边奥北，谭宗元，王洪亚. 基于维度划分的多近邻图相似搜索技术研究[J]. 智能计算机与应用，2026，16(2)：70-76. DOI：10.20169/j.issn.2095-2163.24040103

基于维度划分的多近邻图相似搜索技术研究

边奥北，谭宗元，王洪亚
（东华大学 计算机科学与技术学院，上海 201620）

摘 要：近似最近邻搜索在人工智能、推荐系统等领域应用广泛。基于近邻图的算法以其搜索速度快、搜索精度高的优势备受关注。但是通过实验发现，基于近邻图的算法在较多数据集上仍然存在严重的长尾查询问题。本文提出了维度划分多近邻图的索引构建方案解决长尾查询问题。实验表明，多索引近邻图算法在受长尾查询影响严重的数据集上相对于 HNSW 算法最高有 10X 以上的时间性能提升。

关键词：近似最近邻搜索；长尾查询；多近邻图；维度划分；HNSW 索引
中图分类号：TP399 **文献标志码**：A **文章编号**：2095-2163(2026)02-0070-07

Research on multi-nearest neighbor graph similarity search based on dimension division

BIAN Aobei, TAN Zongyuan, WANG Hongya
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

Abstract: Approximate nearest neighbor search is widely used in artificial intelligence, recommendation systems, and other fields. Algorithms based on nearest neighbor graphs are highly regarded for their fast search speed and high search accuracy. However, experiments have shown that these algorithms still suffer from serious long-tail query problems on many datasets. This paper proposes a solution to the long-tail query problem by introducing a dimension-divided multi-nearest neighbor graph indexing scheme. Experiments demonstrate that the multi-index nearest neighbor graph algorithm achieves a performance improvement of more than 10 times compared to the HNSW algorithm on datasets severely affected by long-tail queries.

Key words: approximate nearest neighbor search; long-tail query; multi-index nearest neighbor graph; dimension-divided; HNSW index

0 引 言

当前，大数据已经成为推动社会经济发展和科学进步的重要驱动力之一。据艾瑞咨询研究院在《中国数智融合发展洞察》报告[1]中表示，全球数据量以 59%以上的年增率快速增长，其中 80%是非结构化和半结构化数据。最近邻搜索在非结构化的数据检索方面有着重要的作用。

同时，大语言模型已经被广泛应用于生产中，但是目前的大模型都是预训练模型，对于训练截止日之后发生的事情一无所知。这会导致大模型不能利用实时的数据，以及缺乏私域数据或者企业数据。

经过系统研究后可知，可以利用向量数据库的记忆功能，从向量数据库中快速地加载和查询需要的数据，这些数据可以作为大模型的外部知识输入，帮助大模型生成更加准确、包含更多私域知识的答案。

在最近邻搜索方向上，基于图的最近邻搜索算法由于其较高的搜索性能成为该领域的领先范式，现已提出多种基于图的最近邻搜索算法。然而，却同样面临着搜索难题、即长尾查询。本文主要针对长尾查询，给出一种新的图索引构建方案、就是基于维度划分的多近邻图。该近邻图索引在受长尾查询影响较为严重的数据集上最高有 10X 的性能提升。

作者简介：边奥北(1997—)，男，硕士研究生，主要研究方向：近似最近邻搜索。Email：b13592278067@163.com；谭宗元(1990—)，男，博士研究生，主要研究方向：近似最近邻搜索；王洪亚(1976—)，男，教授，主要研究方向：近似最近邻搜索。

收稿日期：2024-04-01

1　相关工作

1.1　相关定义

最近邻搜索依赖于测量向量之间的距离，常用的距离度量包括欧氏距离、余弦相似度、汉明距离和曼哈顿距离，本文使用欧氏距离作为度量标准。

定义 1　欧式距离　在 n 维空间中，给定数据集中任意一条向量 $x=(x_1,x_2,\cdots,x_n)$ 和目标向量 $y=(y_1,y_2,\cdots,y_n)$，向量 x 和向量 y 间的距离为：

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{1}$$

欧式距离的值越小，表示两向量越近；反之，说明两向量距离越远。

定义 2　K 近邻　在 n 空间中，给定包含 N 条向量的数据集合 $B=\{b_1,b_2,\cdots,b_N\}$ 和目标数据 t，并确定返回的最近邻的个数，即 K 值，则近似最近邻搜索根据距离函数 $d(t,b)$ 返回目标数据 t 在数据集合 B 中的 K 个最近邻，即：

$$\text{Top } K=\underset{B'\in B,\,|B'|=K}{\operatorname{argmin}}\sum_{b\in B'}d(t,b) \tag{2}$$

