基于最优排序的局部敏感哈希索引

冯小康[1] 彭延国[1] 崔江涛[1] 刘英帆[2] 李 辉[3),4)]

[1)](西安电子科技大学计算机科学与技术学院 西安 710071)

[2)](香港中文大学系统工程与工程管理系 香港 999077)

[3)](西安电子科技大学网络与信息安全学院 西安 710071)

[4)](社会安全风险感知与防控大数据应用国家工程实验室 北京 100084)

摘 要 针对外存环境中海量高维数据近似最近邻(Approximate Nearest Neighbor,ANN)查询面临的“维度灾难”和 I/O 性能瓶颈难题,本文提出了一种基于最优排序的局部敏感哈希(Locality-Sensitive Hashing,LSH)索引方案 O2LSH(Optimal Order LSH).通过引入空间填充曲线为复合哈希键值建立线序并排序,使近邻候选点更多地分布在相同或相邻磁盘页面,实现用少量顺序 I/O 加载到足够多的候选点.本文对多种常用空间曲线技术进行了量化分析,发现:(1)基本排序方案 SK-LSH 使用的 row-wise 曲线具有“维度优先遍历”的特性,容易对 ANN 查询造成多种局限;(2)另一类“邻域优先遍历”特性的曲线能够产生更好的候选点局部分布,且排序性能更加稳定.通过对比,我们选取了一种最优的“邻域优先遍历”曲线构造线序,该线序能够最大程度地改善近邻候选点的局部分布,进一步提升磁盘访问效率和查询精度.在多个真实多媒体数据集上进行了对比实验,证实了 O2LSH 相对于先进 LSH 方案(包括 C2LSH、SK-LSH、SRS 以及 QALSH)在查询精度和 I/O 效率上的优越性.特别地,O2LSH 克服了基本排序方案 SK-LSH 对 LSH 关键参数的敏感性,算法实用性进一步提升.

关键词 近似最近邻;高维索引;局部敏感哈希;空间线序;局部分布

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2020.00930

Locality Sensitive Hashing Index Based on Optimal Linear Order

FENG Xiao-Kang[1] PENG Yan-Guo[1] CUI Jiang-Tao[1] LIU Ying-Fan[2] LI Hui[3),4)]

[1)](*School of Computer Science and Technology, Xidian University, Xi'an* 710071)

[2)](*Department of System Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong, China* 999077)

[3)](*School of Cyber Engineering, Xidian University, Xi'an* 710071)

[4)](*National Engineering Laboratory for Public Security Risk Perception and Control by Big Data, Beijing* 100084)

Abstract Nearest neighbor (NN) search in high-dimensional space is a fundamental paradigm in a wide range of applications, such as text information retrieval, search engine, content-based information query, duplication detection, etc. In these areas, data are usually large-scale and are modeled as high-dimensional features, which introduce two major problems to NN search, the “curse of dimensionality” and I/O performance bottleneck. On the one hand, the dimensionality curse makes exact NNs nearly infeasible to achieve. A lot of research efforts have been devoted to finding approximate nearest neighbors (ANN) which are close enough to the query to achieve a satisfying trade-off between accuracy and efficiency. On the other hand, large-scale feature sets

收稿日期:2018-07-16;在线出版日期:2019-04-29.本课题得到国家自然科学基金(61976168,61702403,61672408,61972309)、国家 111 计划(B16037)、社会安全风险感知与防控大数据应用国家工程实验室主任基金项目、CCF-华为数据库创新研究计划项目(CCF-HuaweiDBIR008B)、中国博士后科学基金(2018M633473)、陕西省自然科学基本研究计划(2015JQ6227,2018JM6073,2019CGXNG-023)、江西省重点研发计划(20181ACE50029)、中央高校基本科研业务基金(XJS190305,JB181505)、西安电子科技大学研究生创新基金资助.**冯小康**,博士研究生,中国计算机学会(CCF)会员,主要研究方向为高维索引与近似查询.E-mail:research@xkfeng.com.**彭延国**,博士,讲师,中国计算机学会(CCF)高级会员,主要研究方向为安全查询、隐私保护、云计算安全.**崔江涛**(通信作者),博士,教授,中国计算机学会理事,杰出会员,主要研究领域为数据与知识工程、大规模数据管理、数据安全与隐私保护.E-mail:cuijt@xidian.edu.cn.**刘英帆**,博士研究生,中国计算机学会(CCF)会员,主要研究方向为大规模复杂数据管理、高维数据相似性搜索.**李 辉**,博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为社会计算、知识发现、图挖掘和大数据中的隐私保护.

are often too massive to fit into the internal memory, external storage (usually disk) becomes a reasonable choice. However, due to the huge speed gap between internal memory and external memory, the resulting input/output communication becomes very expensive, too much NN candidates or improper loading manner would make the NN candidates loading the most time-consuming part of the entire NN search, called the I/O performance bottleneck. LSH is a widely adopted technique due to its excellent error guarantee and high computational efficiency in tackling the "curse of dimensionality". It can enable fast and accurate irrelevant points filtration and offers c-ANN results at a sub-linear time complexity which also makes it an attractive approach for disk-based ANN search. Plenty of LSH based methods have been developed to further boost the ANN performance. However, most of them access candidate objects through significant random I/O operations, which makes them tend to incur the I/O performance bottleneck. In this work, a novel Locality-Sensitive Hashing (LSH) index called Optimal Order LSH (O2LSH) is designed to further address the above two problems. First, O2LSH introduces space-filling curves to sort the compound LSH keys and rearrange the original data points accordingly. In this way, NN candidates can be stored on same or adjacent disk pages so that only a few sequential I/O operations can load enough candidates during the search. A thorough quantitative analysis is then conducted on several common space-filling curves. The results show that there exists two different kinds of characteristic among these space-filling curves, (1) curves of "dimension-first-traverse" characteristic (such as the row-wise curve) tend to introduce several limitations to ANN search; (2) curves of "neighborhood-first-traverse" characteristic (such as Z-order, Gray curve, Hilbert curve, etc.) can produce better local distribution of NN candidate points, and the performance are more stable. Based on the analysis, O2LSH chooses a best "neighborhood-first-traverse" curve as the linear order to maintain as many as NN candidates within same local disk pages. In this way, O2LSH can not only enhance the I/O efficiency but also improve the ANN search accuracy. We conduct empirical experiments on 6 real-world multimedia data sets. The results demonstrate the superior accuracy and I/O efficiency of O2LSH in ANN search, compared with 4 state-of-the-art methods, including C2LSH, SK-LSH, SRS and QALSH. Particularly, O2LSH is no longer sensitive to a key parameter of LSH function as the basic sorting-based solution does, which further improves the algorithm practicality.

Keywords　approximate nearest neighbor; high-dimensional index; locality-sensitive hashing; linear order; local distribution

1　引　言

最近邻(Nearest Neighbor,NN)查询广泛应用于文本信息检索、搜索引擎、基于图像内容的信息查询、数据重复性检测等领域[1].在这些领域,数据的海量、高维性质给快速、有效的最近邻查询提出了巨大的挑战.一方面,高维空间中的"维数灾难"难以克服,越来越多的研究者开始关注近似最近邻(Approximate Nearest Neighbor,ANN)查询算法,即在允许的范围内通过寻求近似解换取查询速度的提升;另一方面,海量数据集一般体量巨大,难以放入内存,外存存储成为更加合理的选择.但是内外存间存在巨大的速度差距,使得二者之间的通信(即I/O)变得十分昂贵.如果通信次数过多,或者外存访问方式不合理,会使得这一环节成为整个近邻查询最耗时的部分,我们称之为I/O性能瓶颈[2].

现有外存ANN查询方案主要包括局部敏感哈希(Locality Sensitive Hashing,LSH)[3-5]系列技术、乘积量化(Product Quantization,PQ)[6-7]技术以及近邻图(Nearest Neighbor Graph,NNG)[8-10]技术等.其中,LSH由于具有出色的误差保证和较高的

计算效率受到越来越多的关注，是目前解决 ANN 查询使用最为广泛的技术. LSH 是一种特殊的哈希函数，具有距离保持的特性，能够将原始空间中邻近的点以较大的概率映射到相同的哈希桶(称二者发生碰撞). 在实际查询时，通过构建复合 LSH 函数(Compound LSH Function)以及建立多索引哈希表能够提升相似点的碰撞概率和不相似点不碰撞的概率，从而减少近邻点的误报(False Positive，FP)和漏报(False Negative，FN)现象. 目前先进的外存 LSH 方案主要有 LSB-Forest[11]、C2LSH[12]、SRS[13]以及 QALSH[14]等. 其查询算法一般分为两个阶段：(1)先在内存中通过快速过滤无关数据确立近邻候选点；(2)从外存读入候选点原始数据，计算真实距离进而精炼出最近邻结果. 一个共同问题是，这些方案确立的候选点大都随机分布在磁盘上. 而为了保证查询精度，一般需要从外存中加载足够多候选点，这就导致这些方案的候选点加载会耗费大量昂贵的随机 I/O 操作，成为整个近邻查询的速度瓶颈.

