. Ottawa: ACM, 2019. 132–140. [doi: 10.1145/3323873.3325018]

[127] Groh F, Ruppert L, Wieschollek P, Lensch HPA. GGNN: Graph-based GPU nearest neighbor search. IEEE Trans. on Big Data, 2023, 9(1): 267–279. [doi: 10.1109/TBDATA.2022.3161156]

[128] Zhu YF, Ma RY, Zheng BH, Ke XY, Chen L, Gao YJ. GTS: GPU-based tree index for fast similarity search. Proc. of the ACM on Management of Data, 2024, 2(3): 142. [doi: 10.1145/3654945]

[129] Karthik V, Khan S, Singh S, Simhadri HV, Vedurada J. BANG: Billion-scale approximate nearest neighbor search using a single GPU. arXiv:2401.11324, 2024.

[130] Ootomo H, Naruse A, Nolet C, Wang R, Feher T, Wang Y. CAGRA: Highly parallel graph construction and approximate nearest neighbor search for GPUs. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering (ICDE). Utrecht: IEEE, 2024. 4236–4247. [doi: 10.1109/ICDE60146.2024.00323]

[131] Li WC, Feng C, Lian DF, Xie YX, Liu HF, Ge Y, Chen EH. Learning balanced tree indexes for large-scale vector retrieval. In: Proc. of the 29th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Long Beach: ACM, 2023. 1353–1362. [doi: 10.1145/3580305.3599406]

[132] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proc. of the 31st Int'l Conf. on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 6000–6010.

[133] Wang YF, Ma HD, Wang DZ. LIDER: An efficient high-dimensional learned index for large-scale dense passage retrieval. Proc. of the VLDB Endowment, 2022, 16(2): 154–166. [doi: 10.14778/3565816.3565819]

[134] Li CL, Zhang MJ, Andersen DG, He YX. Improving approximate nearest neighbor search through learned adaptive early termination. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 2539–2554. [doi: 10.1145/3318464.3380600]

[135] Zheng BL, Yue ZY, Hu Q, Yi XM, Luan XF, Xie C, Zhou XF, Jensen CS. Learned probing cardinality estimation for high-dimensional approximate NN search. In: Proc. of the 39th IEEE Int'l Conf. on Data Engineering (ICDE). Anaheim: IEEE, 2023. 3209–3221. [doi: 10.1109/ICDE55515.2023.00246]

[136] Yue Q, Xu XL, Wang YX, Tao YK, Luo XLY. Routing-guided learned product quantization for graph-based approximate nearest neighbor search. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering (ICDE). Utrecht: IEEE, 2024. 4870–4883. [doi: 10.1109/ICDE60146.2024.00370]

[137] Li MJ, Zhang Y, Sun YF, Wang W, Tsang IW, Lin XM. I/O efficient approximate nearest neighbour search based on learned functions. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering (ICDE). Dallas: IEEE, 2020. 289–300. [doi: 10.1109/ICDE48307.2020.00032]

[138] Tian B, Liu HK, Duan ZH, Liao XF, Jin H, Zhang Y. Scalable billion-point approximate nearest neighbor search using SmartSSDs. In: Proc. of the 2024 USENIX Annual Technical Conf. Santa Clara: USENIX Association, 2024. 69.

[139] Yang MY, Li WT, Jin JB, Zhong XY, Wang XY, Shen ZT, Jia W, Wang W. Effective and general distance computation for approximate nearest neighbor search. In: Proc. of the 41st IEEE Int'l Conf. on Data Engineering (ICDE). Hong Kong: IEEE, 2025. 1098–1110. [doi: 10.1109/ICDE65448.2025.00087]

[140] Deng LW, Chen PH, Zeng XM, Wang TF, Zhao Y, Zheng K. Efficient data-aware distance comparison operations for high-dimensional approximate nearest neighbor search. Proc. of the VLDB Endowment, 2024, 18(3): 812–821. [doi: 10.14778/3712221.3712244]

[141] Chen P, Chang WC, Jiang JY, Yu HF, Dhillon I, Hsieh CJ. FINGER: Fast inference for graph-based approximate nearest neighbor search. In: Proc. of the 2023 ACM Web Conf. Austin: ACM, 2023. 3225–3235. [doi: 10.1145/3543507.3583318]

[142] Wang MZ, Xu WZ, Yi XM, Wu SL, Peng ZY, Ke XY, Gao YJ, Xu XL, Guo RT, Xie C. Starling: An I/O-efficient disk-resident graph index framework for high-dimensional vector similarity search on data segment. Proc. of the ACM on Management of Data, 2024, 2(1): 14. [doi: 10.1145/3639269]

[143] Coleman B, Segarra S, Smola A, Shrivastava A. Graph reordering for cache-efficient near neighbor search. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 2789.

[144] Wang MZ, Wu HT, Ke XY, Gao YJ, Zhu YF, Zhou WC. Accelerating graph indexing for ANNS on modern CPUs. Proc. of the ACM on Management of Data, 2025, 3(3): 123. [doi: 10.1145/3725260]

[145] Kuffo L, Krippner E, Boncz P. PDX: A data layout for vector similarity search. Proc. of the ACM on Management of Data, 2025, 3(3): 196. [doi: 10.1145/3725333]

[146] Zhong XY, Li HT, Jin JB, Yang MY, Chu DM, Wang XY, Shen ZT, Jia W, Gu G, Xie Y, Lin XM, Shen HT, Song JK, Cheng P. VSAG: An optimized search framework for graph-based approximate nearest neighbor search. arXiv:2503.17911, 2025.