1.2　近似最近邻搜索

最近邻搜索是一种常见的数据检索技术，用于查找数据集中与给定查询点最接近的数据点，广泛应用于机器学习、数据挖掘、图像处理等领域。随着人工智能技术的普及和应用的发展，基于向量表示的最近邻搜索方法也得到了学界的高度关注，许多公司和研究机构都在进行最近邻搜索相关技术的研究和应用，例如百度、腾讯、阿里等公司在推荐系统、搜索引擎等领域应用最近邻搜索技术。最近邻搜索主要有 4 个研究方向，分别是基于空间划分的算法、基于哈希的算法、基于向量量化的算法、基于近邻图的算法。

基于空间划分的算法将数据空间按照维度划分为多个子空间，然后根据查询点所在的位置选择相应的子空间搜索，从而减少搜索的时间复杂度，代表性的研究有 KD 树[2]、R 树[3]等。基于哈希的算法将数据集中的点通过哈希函数计算出哈希值存放到哈希表中，查询时比较查询点的哈希值与哈希表中存储的哈希值找到该查询点的最近邻，代表性的算法有 LSH[4]、MinHash[5]、multi_LSH[6]、VHP[7]。基于向量量化的算法通过将原始向量映射到离散的质心向量来减少存储空间，解决数据量较大的问题，代表性算法有 PQ[8]、RQ[9]等。

基于近邻图的算法将所有结点构建成一个近邻图，近邻图中每个点的邻居为该结点在数据集中的 k 个最近邻或者是近似最近邻结点。如 kNN 图，需要通过暴力搜索的方式为每个结点找到该点的 k 个邻居，因此索引构建时间较长。近似 kNN 图中保留其 k 个近似最近邻，如 EFANNA[10]、KNNG[11]、KGraph[12]等算法。另外，还有像 HNSW[13]、NSG[14]、NGT[15]、Vamana[16]等加入了裁边策略的图索引结构，控制结点邻居之间的角度，裁掉冗余邻居，减少搜索中每一跳访问点的数量，该类算法有较大的性能、存储优势。在搜索方面，几乎所有基于近邻图的近似最近邻搜索算法都是基于贪婪路由策略结合最佳优先搜索，从起始点出发，访问该点的所有邻居，然后选择距离最近的邻居作为下一跳，访问该点的所有邻居，依次迭代直到搜索跳数达到预先设置的搜索跳数为止，搜索跳数越多，召回率越高，但是搜索时间也会越长。基于近邻图的最近邻搜索算法中主要关注的衡量指标为搜索速度、搜索精度、索引构建时间、索引占用内存大小。每个结点的邻居由该节点的近邻点组成。搜索时从近邻图的起始点出发，执行基于贪心和宽度优先搜索的搜索算法遍历路径上的结点寻找查询点的最近邻。该类算法相对于其他算法在搜索性能上有较大优势。

HNSW[13]是指分层可导航小世界，该算法拥有较好的搜索性能、较低的索引构建时间、较小的索引空间占用，已经被应用在 Faiss[17]、Elasticsearch[18]等多个应用中。该算法主要思想是将图中的最大出度固定为一个常数，防止数据量增大时导致点的出度较大，影响搜索性能。同时建立分层机制，通过分层的方法将长边保留到上层，通过长边将查询快速导航到其最近邻所在区域。HNSW[13]构建方法如下：

首先，根据计算出每个点所在的层次 $l(0\leqslant l\leqslant \max L)$，0 层为最底层，$\max L$ 层为最高层，层与层之间的元素为包含关系，即 0 层包含所有元素，1 层包含 1 到 $\max L$ 中所有的元素，以此类推。然后，将每个结点增量插入到图中，在从 $\max L$ 到 $\min(L+1,\max L)$ 层逐层执行贪心搜索，找到进入 l 层的初始点，然后将该点插入到 l 层，直到将该点插入 0 层为止。