为改善 ANN 查询的 I/O 效率，有研究者提出一种基于排序的 LSH 索引方案 SK-LSH[15]. 核心思想是，利用 row-wise 空间填充曲线(简称空间曲线)为复合哈希键值构造一种空间线序(简称线序)并对其排序. 原始数据按照新的顺序在磁盘上重新排列可以让更多近邻候选点存放在相同或者相邻磁盘页面上，使得近邻候选点加载更多地通过顺序 I/O 完成. 而且，由于一个磁盘页面通常能容纳多个候选点，单次 I/O 因此可以加载到更多候选点，整体 I/O 次数也被降低.

然而，基本的排序方案没有深入考察 row-wise 曲线下的近邻候选点局部分布质量，在实际 ANN 查询时存在三方面局限：(1) row-wise 线序在局部排列时易产生较多近邻点漏报，降低了查询精度；(2)线序的局部排列效果对 LSH 关键参数(主要是哈希函数桶宽 W)变化敏感，依赖较大的桶宽，而且需要大量的调参才能确定最优参数值，影响算法实用性；(3)较大的桶宽会引入更多近邻干扰点，需要更多哈希函数才能充分过滤，增加了存储和计算开销. 实际应用中存在多种空间曲线技术[16]，这些曲线具有不同的特性，可能对近邻候选点的局部分布产生不同影响. 由于候选点的局部分布质量是决定排序方案近邻查询性能的关键，因此，本文围绕不同空间曲线特性对近邻候选点局部分布的影响展开深入研究，期望进一步提升 ANN 查询性能. 本文主要贡献如下：

(1)对四种典型空间曲线下的近邻分布质量开展量化分析和对比，发现基本方案中所使用的曲线具有一种“维度优先遍历”的特性，这是造成其局限的主要原因；

(2)通过对比，发现“邻域优先遍历”特性的曲线能产生更好的近邻分布且性能更加稳定. 通过比较，选取一种排序效果最优的“邻域优先遍历”曲线并设计了一种新的 LSH 索引方案(Optimal Order LSH，O2LSH)，能够最大程度地改善近邻候选点的局部分布，进一步提升 ANN 查询性能；

(3)建立索引结构并设计相应的 ANN 查询算法，在多个真实数据集上进行了对比实验，证实了 O2LSH 相对于其他先进 LSH 方案(包括 C2LSH、SK-LSH、SRS 和 QALSH)在 ANN 查询性能上的优越性. 并且新方案克服了基本方案的几个局限，不再对哈希关键参数敏感，算法实用性得以提升.

本文第 2 节介绍近邻查询和 LSH 相关基本概念；第 3 节分析现有外存索引研究现状，并对不同曲线下的近邻分布质量展开研究，寻找最优线序，进一步改善 I/O 性能；第 4 节给出完整的索引构建和查询算法；第 5 节进行实验对比；第 6 节进行总结.

2 相关工作

给定一个高维特征向量集 $\boldsymbol{D}\subset R^d$，共包含 n 个 d 维空间的特征向量(或特征点)，令 $\|,\|$ 代表两个点间的欧式距离. 对于查询点 $\boldsymbol{q}\in R^d$，ANN 查询是指从 $\boldsymbol{D}$ 中搜索到一点 $\boldsymbol{x}$，使得 $\|\boldsymbol{q},\boldsymbol{x}\|\leq c\|\boldsymbol{q},\boldsymbol{x}^*\|$. 其中 $\boldsymbol{x}^*$ 是 $\boldsymbol{q}$ 在 $\boldsymbol{D}$ 中的真实最近邻，$c>1$ 是近似查询的比率.

2.1 局部敏感哈希(LSH)

LSH 技术最早由 Indyk 和 Motwani 等人提出并应用于 ANN 查询[3-5]，基本定义如下：

定义 1. LSH 函数. 令 $B(\boldsymbol{q},r)$ 表示 d 维空间中以 $\boldsymbol{q}$ 为中心点，半径为 r 的超球. 给定近似比率 c 和两个概率值 p_1、p_2($p_1>p_2$)，对于 R^d 空间中任意两点 $\boldsymbol{x}$、$\boldsymbol{q}$，如果函数族 $\boldsymbol{H}=\{h:R^d\rightarrow Z\}$ 满足如下两个条件，则称 $\boldsymbol{H}$ 是 (r,cr,p_1,p_2)-敏感的：

(1)若 $\boldsymbol{x}\in B(\boldsymbol{q},r)$，则 $\Pr[h(\boldsymbol{q})=h(\boldsymbol{x})]\geq p_1$；

(2)若 $\boldsymbol{x}\notin B(\boldsymbol{q},cr)$，则 $\Pr[h(\boldsymbol{q})=h(\boldsymbol{x})]\leq p_2$；

式(1)是一种欧式空间中常用的 LSH 函数：

$$h(\boldsymbol{x})=\frac{\lfloor \boldsymbol{a}\cdot\boldsymbol{x}+b\rfloor}{W} \tag{1}$$

首先随机生成一个 d 维向量 $\boldsymbol{a}$，其每个分量独立服从标准正态分布 $N(0,1)$，将 $\boldsymbol{a}$ 所在直线等分成宽度为 W 的区段. 对于数据集 $\boldsymbol{D}$ 中任意一点 $\boldsymbol{x}$，将其往 $\boldsymbol{a}$ 上投影，并沿着 $\boldsymbol{a}$ 的方向偏移 b 的距离，投影点最终落入区段的编号即作为 $\boldsymbol{x}$ 最终的哈希值 $h(\boldsymbol{x})$. 其中，b 是一个随机数，服从 $[0,W)$ 上的均匀分布.

对于式(1)，相距 s 的两个点 $\boldsymbol{x}_1,\boldsymbol{x}_2$ 发生碰撞(落入相同区段)的概率可用如下公式计算：

$$\begin{aligned}p(s)&=\Pr[h(\boldsymbol{x}_1)=h(\boldsymbol{x}_2)]\\&=\int_0^W\frac{1}{s}f_2\left(\frac{t}{s}\right)\left(1-\frac{t}{W}\right)\mathrm{d}t\end{aligned} \tag{2}$$

其中 $f_2(x)=\dfrac{2}{\sqrt{2\pi}}\mathrm{e}^{-\frac{x^2}{2}}$. 该公式表明，桶宽 W 固定时，$\boldsymbol{x}_1,\boldsymbol{x}_2$ 发生碰撞的概率随二者之间距离 s 的增加而单调递减. 因此称函数 $h(\boldsymbol{x})$ 是 (r,cr,p_1,p_2)-敏感的哈希函数，这里 $p_1=p(r)$，$p_2=p(cr)$.

2.2　基于 LSH 的近邻候选点鉴别机制

LSH 函数具有距离保持的特性，能够让原始空间中邻近的点以较大的概率碰撞，原始距离越大，碰撞概率越小. 在单个 LSH 函数上，取与查询点 $\boldsymbol{q}$ 落入相同区段的点作为近邻候选点，可以成功过滤掉很多无关数据点. 但一些距离 $\boldsymbol{q}$ 较远的点也能与之碰撞，这些点被称作假阳性点(False Positive，FP). FP 点会降低查询的准确率. 为提升过滤能力，通常从 $\boldsymbol{H}$ 中随机生成 m 个 LSH 函数组合成一个复合哈希函数 $G(\cdot)=(h_1(\cdot),h_2(\cdot),\cdots,h_m(\cdot))$ 进行联合过滤. 对于数据点 $\boldsymbol{x}$，$G(\boldsymbol{x})=(h_1(\boldsymbol{x}),h_2(\boldsymbol{x}),\cdots,h_m(\boldsymbol{x}))$ 被称作 $\boldsymbol{x}$ 的复合哈希键值. 如果 $\boldsymbol{x}\in B(\boldsymbol{q},r)$，则 $\Pr[G(\boldsymbol{x})=G(\boldsymbol{q})]\geq p_1^m=P_1$，如果 $\boldsymbol{x}\notin B(\boldsymbol{q},cr)$，则 $\Pr[G(\boldsymbol{x})=G(\boldsymbol{q})]\leq p_2^m=P_2$. 因此 $G(\cdot)$ 是 (r,cr,P_1,P_2)-敏感的.