[147] Yahoojapan/NGT. 2025. https://github.com/yahoojapan/NGT

[148] Compute Express Link. 2025. https://en.wikipedia.org/wiki/Compute_Express_Link

[149] Jang J, Choi H, Bae H, Lee S, Kwon M, Jung M. CXL-ANNS: Software-hardware collaborative memory disaggregation and computation for billion-scale approximate nearest neighbor search. In: Proc. of the 2023 USENIX Annual Technical Conf. Boston: USENIX Association, 2023. 585–600.

[150] Jang J, Choi H, Bae H, Lee S, Kwon M, Jung M. Bridging software-hardware for CXL memory disaggregation in billion-scale nearest

neighbor search. ACM Trans. on Storage, 2024, 20(2): 10. [doi: 10.1145/3639471]

[151] Cai YZ, Shi JY, Chen YZ, Zheng WG. Navigating labels and vectors: A unified approach to filtered approximate nearest neighbor search. Proc. of the ACM on Management of Data, 2024, 2(6): 246. [doi: 10.1145/3698822]

[152] Wu W, He JL, Qiao Y, Fu GH, Liu L, Yu J. HQANN: Efficient and robust similarity search for hybrid queries with structured and unstructured constraints. In: Proc. of the 31st ACM Int'l Conf. on Information & Knowledge Management. Atlanta: ACM, 2022. 4580–4584. [doi: 10.1145/3511808.3557610]

[153] Yin ZQ, Gao JY, Balsebre P, Cong G, Long C. DEG: Efficient hybrid vector search using the dynamic edge navigation graph. Proc. of the ACM on Management of Data, 2025, 3(1): 29. [doi: 10.1145/3709679]

[154] Zhang QX, Xu ST, Chen Q, Sui GX, Xie JD, Cai ZZ, Chen YQ, He YX, Yang YQ, Yang F, Yang M, Zhou LD. VBase: Unifying online vector similarity search and relational queries via relaxed monotonicity. In: Proc. of the 17th USENIX Symp. on Operating Systems Design and Implementation. Boston: USENIX Association, 2023. 377–395.

[155] Wei CX, Wu B, Wang S, Lou RJ, Zhan CQ, Li FF, Cai YZ. AnalyticDB-V: A hybrid analytical engine towards query fusion for structured and unstructured data. Proc. of the VLDB Endowment, 2020, 13(12): 3152–3165. [doi: 10.14778/3415478.3415541]

[156] Engels J, Landrum B, Yu SD, Dhulipala L, Shun JL. Approximate nearest neighbor search with window filters. In: Proc. of the 41st Int'l Conf. on Machine Learning. 2024. 497.

[157] Xu YX, Gao JY, Gou YT, Long C, Jensen CS. iRangeGraph: Improvising range-dedicated graphs for range-filtering nearest neighbor search. Proc. of the ACM on Management of Data, 2024, 2(6): 239. [doi: 10.1145/3698814]

[158] Zhang FY, Jiang MX, Hou GH, Shi JM, Fan H, Zhou WC, Li FF, Wang SB. Efficient dynamic indexing for range filtered approximate nearest neighbor search. Proc. of the ACM on Management of Data, 2025, 3(3): 152. [doi: 10.1145/3725401]

[159] Jiang MX, Yang Z, Zhang FY, Hou GH, Shi JM, Zhou WC, Li FF, Wang SB. DIGRA: A dynamic graph indexing for approximate nearest neighbor search with range filter. Proc. of the ACM on Management of Data, 2025, 3(3): 148. [doi: 10.1145/3725399]

[160] Prokhorenkova L, Shekhovtsov A. Graph-based nearest neighbor search: From practice to theory. In: Proc. of the 37th Int'l Conf. on Machine Learning. 2020. 723.

[161] Houle ME. Dimensionality, discriminability, density and distance distributions. In: Proc. of the 13th IEEE Int'l Conf. on Data Mining Workshops. Dallas: IEEE, 2013. 468–473. [doi: 10.1109/ICDMW.2013.139]

[162] Amsaleg L, Chelly O, Furon T, Girard S, Houle ME, Kawarabayashi KI, Nett M. Estimating local intrinsic dimensionality. In: Proc. of the 21st ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Sydney: ACM, 2015. 29–38. [doi: 10.1145/2783258.2783405]

[163] He JF, Kumar S, Chang SF. On the difficulty of nearest neighbor search. In: Proc. of the 29th Int'l Conf. on Machine Learning. Edinburgh: Omnipress, 2012. 41–48.

[164] YouTube for Press. 2025. https://blog.youtube/press/

附中文参考文献

[6] 刘泽垣, 王鹏江, 宋晓斌, 张欣, 江奔奔. 大语言模型的幻觉问题研究综述. 软件学报, 2025, 36(3): 1152–1185. http://www.jos.org.cn/1000-9825/7242.htm [doi: 10.13328/j.cnki.jos.007242]

[23] 张雪凝, 刘兴波, 宋井宽, 聂秀山, 王少华, 尹义龙. 面向大规模图像检索的哈希学习综述. 软件学报, 2025, 36(1): 79–106. http://www.jos.org.cn/1000-9825/7141.htm [doi: 10.13328/j.cnki.jos.007141]

作者简介

宋子文, 博士生, CCF 学生会员, 主要研究领域为数据库系统, 大数据管理, 向量搜索.

王斌, 博士, 教授, 博士生导师, 主要研究领域为算法设计与分析, 流数据查询处理, 分布式系统.

张喜瑞, 硕士生, 主要研究领域为大数据管理, 向量数据库.

赵世豪, 硕士生, 主要研究领域为近似最近邻搜索, 向量数据库.

杨晓春, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为大数据管理与知识工程, 数据质量管理, 数据隐私保护与推荐系统.