2　多索引近邻图

2.1　基于近邻图算法分析

近些年来，已经相继提出了多种基于图的最近邻搜索算法。然而，却面临着同样的搜索难题、即长尾查询。在大多数数据集（例如：Sift1M[19]、Gist[19]、

Glove[20]、Trevi[21]等,数据集有关信息将在第 3 节做详细介绍)上基于图的最近邻搜索算法 Recall@ 1 = 0.9 时仅仅需要 ms 级的时间,但是想要到达更高的召回率,尤其是 Recall@ 1 = 1 时将导致成百上千倍的时间消耗,或者根本无法达到更高的召回率。接下来研究以 HNSW[13]为例分析长尾查询在各个数据集的影响。

表 1 展示了数据集 Sift1M[19]、Gist[19]、Glove[20]、ImageNet 分别在 Recall@ 1 为 0.90、0.99、1.00 时单个查询所需要花费的时间,"-"表示在有限的搜索时间上该数据集在 HNSW[13]上没有达到对应的召回率。可以发现,这几个数据集都能够迅速达到 0.90 的召回率,然而需要花费数十倍时间才能到达 0.99。但是,由于目前基于图的算法使用的贪心搜索算法中针对所有查询点的搜索跳数是固定值,并没有有效地识别长尾查询的适应性算法。因此,只能通过增加所有查询的搜索跳数,扩大搜索范围的方式提升图召回率,对图性能造成严重的性能损失。

表 1 各数据集上长尾查询统计

Table 1 Long-tail query statistics on various datasets ms

数据集	Recall@ 1 = 0.90	Recall@ 1 = 0.99	Recall@ 1 = 1.00
Sift1M[19]	0.150	0.650	5.50
Gist[19]	2.480	32.200	-
Glove[20]	6.408	42.444	-
ImageNet	4.130	-	-

针对长尾查询问题,本文多次随机构建近邻图索引,统计每一个索引上的长尾查询,发现不同近邻图上对应的长尾查询并不相同。利用不同近邻图长尾查询不同的特性,提出多近邻图索引的构建方案。在多近邻图索引中,一个查询成为长尾查询需要同时满足该查询在多个近邻图上是长尾。该方案大大降低该查询点成为长尾查询的概率,提高了图索引的搜索效率。

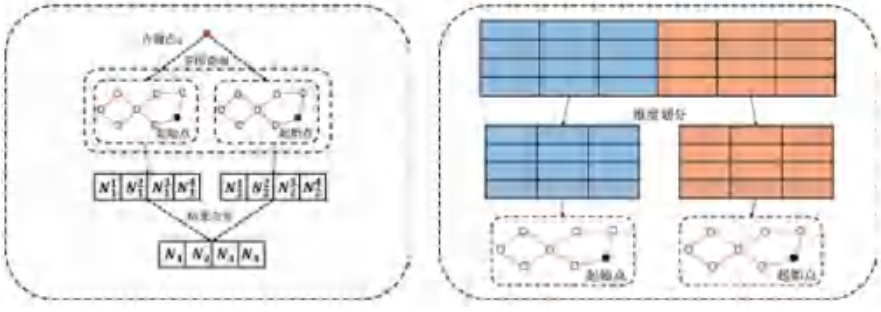
2.2 多近邻图索引的构建与搜索

使用全部数据多次构建随机近邻图索引结构,由多个子图共同组成多近邻图索引结构,如图 1(a)中 2 个近邻图结构。搜索方式见图 1(a),分别在 2 个近邻图中搜索查询点 q, 然后将两近邻图的搜索结果合并,按照距离从小到大排序, 得到 q 的近似最近邻点。

显然,多近邻图索引在索引构建方面存在 2 个严重的问题:

(1)多近邻图索引的构建时间也将会成倍增加,索引构建时间是衡量相似最近邻搜索算法好坏的一个重要的指标。像上文中这样简单的将数据构建成多个近邻图索引,多近邻图索引的构建时间将是单索引结构的数倍,本文在 2.3 节中提出维度划分的方案解决多近邻图构建时间长的问题。

(2)多近邻图索引组成索引结构内存空间占用较大。存储多个近邻图的内存空间代价同样是单个图索引的数倍空间开销,这种方案不利于大规模数据场景的部署,同样是不能接受的。本文在 2.4 节中提出空间合并的策略解决多近邻图结构内存消耗大的问题。



(a) 多索引近邻图的搜索过程 (b) 维度划分的多近邻图构建方式

图 1 多索引搜索过程及划分多索引构建过程展示

Fig. 1 Demonstration of multi-index search process and partitioning in multi-index construction