复合哈希函数能有效解决假阳性问题，但也会产生另外一个问题，假阴性现象：随着哈希函数的增多，一些原本较近的点可能在部分哈希函数上无法发生碰撞而被错误地排除掉，这些点称为假阴性点(False Negative，FN). FN 点增多会降低近邻的召回率，同样会降低查询精度. 这些点解决办法是，建立 L 组复合哈希函数，多个复合函数间用“或”的关系，即数据点只要在一个复合函数上发生碰撞就被认作候选点. 通常，数据集连同其在一组复合哈希函数上的键值集合构成一个哈希表，多组复合哈希函数意味着多个哈希表的存储，如果所需构建的哈希表过多，则空间开销会显著增大.

基于以上近邻候选点机制发展出了很多 ANN 查询方案，查询精度和效率不断提升. 不过，早期的 LSH 方案[4,17-18]主要面向内存，且需要构建大量哈希表，这在实际海量高维数据集上并不实际. LSB-Forest 是第一个面向外存的 LSH 索引结构，通过将复合哈希键值转化为一个二进制的 z-order 值，实现在一个 B 树索引上适应多种搜索半径，极大地降低了索引规模. C2LSH 和 QALSH 采用动态碰撞计数的方法筛选候选点，根据两个点复合哈希键值间的碰撞次数来衡量它们距离的远近，取得了更高的查询精度. SRS 致力于建立 tiny 级的索引结构，借助原始距离与投影空间距离的对应关系，实现在少量哈希函数上进行候选点的初步筛选.

3　基于空间曲线进一步改善 I/O 性能

3.1　现有外存索引 I/O 性能

现有外存 LSH 索引技术在解决 ANN 查询问题时有着各自的优势，但存在一个共同问题：I/O 效率较低. 主要原因是，这些方法在确立候选点之后选择直接将候选点从外存中读入以进行精炼. 这种情况下，候选点一般都是随机分布在磁盘上的，如图 1(a). 这就意味着每一个候选点的加载平均会消耗一次昂贵的随机 I/O 操作. 又因为在过滤环节无法精确地确立候选点，为了保证近邻查询的精度，大多数情况下都需要加载足够多的候选点. 最终造成这类方法在候选点加载环节十分耗时，容易形成速度瓶颈.

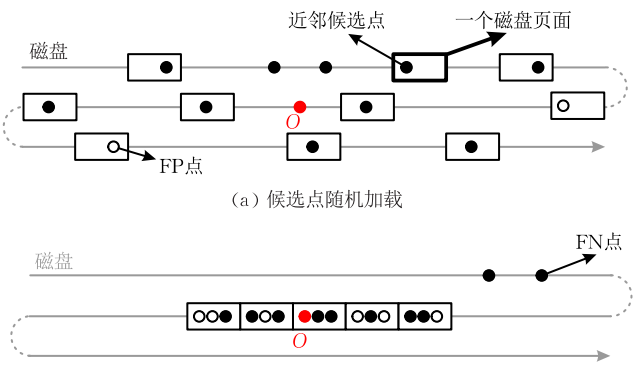


图 1　现有外存索引中候选点分布对比

为了克服这一问题，需要进一步研究改善 ANN 查询算法的 I/O 性能. 由于 I/O 操作主要发生在候

选点加载之时. 相应地,改善 I/O 性能的途径主要有两种:(1)降低 I/O 次数,主要通过提升过滤能力以减少所要加载的候选点来实现;(2)优化 I/O 方式,减少昂贵的随机 I/O 访问,更多地使用顺序 I/O,并且提升单次 I/O 操作的候选点加载量. 对于第一种,Tang 等人提出了一种缓存候选点紧凑编码的技术[19]. 在候选点加载前利用紧凑编码进行上下界判断,能够较早地将一些无关点排除. 不过本质上其候选点仍旧是随机加载,存在加载效率低的问题.

Liu 等人提出的基于排序的 LSH 索引方案 SK-LSH[15],借助空间填充曲线技术实现候选点的局部分布,可以同时实现以上两个目标. 如图 1(b),数据按照空间线序重新排列之后,邻近的点会更多地排列在连续的磁盘页面,这样就可以采用连续的 I/O 来读取. 并且单次的磁盘访问平均能够加载到更多最近邻候选点,从而在保证查询精度的前提下,降低磁盘访问次数.

3.2 空间曲线及其特性

空间填充曲线是一种比较成熟的空间降维技术,能够将多维空间映射到一维[16,20]. 其填充过程为:将多维空间划分成超立方体网格,按照某种规则不重复地遍历所有网格并递增地为所遍历网格分配一个独特的编号,编号递增的顺序就是曲线为多维空间规定的一维空间线序. 空间曲线的一个特性是,曲线上编号邻近的网格在实际空间中往往也是邻近的,如果数据依照空间线序在磁盘上存放,就能实现让很多邻近点在磁盘局部聚集存放的目标.

空间曲线的这种特性使得其很适合作为多维数据索引并应用于相似性数据搜索[11,15-16,20-22]. 其中,LSB-Forest 是最早将空间曲线技术引入 LSH 索引的方法,因为复合 LSH 键值也是一种多维数据. 不过该方法关注的重点是如何借助 z-order 空间曲线技术实现近邻搜索半径自适应扩充,并没有深入考虑空间曲线对 I/O 效率的改善,因此 LSB-Forest 在进行近邻查询时仍旧需要耗费大量的随机 I/O.

SK-LSH 率先指出了空间曲线下近邻候选点的局部分布对改善 I/O 效率的意义,提出了第一个完整的基于排序的 LSH 索引框架,实现了 I/O 效率的提升. 但是 SK-LSH 没有深入考虑曲线特性的不同对近邻点局部分布的影响,选择了一种较为简单的 row-wise 曲线构造空间线序,导致近邻查询算法依旧存在多种局限(见第 1 节).

本文进一步探究使用空间曲线技术改善 LSH 索引的性能. 我们对多维索引和相似性搜索领域的相关工作进行了考察,汇总了几类最为常用的空间曲线技术,分别是 row-wise 曲线、z-order 曲线、Gray 曲线和 Hilbert 曲线,见图 2. 这些曲线在特性上存在较大的差异,有可能对 LSH 的索引性能产生不同的影响,本文选取这几类曲线作为代表来进行下一步的研究.

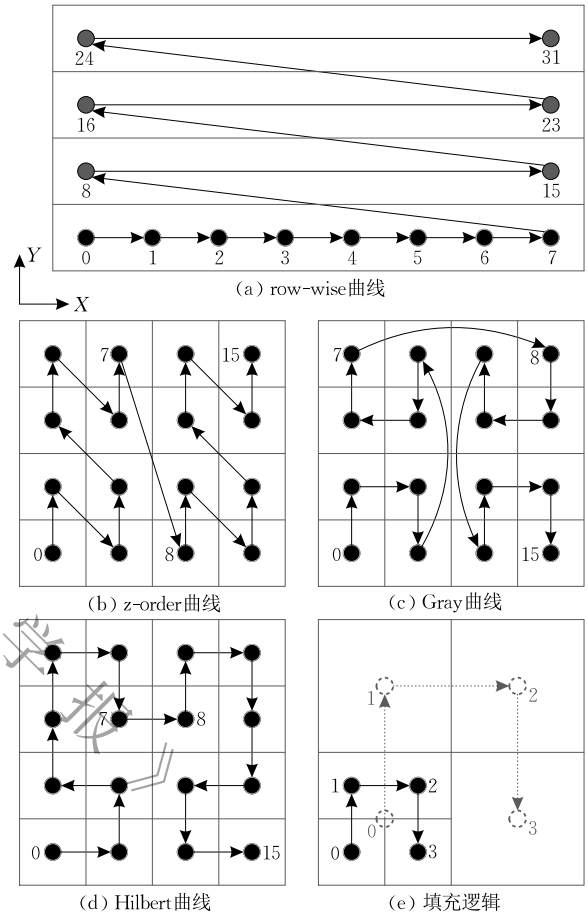


图 2 几种典型的空间填充曲线

直观地,能够看到 row-wise 曲线与其他曲线在填充特性上的不同. 最明显的区别体现在曲线填充时优先级的分配上. row-wise 曲线倾向于为不同维度设定不同的填充优先级,如图 2(a),X 维度的填充优先级要大于 Y 维度. 因此,曲线会率先填充高优先级的 X 维度,完成之后进入次高优先级的 Y 维度上移动一个网格,然后返回高优先级的 X 维度继续填充. 假设二维空间中某一网格的坐标是(x,y),那么它在曲线上的编号就是 $x+y\cdot I$,其中 I 代表 Y 维度上的网格总数量. 不妨称这种填充特性为“维度优先遍历(Dimension-First-Traverse,DFT)”特性.

另外几类曲线则呈现出一种从邻域空间逐步向外扩充的填充趋势,即曲线总是率先填充一个小的

邻域，完成后，再扩大到一个较大的邻域继续填充. 在从局部逐渐向外扩充的过程中，曲线的填充逻辑呈现出明显的递归性. 以图 2(c)中的 Gray 曲线为例，图 2(e)显示的是 Gray 曲线的两阶填充逻辑，可以看到，曲线在左下角 2×2 小网格里的填充逻辑和将整个 4×4 网格看作一个 2×2 大网格的填充逻辑是一致的. 同样，z-order 曲线和 Hilbert 曲线也呈现出类似的填充规律. 不妨称这种填充特性为“邻域优先遍历(Neighborhood-First-Traverse，NFT)”特性.

3.3　最优空间线序

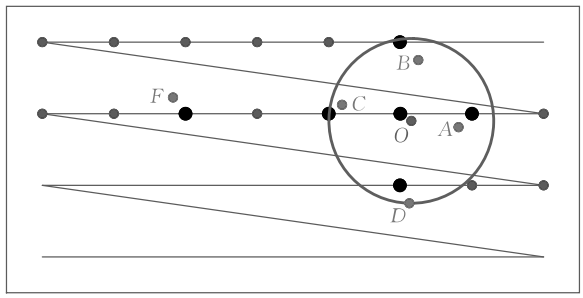
曲线填充特性的不同可能对近邻候选点的局部分布产生不同影响，为便于比较，不妨定义一种曲线上的近邻候选点局部分布质量评价标准：一条空间填充曲线，如果能在相同的局部范围内汇集更多近邻点，就称该曲线能够产生更好的近邻候选点局部分布(简称局部分布). 该评价标准有两个关键点，一个是“局部范围”，一个是“更多近邻点”. 前者用来保证近邻候选点的加载采用顺序 I/O 来完成，后者用来反映候选点局部分布的质量. 这两者必须同时满足才能称曲线产生了更好的局部分布.

直观上看，NFT 曲线能够比 DFT 曲线产生更好的近邻分布. 因为近邻点一般分布在查询点周围的邻域，NFT 曲线优先遍历邻域，有可能更早地“捕获”近邻点；但是对于 DFT 曲线，这取决于近邻点的位置：若近邻点落在高优先级维度上，就会被很快捕获，否则则很难聚集在局部. 而且后面这种情况在高维空间会更加频繁，因为大部分维度都是较低优先级的维度.

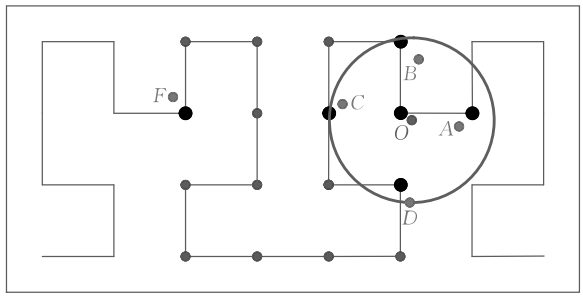
3.3.1　直观对比

为更清晰地说明这一点，首先我们选取 row-wise 曲线和 Hilbert 曲线分别代表 DFT 和 NFT 曲线在二维空间作一些简单对比. 我们对比了两个示例，分别见如图 3 和图 4. 其中，O 是查询点，$\{A,B,C,D\}$是其真实前 4 最近邻，F 是一个距离 O 比较远的点. 灰色和黑色圆点是空间中部分网格中心，其中黑色圆点突出显示几个待观察点归属的网格. 定义两点在曲线上的距离为二者沿着曲线相距的网格数. 每个示例图中前两个子图是原始空间中的图示，第三个子图是将原始空间映射到 1 维空间后的图示.

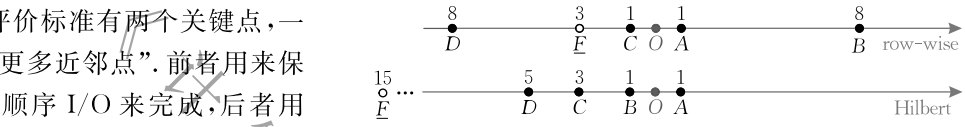
先看第一个示例. 如图 3(c)，可以直观地看到在这一示例中经过 Hilbert 曲线映射后的近邻点分布要比经过 row-wise 曲线映射后的更加集中.



(a) row-wise

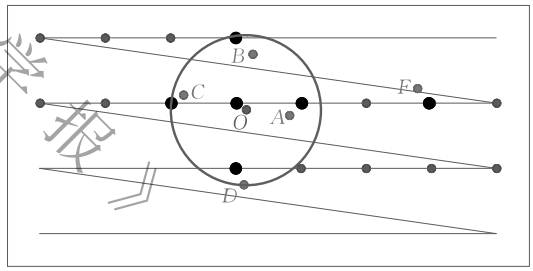


(b) Hilbert

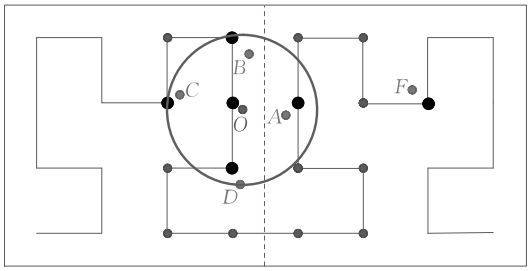


(c) 映射后的近邻分布

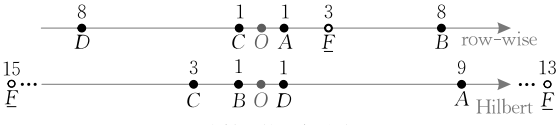
图 3　二维空间近邻分布对比(一)



(a) row-wise



(b) Hilbert



(c) 映射后的近邻分布

图 4　二维空间近邻分布对比(二)

表 1 列举了$\{A,B,C,D,F\}$几个点在曲线上到 O 的距离. 根据平均距离和标准差可以看到，Hilbert 曲线上四个近邻点的局部分布要优于 row-wise 曲

线(平均距离更小,分布更加集中).究其原因,在于row-wise曲线没有较早地捕获B、D两点.正如之前所预料的,由于B、D两点相对于O落在了较低优先级的维度上,需要等待曲线完整地遍历一次高优先级的维度之后才会被捕获,因此二者在曲线上距离O都比较远.对于Hilbert曲线,由于邻近的空间总是会被率先遍历,因此位于O邻域的4个近邻都能很快地被捕获.

表1 近邻分布统计

曲线	O与$\{A,B,C,D,\underline{F}\}$距离	近邻平均距离与标准差
row-wise	$\{1,8,1,8,\underline{3}\}$	4.5,3.0
Hilbert	$\{1,1,3,5,\underline{15}\}$	**2.5,1.4**

如果给近邻查询限定一个加载范围R,表示只将距离查询点R个网格之内的点作为近邻候选点.那么,当$R=3$时,row-wise曲线上的近邻召回率是50%,Hilbert曲线上是75%;$R=5$时,row-wise曲线的召回率还是50%,Hilbert则能够返回全部近邻.直到$R=8$时,row-wise曲线才能返回全部近邻,在此之前,B,D两点会一直被漏报.这种近邻易漏报的问题就是row-wise曲线在近邻查询时的第一个局限,是由于曲线维度优先遍历的特性造成的.

此外,维度优先遍历的特性还会产生近邻误报问题.图3(a)中,原本距离O比较远的F点由于落在了高优先级的维度上,使得其在row-wise曲线上距离O只差3个网格.这就意味着,只要$R\geq 3$,row-wise会一直把F点当作近邻候选点返回,造成近邻误报的问题.而在图3(b)中,Hilbert曲线由于邻域优先遍历的特性,会先于F之前捕获处在局部小邻域的4个近邻点,而后扩充到更大的邻域捕获F点,从而可以更好地避免这种问题的发生.