2.3 维度划分解决多近邻图索引构建时间长的问题

图索引的构建时间主要花费在结点之间的距离计算上,而距离计算的时间消耗取决于向量维度,维度越高,距离计算代价越大。因此,本文提出了基于向量维度划分的优化策略,解决近邻图索引构建时间长的问题。首先,将向量集合 S 按照维度切分,原始数据中向量的维度为 d,对于 $s \in S$,按照维度切分后得到 l 个向量 $S_1, S_2, S_3, \cdots, S_{l-2}, S_{l-1}, S_l$ 切片,每个切片中包含 n 个向量,每个分片的维度分别为 $d_1, d_2, d_3, \cdots, d_{l-2}, d_{l-1}, d_l$。 然后,将切分后的各个切片分别构建近邻图 $G_1, G_2, G_3, \cdots, G_{l-2}, G_{l-1}, G_l$。 特别地,由图 1(b)看到,以四维向量数据集为例,将四维

向量数据按照维度切分为 2 个分片，每个分片的维度为 2，为各分片分别构成近邻图结构。这样就会使构建多近邻图的距离计算时间之和等价于单独构建一个全维度的距离计算时间，相对于 2.2 节中的多近邻图构建方案，维度划分策略降低多近邻图索引构建时间，但是基于维度划分的多近邻图构建时间仍高于 HNSW 算法构建时间。

多索引近邻图与 HNSW 构图时间对比见表 2。由表 2 可知，在 Trevi[21] 数据集上，维度划分方法的索引构建时间与单个 HNSW[13] 索引构建时间基本相同，而在其他数据集上维度划分方法索引构建时间比单个 HNSW[13] 索引构建时间最多高出 57%。经分析是因为在索引构建过程中主要的构建时间不仅包含点与点之间的距离计算，还有其他的代码执行时间消耗，例如堆排序等过程。而在维度较高的数据上，由于距离计算的时间远高于其他的时间消耗，因此在 Trevi[21] 上，多索引与单索引时间接近。而维度较低的数据集上，如 Glove[20] 上多近邻图索引时间消耗高于单索引，但是相比于不使用划分时的 2 倍构建时间代价，向量维度划分方案有效地降低了多近邻图索引的构建时间。

表 2　多索引近邻图与 HNSW 构图时间对比

Table 2　Comparison of construction time between multi－index nearest neighbor graphs and HNSW

数据集	多索引构建时间/ms	HNSW 构建时间/ms	多索引构建时间 / HNSW 构建时间
Gist[19]	63 630 + 63 621	89 174	1.43
Sift1M[19]	16 576 + 16 677	24 638	1.35
Trevi[21]	9 035 + 9 092	17 789	1.01
Glove[20]	22 246 + 22 111	28 286	1.57

向量维度划分方法中，分片个数对多近邻图索引有重要的影响。本文通过实验发现，在构建多个相同索引情况下，随着索引数的数量增加，长尾查询的数量会逐渐减少。搜索跳数等于 100 时，在 Sift1M[19] 数据集上构建双索引下有 153 个长尾查询，而在索引个数增加到 6 个时，只有 2 个长尾查询。多近邻图索引的长尾查询交集个数会随着索引个数的增加而降低，但是带来如下问题：

(1)多近邻图索引占用空间代价增加。

(2)索引构建时间受构建过程堆排序等代码运行时间影响较大。

(3)索引复杂性提高。即使有较好的搜索性能，由于其较大的空间占用以及构建时间的增加，也不能显著增加多近邻图的数目。

因此，在本文中多近邻图索引结构推荐索引个数为 2 个。切分维度选择上，本文采用平均分配的方式、即 2 个切片各自占用一半维度，使得切分之后的向量之间最大程度保留近邻信息。

2.4　空间合并策略解决内存消耗问题

图算法为了提高搜索效率，需要将原始向量数据以及图索引结构依次添加到内存中，需要使用大量的内存空间。对于多索引结构的存储而言，相对于单图结构会使用更多的内存空间。同时，研究发现在较多数据集上有大量的数据点的邻居并没有被全部占用，造成了极大的空间浪费。在 Sift1M[19]、Gist[19]、Glove[20]、Trevi[21] 等多个数据集上经实验发现，当最大邻居数为 32 时，索引的平均出度最大只有 16.139，最小仅有 6.261，邻居空间全部占用节点所占比例最大仅有 1.9%。

基于该观察，可以将多索引合并，共享邻居空间。具体做法如图 2 所示(以双索引为例)：将向量数据按照 2.3 节中的切分方案得到 S_1，S_2，分别使用 HNSW[13] 算法构建索引结构 G_1，G_2。基于此，将 2 个子图合并到一个结构中，每个点的邻居空间中，从左到右存放 G_1 的邻居集，从右到左存放 G_2 邻居集，两者通过-1 分割，以 0 号结点为例，首先将 G_1 中 1、3 结点插入，以-1 为分隔符，然后将 G_2 中 0 号结点的 2 个邻居 2、5 插入。当查询点 q 搜索时，首先将 q 按照相同方法切分为 q_1，q_2；然后，q_1，q_2 分别访问对应的邻居空间，例如遍历到结点 3 时，q_1 从左向右访问 5、0、2 结点，q_2 访问 4 结点。最后，将搜索的最近邻结果合并。

图 2　双索引合并示意图

Fig. 2　Merge diagram of dual indexes

空间合并的时间代价相对于索引的构建时间是可以忽略不计的。索引合并只需要遍历一次近邻图索引即可，时间复杂度为 $O(m \times N)$，这里 N 为结点总数，m 为近邻图的最大出度，较小的固定值远小于 N。而索引构建时间复杂度为 $O(N \times \log^2 N)$。空间代价相对于多近邻图结构降低 37.5%，但是索引占用内存空间仍比 HNSW[13] 高出 25%，详细分析见第 3 节实验部分。

3 实验

本文的仿真实验中使用的服务器硬件配置为：CPU 型号 Intel(R)Xeon(R)Silver4110CPU@2.10 GHz，内存大小 64 GB，服务器系统 Ubuntu 18.04.6 LTS，本文中实验使用的算法均基于 Faiss-1.5.0 实现，索引构建使用 16 线程，性能对比使用单线程。在维度划分的多近邻图上，单个索引的最大出度 m 设置为 16，最大出度越大、该图的质量越好，连通性越强，但是会造成较大的空间占用，经测试 m = 16 时即可达到较好的搜索效果；efconstruction 表示索引构建过程中搜索候选点时的搜索跳数，该参数越大表示搜索到的候选邻居为其最近邻的概率越大，但是会造成构图时间较长，本文将 efconstruction 均设置为 50，在该参数下多近邻图索引算法即可发挥较好性能，防止较大时造成其构图时间过长。合并索引的最大出度设置为 20，防止空间合并时邻居空间不足；HNSW[13] 算法与多近邻图算法参数设置相似：m = 16，efconstruction = 50。在 4 个公开数据集上展开实验，数据集信息具体见表 3。

表 3 数据集

Table 3 Datasets

数据集	维度	数据集大小	查询点个数
Gist[19]	960	1 000 000	1 000
Sift1M[19]	128	1 000 000	10 000
Glove[20]	100	1 192 514	200
Trevi[21]	4 096	99 900	200

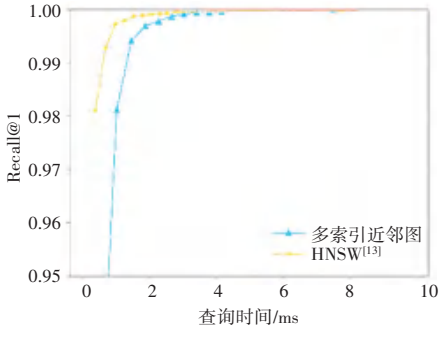
Gist[19] 是一个图像数据集，包含约 100 万个数据点，共 960 个维度。SIFT[19] 数据集包含的是图像的局部特征，就是在空间中对图片的位置、尺度、旋转固定量等进行提取获得特征的组合，数据向量的维度为 128。Glove[20] 包含 1 192 514 个从 Tweets 中提取的 100 维单词特征向量。Trevi[21] 由 400 000×1 024 位图(.bmp)图像组成，每张图像包含一个 16×16 的图像补丁数组。每个补丁被采样为 64×64 灰度，具有规范的尺度和方向。因此，Trevi[21] 补丁数据集由大约 10 万个 4 096 维向量组成。

算法评价指标采用召回率。对于给定的一个查询点 q，通过近似最近邻搜索方法得到该查询点 q 的 k 个最近邻向量组成的集合 S'，而该查询点真实的 k 个最近邻向量的集合是 S，查询召回率定义公式如下：