不过,NFT曲线并非总是能够完整地召回同一邻域范围中的近邻.由于NFT曲线需要递归地对空间进行划分,并优先遍历某一个子空间,这就造成子空间之间存在遍历优先级的不同,如果某一查询点的邻域跨越了多个子空间.则其近邻点就有可能获得差距很大的线序.如图4(b)和图4(c),O的近邻A点由于与其他三个近邻点落入了不同子空间之内(由虚线分隔),其所获得的线序就与另外三个点产生了很大差距.从而无法在映射后和其他点聚集在局部,很容易成为FN点.而对于row-wise曲线,近邻点映射之后的分布几乎没有变化.如果取近邻加载范围$R=3$,则row-wise依旧会产生较多的FN点(即B、D),并引入FP点F;而Hilbert曲线则会引入一个FN点(即A),不过依旧能很好地规避FP点(即F).

因此,直观上看,两种曲线映射都存在一定的优势和劣势区间.DFT曲线能够较好地召回高优先级维度上的近邻点,但是也容易在高优先级维度上引入FP点,而处在较低优先级维度上的近邻则比较容易形成FN点;NFT曲线由于优先遍历邻域,能够较好地避免非邻域中的点成为FP点的情况,且对于处在曲线遍历邻域中的近邻也能够较好的召回.但是对于跨越了多个曲线填充时划分的子空间的近邻邻域,则有可能形成FN点.

3.3.2 量化分析

为了更加清楚地考察两类曲线在优势和劣势区间上的具体差距,特别是在多维空间(10～20维),我们在本节作进一步的分析.由于目前难以在多维空间对曲线性能做出理论上的分析,本文选择在生成的一系列数据集上作进一步的量化观察.

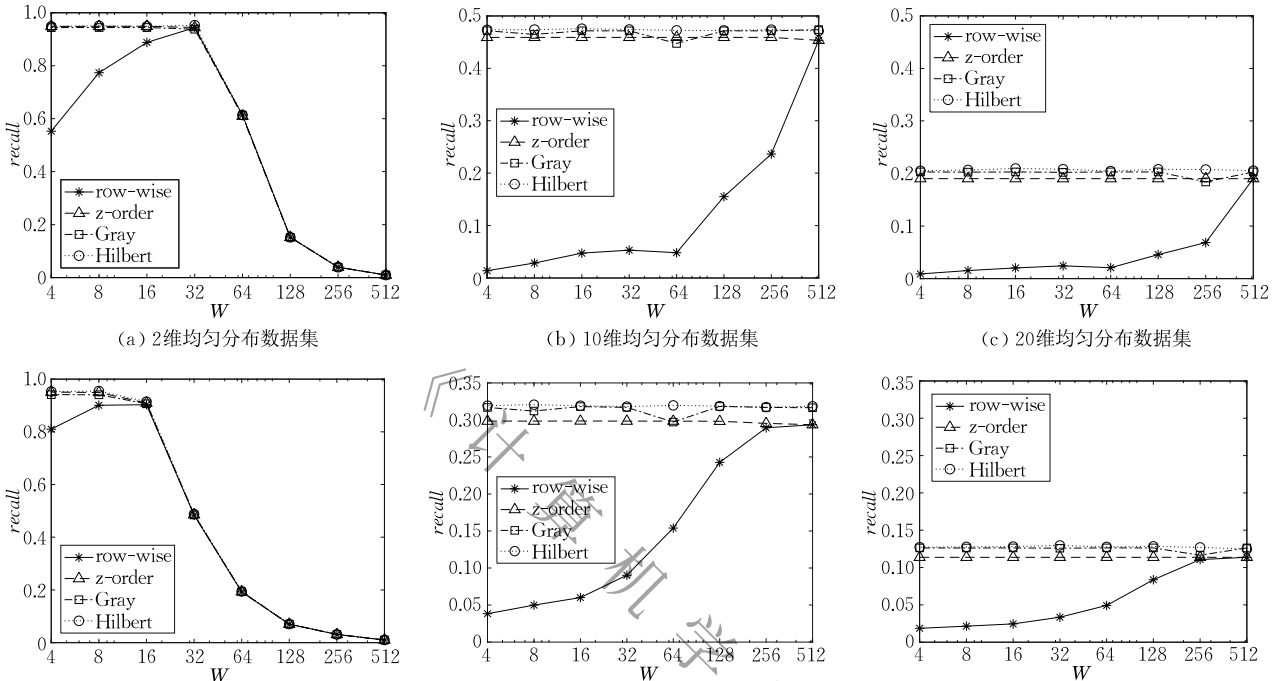
数据集维度设定. 需要说明的是,生成数据集的维度参照的是实际排序方案中复合哈希键值的维度而非原始数据的维度.因为基于排序的LSH方案并不直接在原始空间进行曲线映射,而是将原始数据经过复合LSH函数投影得到复合哈希值,在复合哈希值所在的多维空间进行映射并排序.这么做,一是因为原始数据往往具有很高的维度,直接映射会有很大的计算开销;二是因为一些研究表明大多数真实高维数据集的本征维度通常远远小于其原始维度[23].凭借LSH函数出色的距离保持能力,复合哈希值能充分反映原始数据间的相似性,因此对复合哈希值排序可以近似等价于对原始数据排序.

数据集生成. 本文一共生成了两组观察数据集,分别是均匀分布数据和高斯分布数据.每一组数据集包含$\{2,10,20\}$三个维度的观察数据集.每个数据集含有200000个基本数据点和200个查询点,所有维度的坐标范围限定在$[0,1024]$.

性能考察指标. 实验过程如下,先将每个数据集所在的空间划分为宽度为W的网格,然后选择一种空间曲线遍历所有网格进行填充.这样,每个数据点都会落入一个网格从而被映射到一个线序值,将数据点按照线序的顺序重新排列存放在磁盘.近邻查询过程类似图3和图4中的过程,即给定候选点范围,计算近邻召回率.不过,由于高维空间变得稀疏,候选点范围R不再设定为曲线距离而是设置为数量.即给定查询点q,从曲线上q落在的线序值开始,左右各加载R个点作为近邻候选点.最后从这

$2R$ 个候选点中找出 k 个最近邻，计算其中真实 k 个最近邻的比例作为近邻召回率，以此衡量该曲线上的局部分布质量. 实验中，取 $R=500$，$k=10$，改变 W 以变化空间划分粒度，对比结果见图 5.

通过观察，可以发现：(1) row-wise 曲线对 W 的变化十分敏感，而几类 NFT 曲线的性能对 W 的变化则比较稳定，且一直优于 DFT 曲线的性能；(2) row-wise 曲线比较“依赖”大的桶宽：在桶宽较小时，row-wise 曲线的召回率同其他几类 NFT 曲线存在很大差距，只有在 W 足够大($W=512$)时才会接近其他几类 NFT 曲线；(3) 在 2 维空间中，当 W 较小时，几类曲线的表现正常(即符合前两点规律)，但是当 W 增大到一定程度后，所有曲线的性能会急剧下降，如图 5(a)中 $W=32$，图 5(d)中 $W=16$.



(a) 2维均匀分布数据集　(b) 10维均匀分布数据集　(c) 20维均匀分布数据集

(d) 2维高斯分布数据集　(e) 10维高斯分布数据集　(f) 20维高斯分布数据集

图 5　空间曲线近邻查询量化对比

关于前两点，之前提到，row-wise 曲线容易在局部错失很多近邻点，形成近邻漏报，而设置较大的桶宽能够帮助缓解这一问题. 因为较大的桶宽能够更早地将这些漏报点捕获在同一个网格. 如图 3(a)和图 4(a)，B、D 两点在当前较小的桶宽下成为了漏报点，如果将网格的宽度扩大一倍，则有可能一开始就将 A，B，C，D 划分在和 O 同一个网格，从而被很快加载. 然而根据式(2)，桶宽增大会使得两个较远点的碰撞概率增大，导致候选集中混入更多假阳性点，容易降低近邻查询精度. 为了充分过滤这些干扰点以保证查询精度，需要增加 LSH 函数以提升过滤强度，然而这么做会使得索引和计算开销增大. 这就是之前提到的基本排序方案的第三个局限，我们将在 5.1 节分析 W 对近邻查询的影响中对此作进一步的讨论.

相比之下，几类 NFT 曲线的性能则一直比较稳定，且一直优于 DFT 曲线的性能，这种情况在多维空间尤为明显. 这说明了在多维空间中，邻域优先遍历的特性能够比维度优先遍历更好地进行 FP 点的过滤，同时减少 FN 点的产生，从而使得近邻查询时，局部邻域内能够聚集更多高质量近邻点，从而具有更高的近邻召回率.