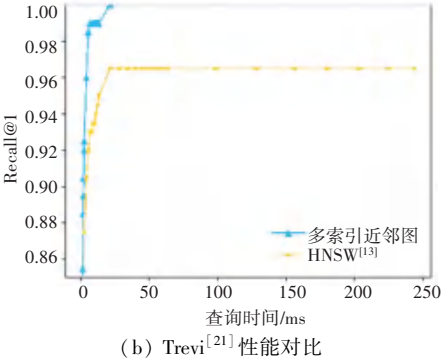
$$\mathrm{Recall}(q) = \frac{|S' \cap S|}{|S'|} = \frac{|S' \cap S|}{k} \tag{3}$$

在相同的召回率下，平均查询时间越短，证明算法的性能越好。

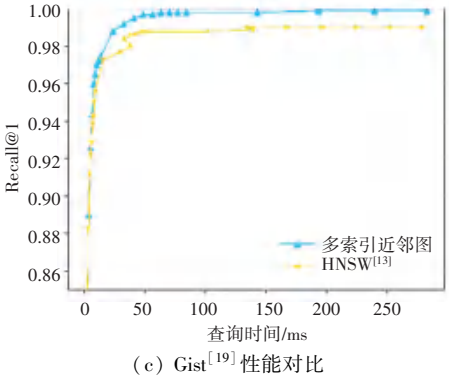
在以上 4 个数据集上将维度划分的多近邻图方法与 HNSW 方法进行比较，结果如图 3 所示。



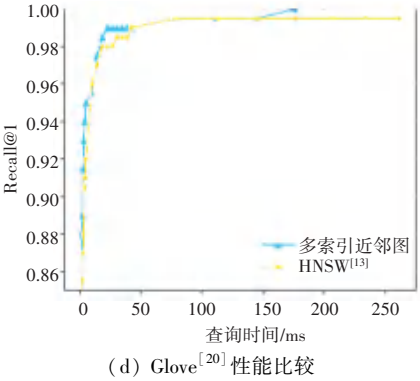
(a) Sift1M[19] 性能对比



(b) Trevi[21] 性能对比



(c) Gist[19] 性能对比



(d) Glove[20] 性能比较

图 3 性能对比

Fig. 3 Performance comparison

在 Gist[19]、Trevi[21]、Glove[20] 上，基于维度划分

的多近邻图算法在较低 Recall 时与 HNSW[13] 方法相近，而 Gist[19] 方法在 Recall@1=0.96 以上时性能高于 HNSW[13] 方法，Recall@1=0.99 时，维度划分的多近邻图方法比 HNSW[13] 方法快 5 倍，并且该方法可以达到更高的召回率。在 Trevi[21] 数据集上，HNSW[13] 算法上能达到的最高召回率为 0.965，查询时间为 243.704 4 ms，而划分方法能够到达的最高召回率为 1，查询时间为 21.550 ms。在 Glove[19] 数据集上，较低召回率时维度划分的多近邻图方法与 HNSW[13] 算法性能相近，在 Recall@1 为 1 时有较大优势。而在 Sift1M[19] 上基于维度划分的多近邻图方法与 HNSW[13] 算法的性能相近，主要是因为 Sift1M[19] 数据集使用基于图的最近邻搜索算法构建索引，并没有严重的长尾查询问题，仅在 10 ms 即可到达 Recall@1 为 1。

在索引构建时间上，由于该实验与 2.3 节中使用的参数相同，因此索引构建时间相同。将向量按照维度划分能够有效地解决像 2.2 节中那样简单构建多个近邻图引起的构图时间成倍增加的问题，但是依然没有解决构建多近邻图时堆排序等过程对构建时间的影响，因此在维度较低的数据集上该方法的索引构建时间比 HNSW[13] 算法最多高 57%左右，而在维度较高的数据上，例如 Trevi[21] 上构建时间能与 HNSW[13] 算法相近。

在内存方面，本文在构建多近邻图时，单个索引使用的最大邻居个数均为 $m = 16$，但是利用索引合并技术，合并后的索引最大邻居个数控制在 $m = 20$。因此，索引的空间代价相比于 HNSW[13] 算法高出了 25%。