此外，几类 NFT 曲线的近邻分布质量基本不受网格宽度 W 变化的影响，也就不会产生 DFT 曲线的上述局限性. 由于网格宽度在下一步设计索引结构时对应着 LSH 的桶宽，是索引结构中一个十分重要的参数. NFT 曲线的这一特性会为克服基本排序方案对 LSH 关键参数的敏感性以进一步提升查询算法的实用性奠定基础.

第三点反映了使用空间曲线实现高效近邻查询的前提：对数据空间充分划分. 只有空间中划分出足够多的网格，在使用曲线填充后，线序值才会形成足够的区分力. 否则，近邻点和干扰点混在一起，难以有效区分，近邻查询性能就会下降，这种情况主要出现在网格粒度较大的时候. 举例，图 5(a)中，当 $W=64$ 时，每个维度最多划分出 $1024/64=16$ 个区段，

整个空间则只有 $16^2=256$ 个网格，相对于 200 000 的数据总量，区分力相对来讲是不够的. 不过，这一问题会随着维度的升高而逐渐消失，因而，在几个更高维度的数据集上，几类 NFT 曲线的性能都没有随 W 的变化而有明显的波动.

数据的不同特性对曲线的以上三点趋势没有产生较大的影响. 相对于 DFT 曲线与 NFT 曲线较大的性能差距，几类 NFT 曲线之间的性能则比较接近，尤其是 Hilbert 曲线和 Gray 曲线. 不过，根据图 5，整体上看，Hilbert 曲线的性能更好一些. 此外，图 5 中的结果是重复生成数据集 10 次后的平均结果，为了更加细致地比较 Hilbert 曲线和 Gray 曲线的优劣，实验详细统计了这两种曲线在 10 次实验中 8 种不同 W 取值下各自取得最优性能的次数(见表 2)，结果显示 Hilbert 曲线占优次数更多. 因此，本文提出了一种新的基于排序的 LSH 索引方案(Optimal Order LSH，O2LSH)，该方案利用 Hilbert 曲线构造线序为复合哈希键值排序，实现近邻候选点的最优排序，从而最大程度地改善磁盘访问效率，同时提升最近邻查询的准确率.

表 2 Gray 曲线与 Hilbert 曲线优势次数比较

曲线占优次数	均匀分布数据		高斯分布数据	
	10 维	20 维	10 维	20 维
Gray 曲线	35	32	31	28
Hilbert 曲线	45	47	49	52

4 O2LSH

本节详细介绍如何在 O2LSH 中部署更优的线序，继而建立索引结构并进行 ANN 查询的过程.

4.1 线序定义

更优线序的构建是 O2LSH 进行索引与查询的核心. 为便于描述，首先给出一些必要的定义.

定义 2. 原始数据点的线序值. 给定 d 维数据点 $\boldsymbol{p}\in R^d$，其在复合哈希函数 G 下的哈希值 $K=C(\boldsymbol{p})=\{k_i\}_{=1}^{m}$ 是一个 m 元组. 构造 m 维空间的 Hilbert 曲线[24]将 K 在曲线上映射，得到一个整数值，称其为 $\boldsymbol{p}$ 的线序值，记为 O_p.

定义 3. 线序值最长公共前缀. 数据点 $\boldsymbol{p}$ 的线序值 O_p 可以表示成一个二进制序列 $O_p=o_1o_2\cdots o_U$，其中 U 是二进制串的长度. 我们用 $pref(O,l)=(o_1o_2\cdots o_l)$ 表示 O_p 的长为 l 的前缀，其中 $0\leq l\leq U$. 对于两个给定线序值 $O_1=(o_{1,1}o_{1,2}\cdots o_{1,U})$ 和 $O_2=(o_{2,1}o_{2,2}\cdots o_{2,U})$，如果有 $pref(O_1,l)=pref(O_2,l)$ 成立，且 $pref(O_1,l+1)\neq pref(O_2,l+1)$，则称 $pref(O_1,l)$ 是二者的最长公共前缀，最长公共前缀长度(Length of Longest Common Prefix，LLCP)为

$$LLCP(O_1,O_2)=l \tag{3}$$

如果 $pref(O_1,U)=pref(O_2,U)$，则称 O_1,O_2 相等，$LLCP(O_1,O_2)=U$.

根据线序值最长公共前缀的定义可以进一步定义 O2LSH 所建立的线序关系：

定义 4. 线序关系. 给定两个线序值 O_1,O_2，定义它们之间的偏序关系如下：

$$\begin{cases} O_1<_O O_2, & \text{如果 } l<U,\text{且 } o_{1,l+1}<o_{2,l+1} \\ O_1=_O O_2, & \text{如果 } l=U \\ O_2<_O O_1, & \text{如果 } l<U,\text{且 } o_{1,l+1}>o_{2,l+1} \end{cases} \tag{4}$$

其中，$l=LLCP(O_1,O_2)$.

4.2 构建索引结构

线序定义好之后，可以进行 O2LSH 索引结构的构建，过程如下：

(1) 将数据集 $\boldsymbol{D}$ 中每个数据点在一个复合哈希函数上投影，将得到的复合哈希键值在 Hilbert 曲线上映射，得到所有点的线序值集合 $\boldsymbol{O}$；

(2) 按照定义 4 中定义的线序对 $\boldsymbol{O}$ 中的线序值排序 Z，得到排序后的线序值集合 $\boldsymbol{O}'$；

(3) 依照 $\boldsymbol{O}'$ 中的顺序重新排列原始数据，得到排序后的数据集 $\boldsymbol{D}'$，将 $\boldsymbol{D}'$ 存放到磁盘上，如图 6；

(4) 假定一个磁盘页面能容纳 B 个数据点，整个数据集 $\boldsymbol{D}'$ 将占据 $\eta=\lceil n/B\rceil$ 个磁盘页面. 对于每个页面，提取第一个和最后一个数据点的线序值 α 和 β 为代表. 为所有代表线序值建立 B^+ 树进行索引，即可有效索引所有原始数据 $\boldsymbol{D}'$. 由 B^+ 树和磁盘中存放的 $\boldsymbol{D}'$ 共同构成 O2LSH 的一个哈希表，如图 6 所示.

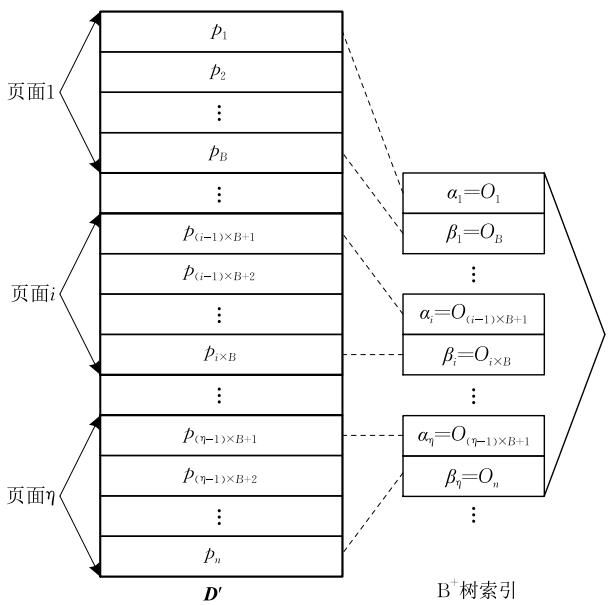


图 6 O2LSH 的一个哈希表

(5) 重复(1)～(4)的步骤建立 L 个哈希表 $T=\{T_i\}_{=1}^{L}$，完成 O2LSH 的多索引哈希表的构建.

可以看到，O2LSH 的索引结构主要由两部分组成：一是依照新的空间线序在磁盘上重新排列的原始数据，二是由代表哈希键值组成的 B^+ 树索引. 索引结构最大的特点主要有两方面：(1) 使用更加优秀的线序对数据点重新排列，近邻点的局部分布质量更高；(2) 以磁盘页面为单位进行组织与索引. 我们知道，磁盘中的数据通常是以页面为单位存储和访问的，传统的 LSH 外存索引采用随机读取的方式加载候选点，虽然每一次 I/O 能读到一个页面的数据，但是其中一般只有一个点是所需要的，这就存在 I/O 利用率低的问题. O2LSH 的索引结构以页面为单位组织和索引数据，将来候选点加载时也是以页面为单位读取，得益于更好的局部分布，一次 I/O 操作就能够获得更多高质量候选点，I/O 效率得以提升. 此外，从索引构建的第(4)步可以看到，B^+ 树并不直接在全部复合哈希键值上构建，而是以每个数据页面提取的代表哈希键值为基础. 这样做，能够保证 B^+ 树索引的效果，也能进一步降低 B^+ 树索引的规模，从而能够最大程度地降低 B^+ 树的高度，提升近邻查询时的搜索效率.