4　结束语

基于近邻图的最近邻搜索算法在大量数据集上受长尾查询的严重影响，本文提出维度划分的多近邻图的最近邻搜索算法，由于近邻图索引构建存在随机性，构建多个近邻图索引上对应的长尾查询并不相同。本文在多个数据集上与目前性能最好的基于近邻图的最近邻搜索算法 HNSW[13] 在性能、索引构建时间、索引占用内存空间等方面做了详细的比较。结果显示该方法能够有效地解决长尾查询的问题，多近邻图算法在受长尾查询影响严重的数据集上相对于 HNSW 算法最高有 10X 以上的时间性能提升。通过维度划分、索引空间合并降低了 2.2 节中的多近邻图索引构建时间、索引空间占用。但是，相对于 HNSW 算法，在维度较低的数据集上索引构建时间最多高出 57%左右，维度较高数据集上仅高出 1%。索引占用空间相对于 HNSW 高出 25%。

参考文献

[1] 艾瑞咨询. 云原生技术发展趋势与生态分析报告[EB/OL]. (2022-07-01). https://report.iresearch.cn/report_pdf.aspx?id=4027.

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

[3] DASGUPTA S, FREUND Y. Random projection trees and low dimensional manifolds[C]// Proceedings of the 40th Annual ACM Symposium on Theory of Computing. New York: ACM, 2008: 537-546.

[4] GIONIS A, INDYK P, MOTWANI R. Similarity search in high dimensions via hashing[C]// Proceedings of the 25th International Conference on Very Large Data Bases(VLDB '99). New York: ACM, 1999: 518-529.

[5] BRODER A Z, CHARIKAR M, FRIEZE A M, et al. Min-wise independent permutations [C]//Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. New York: ACM, 1998: 327-336.

[6] LV Q, JOSEPHSON W, WANG Z, et al. Multi-probe LSH: Efficient indexing for high-dimensional similarity search[C]// Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna, Austria: VLDB Endowment, 2007: 950-961.

[7] LU Kejing, WANG Hongya, WANG Wei, et al. VHP: Approximate nearest neighbor search via virtual hypersphere partitioning[J]. Proceedings of the VLDB Endowment, 2020, 13(9): 1443-1455.

[8] JEGOU H, DOUZE M, SCHMID C. Product quantization for nearest neighbor search[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, 33(1): 117-128.

[9] CHEN Y, GUAN T, WANG C. Approximate nearest neighbor search by residual vector quantization[J]. Sensors, 2010, 10(12): 11259-11273.

[10] FU Cong, CAI Deng. EFANNA: An extremely fast approximate nearest neighbor search algorithm based on knn graph[J]. arXiv preprint arXiv, 1609.07228, 2016.

[11] HARWOOD B, DRUMMOND T. FANNG: Fast approximate nearest neighbour graphs [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2016: 5713-5722.

[12] KALANTIDIS Y, AVRITHIS Y. Locally optimized product quantization for approximate nearest neighbor search [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ : IEEE, 2014: 2321-2328.

[13] MALKOV Y A, YASHUNIN D A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 42(4): 824-836.

[14] FU Cong, XIANG Chao, WANG Changxu, et al. Fast approximate nearest neighbor search with the navigating spreading-out graph[J]. arXiv preprint arXiv, 1707.00143, 2017.

[15] IWASAKI M. Neighborhood graph and tree for indexing high

dimensional data[J]. Yahoo Japan Corporation, 2015, 22: 2020.
[16] JAYARAM S S, DEVVRIT F, SIMHADRI H V, et al. Diskann: Fast accurate billion-point nearest neighbor search on a single node [C]//Proceedings of the 33rd International Conference on Neural Information Processing Systems. New York: ACM, 2019: 1233.
[17] SHAY B. Elasticsearch is a distributed search and analytics engine optimized for speed and relevance on production-scale workloads [EB/OL]. (2024-04-01). https://github.com/elastic/elasticsearch.
[18] Facebook Research. Faiss: A library for efficient similarity search. [EB/OL]. (2024-01-01). https://github.com/facebookresearch/faiss.
[19] LAURENT A, HERVÉ J. Datasets for approximate nearest neighbor search [EB/OL]. (2024-04-01). http://corpus-texmex.irisa.fr.
[20] PENNINGTON J, SOCHER R, MANNING C D. Glove: Global vectors for word representation [C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). ACL, 2014: 1532-1543.
[21] WINDER S A J, BROWN M. Learning local image descriptors [C]//Proceedings of 2007 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2007: 1-8.