4.3　最近邻查询算法

O2LSH 进行 k-近似最近邻查询(k-ANN)的算法流程见算法 1. 为便于控制查询性能，算法的终止条件设置为查询时允许加载的数据页面数量 N_P. 查询过程大体可分为两个阶段：(1) 候选页面确定阶段；(2) 候选点加载与最近邻验证阶段. 第一个阶段的任务是找出“质量”最好的 N_P 个起始页面. 这需要定义查询点 q 到一个数据页面的距离，作为衡量一个数据页面“质量”好坏的标准.

定义 5.　查询点到数据页面的距离. 给定查询点 $\boldsymbol{q}$ 和一个磁盘数据页面 P，O_q 是查询点的线序值，α 和 β 是页面 P 的代表线序值($\alpha\leq_O\beta$)，定义 $\boldsymbol{q}$ 到页面 P 的距离如下：

$$DIST(q,P)=\begin{cases} dist(O_q,\alpha), & \text{如果 } O_q\leq_O\alpha\leq_O\beta \\ 0, & \text{如果 } \alpha\leq_O O_q\leq_O\beta \\ dist(O_q,\beta), & \text{如果 } \alpha\leq_O\beta\leq_O O_q \end{cases} \quad (5)$$

其中 $dist(,)$ 表示两个线序值间的距离. 例如两个线序值 O_1，O_2，其计算方式为 $dist(O_1,O_2)=U-LLCP(O_1,O_2)$.

根据这一定义，即可进行近邻查询的初始页面定位和迭代搜索，详细算法如下.

算法 1.　k-ANN 查询.

输入：$\{T_i\}_{=1}^{L}$，$\boldsymbol{q}$，k，N_P

输出：kNN

1. (初始化)初始化候选页面集合 $P_C=\varnothing$；

2. (定位初始页面)在每一个哈希表 T_i 上定位到距离 $\boldsymbol{q}$ 最近的页面，记为 T_i 的左页面 P_{iL}，其右侧页面记为 T_i 的右页面 P_{iR}；

3. (迭代确定候选页面)根据定义 5，计算出 P_C 中当前距离 $\boldsymbol{q}$ 最近的一个页面 P^*. 若 P^* 来自某哈希表 T_i 的左页面 P_{iL}，则更新 P_{iL} 指向 P^* 的左侧页面；否则，更新 P_{iR} 指向 P^* 的右侧页面. 重复此步骤 N_P 次；

4. (加载候选点，验证 k 最近邻)对每个哈希表 T_i，沿磁盘顺序加载从 P_{iL} 到 P_{iR} 间的所有页面. 计算 $\boldsymbol{q}$ 与所有加载点的距离，最近的 k 个作为 kNN 结果返回.

4.4　近邻查询效果分析

O2LSH 与 SK-LSH 都是基于排序的 LSH 索引方案，二者在索引与查询机制上十分类似. 最大的不同，在于 O2LSH 使用了 NFT 特性的曲线，能够进一步改善近邻候选点的局部分布，使得 O2LSH 在近邻查询时能够加载到更多高质量的近邻点，从而获得比 SK-LSH 更好的查询性能.

结合图 7 可以更加清楚地说明这一点. 图 7 描绘了这两种方法在索引构建好之后数据点在磁盘中的分布情况. 图中一个方框代表一个磁盘页面，可以容纳多个数据点. 对于给定的查询点 O，所有的黑色实心点是其真实 k 近邻. 查询时，首先在 B^+ 树中搜索确立 O 所对应的磁盘页面，即起始页面(算法 1 步骤 2)，然后执行一个双向搜索的过程确立一系列候选磁盘页面(算法 1 步骤 3)，最后从这些候选页面中加载数据点进行精炼(算法 1 步骤 4).

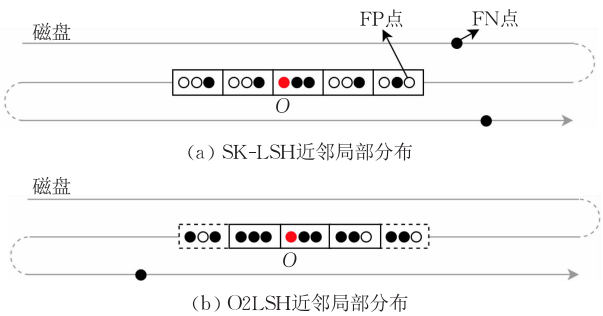


图 7　O2LSH 与 SK-LSH 近邻局部分布对比

对于 SK-LSH，row-wise 曲线 DFT 的特性会给其所确立的候选页面中引入更多 FP 点，并丢失更多真实近邻点(即 FN 点)，近邻查询的准确率因此随之下降. 而对于 O2LSH，NFT 特性的曲线能更好地过滤 FP 点并丢失更少的真实近邻. 最终的结果，O2LSH 能够产生更好的近邻局部分布，即在相

同(甚至更少)的局部页面中汇集更多近邻点(如图7),从而可以用相同(甚至更少)的I/O开销加载到这些高质量候选点,且取得更高的近邻查询精度.根据5.3节的近邻查询实验结果,O2LSH确实在I/O性能和近邻查询准确率上同时取得了对SK-LSH的领先.

4.5 复杂度分析

空间复杂度. O2LSH的索引结构共包含L个哈希表,每个哈希表包含一个B^+树和一个原始数据集.假设一个磁盘页面能容纳B个点,则一个B^+树的空间开销是$O(nm/B)$,其中n是数据集规模,m是复合哈希键值的维度.因此O2LSH总共的空间开销是$O(Ln(m/B+d))$.通常情况下$m/B \ll d$,因此可以看到O2LSH索引中最主要的空间开销是原始数据.

时间复杂度. O2LSH的时间开销主要包括两部分:(1)在L个B^+树中搜索定位起始页面的时间;(2)加载并处理N_P个候选磁盘页面的时间.第一部分中,总共需要处理$O(LE)$个页面,其中E表示每个B^+树的高度.在O2LSH中,每个B^+树通过索引$\eta=\lceil n/B \rceil$个页面的代表哈希键值来索引全部原始数据,B^+树的序$b=Bd/m$,因此$E=\log_b 2\eta$.由于一个磁盘通常能容纳非常多的复合哈希键值,也就是说b的值很大,因此O2LSH中B^+树的高度都不会很大,因此整个第一部分不会耗费太多时间.第二部分的耗时包含两个方面,一个是加载N_P个页面所耗费的I/O时间,另一部分是计算全部候选点的原始距离进行近邻点精炼的时间$O(BdN_P)$.由于N_P决定着O2LSH的I/O开销,实际中我们一般会将其设置为比较小的值,从而让O2LSH的第二部分也不会耗费太多时间.

I/O开销. 根据时间开销的分析,可以直观地看出O2LSH的I/O开销,共包括两部分.一个是在L个B^+树中搜索定位初始页面所需要的磁盘访问,一个是对N_P个候选页面的访问,总共是$LE+N_P$.

5 实验结果与分析

本节设计实验评估O2LSH的ANN查询性能,并与几类最先进的LSH索引技术,包括C2LSH、SK-LSH、SRS以及QALSH,进行综合的性能对比.实验在一台惠普工作站上进行,CPU主频3.6 GHz,内存64 GB,硬盘2 TB,操作系统为Ubuntu 14.04 LTS,代码采用C/C++编写,编译器为g++4.6.3.

选取了6个公开的真实多媒体特征数据集进行对比实验.数据类型涵盖文本、图像和音频数据.详细信息见表3,其中n表示数据集包含的特征向量个数,d表示特征向量维度.实验中,Mnist和Gist1M数据集的磁盘页面大小设置为16 KB,其余数据集为4 KB.

表3 实验数据集

名称	n	d	特征来源
WT①	269 648	128	小波纹理图像特征
Sift1M②	1 000 000	128	图像SIFT特征
Audio③	53 387	192	音频特征
Glove200d④	1 192 514	200	twitter文本词频特征
Mnist⑤	69 000	784	手写字体图像特征
Gist1M⑥	1 000 000	960	全局图像特征

5.1 性能评估标准

从三个方面衡量算法在外存环境下进行ANN查询的性能:*ratio*、I/O和空间开销.

ratio. 评估ANN查询的准确率.给定查询点$\boldsymbol{q}$,令$o_1^*,o_2^*,\cdots,o_k^*$依次代表$\boldsymbol{q}$的前k个真实最近邻,令$o_1,o_2,\cdots,o_k$依次代表查询算法返回的k个最近邻.则该查询点$\boldsymbol{q}$的*ratio*的计算如下:

$$ratio(\boldsymbol{q})=\frac{1}{k}\sum_{i=1}^{k}\frac{\|o_i,\boldsymbol{q}\|}{\|o_i^*,\boldsymbol{q}\|} \tag{6}$$

I/O. 算法进行ANN查询过程中访问磁盘页面的次数,是衡量算法进行外存ANN查询效率的一个重要指标.

空间开销. 指算法进行ANN查询所需数据总的存储开销,一般包括数据集和索引两部分.

每个数据集随机选取200个点作为查询点,取所有查询点的平均性能作为最终性能.

5.2 O2LSH与SK-LSH性能对比

本文的O2LSH和基本排序方案SK-LSH都是基于空间线序的LSH索引技术,二者具有直接的竞争关系.本小节专门对这两个方法进行详细的ANN查询对比.在四个不同参数(LSH函数桶宽W、复合哈希函数中LSH函数个数m、ANN查询终止条件N_p和哈希表数目L)变化下评估两种方法进行$k=10$近邻查询的性能.图8列出的是在Sift1M数据集上的对比结果.

① http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm
② http://corpus-texmex.irisa.fr/
③ http://www.cs.princeton.edu/cass/audio.tar.gz
④ http://nlp.standford.edu/projects/glove/
⑤ http://yann.lecun.com/exdb/mnist/
⑥ http://corpus-texmex.irisa.fr/

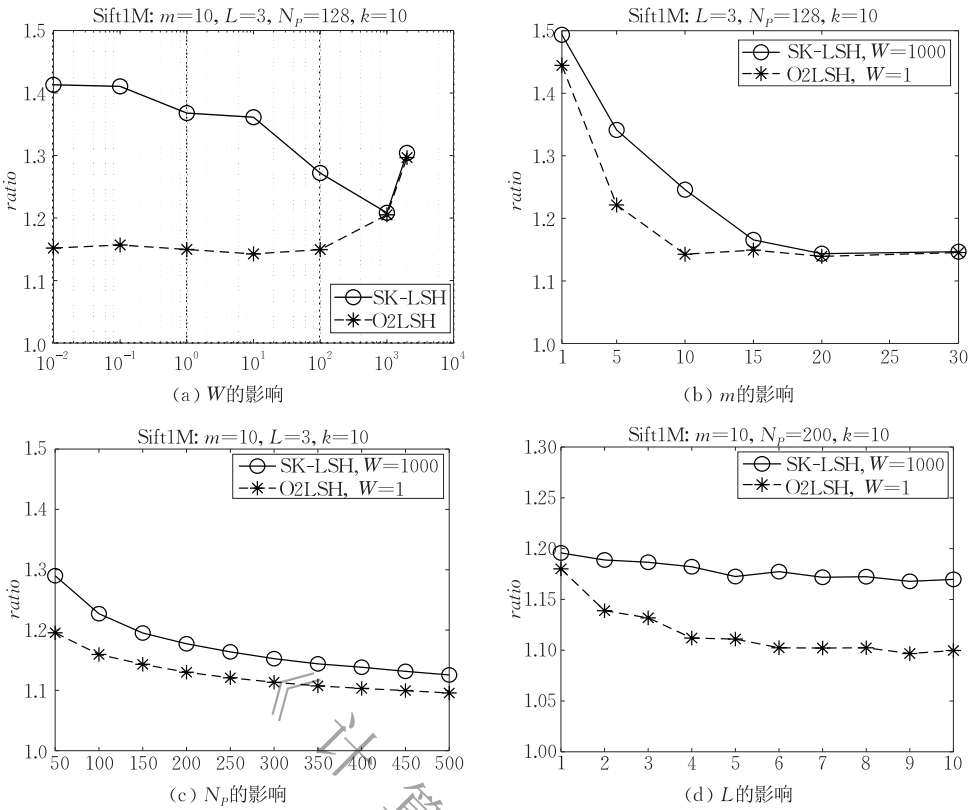


图 8　O2LSH 与 SK-LSH 性能对比

***W* 的影响**. 图 8(a)中两种方法的 LSH 函数桶宽 W 都以 10 倍的速率变化. 可以看到，SK-LSH 的准确率对桶宽变化十分敏感：W 偏小或偏大时 *ratio* 都比较差，只有在设置合适的桶宽(图中为 $W=1000$)时才能取得最好的准确率；而 O2LSH 的准确率则相对比较稳定，在 W 很大的变动范围内(从 $10^{-2}\sim10^{2}$ 跨 4 个数量级的范围)*ratio* 几乎没有波动，一直处于最优值，且其最优值优于 SK-LSH 的最优 *ratio*. 但是在 W 取 1000 和更大数值时，O2LSH 的 *ratio* 开始变差，出现和 SK-LSH 的性能重合的现象.

正如前面所分析的，SK-LSH 对 W 比较敏感是因为 DFT 特性让 row-wise 曲线在局部容易错失近邻点，设置较大的桶宽一定程度上能弥补这一问题. 但根据图 8(a)，两种曲线的桶宽都不宜设置过大. 图 8(a)中，我们看到 SK-LSH 曲线的准确率在 $W>1000$ 之后没有进一步改善，而是出现拐点；O2LSH 在 $W>1000$ 之后准确率也急速变差. 其原因都是桶宽设置过大使得单个哈希函数上划分不出足够多的桶，使得整个空间无法被充分划分造成哈希值混淆在一起，近邻难以区分，最终导致准确率降低. 这和图 5 中出现的第三点现象是同一个原因，反映了应用空间曲线实现高效近邻查询的前提：对数据空间充分划分. 为了便于在实际应用中将基于排序的方法应用于不同的数据集，我们给出了一种评估空间划分程度的方法.

首先计算不同数据集在 LSH 函数上投影的平均范围，算法如下：

算法 2.　计算数据集的 LSH 平均投影范围.

输入：数据集 $\boldsymbol{D}$

输出：平均投影范围 R_H

1. 随机生成足够多的 LSH 随机向量 $\boldsymbol{a}$；

2. 将数据集 $\boldsymbol{D}$ 在所有随机向量上投影，计算每个向量上的投影值范围；

3. 所有投影值范围取平均即作为 R_H 返回.

我们在 Sift1M 数据集上共生成 1000 个 LSH 随机向量，根据这一算法可以计算得到 Sift1M 数据集在 LSH 函数上的平均投影范围，约为 3067. 这就意味着，$W=1000$ 时，每个维度平均能划分约 $\lceil R_H/W\rceil=\lceil 3067/1000\rceil=4$ 个桶，整个空间约能产生 1 048 576(即 4^{10})个网格，稍微大于 Sift1M 数据集的规模(即 1 000 000)；而当 $W=2000$ 时，每个维度平均只能划分 $\lceil 3067/2000\rceil=2$ 个桶，整个空间只有约 1024 个网格，网格数量远远小于数据集规模，这种情况下很难保证对数据集进行充分划分，因此不管采用何种空间曲线，哈希值都极容易混淆在一起难以区分. 而反过来，当 $W=100$ 时，整个空间约能产生 30^{10} 个网格，空间被充分地划分，因此 O2LSH 能够在

此情况下产生较好的 ANN 查询效果(如图 8(a)).

SK-LSH 对桶宽敏感且青睐较大桶宽除了会遇到空间划分不充分的问题,还会产生其他两个局限:(1)W 的最优取值范围很小,使得 SK-LSH 面对新数据集时需要进行大量的调参工作才能确立最优性能,算法实用性受到很大影响,这是第一章提到的 SK-LSH 的第二个局限;(2)较大的桶宽在更早地捕获更多近邻点的同时也会引入更多假阳性点,为了对这些新的干扰点进行充分过滤,需要增加 LSH 函数个数,最终导致 LSH 函数开销增大,见图 8(b)中 m 变化时的分析.

***m* 的影响**. 图 8(b)中,SK-LSH 的桶宽定为其在图 8(a)中取得最优 *ratio* 时的桶宽,即 $W=1000$. 对于 O2LSH,由于其对 W 取值不太敏感,W 取 1. 可以看到,两个方法的 *ratio* 一开始都随着 m 值的增大而迅速下降,说明增加哈希函数确实能过滤掉更多假阳性点. 但当 m 增大到一定程度,再增加 m,*ratio* 基本趋于平稳. 说明存在一定的哈希函数数量能够充分过滤掉假阳性点,没有必要一味增加哈希函数. 而且更多的哈希函数意味着更多的存储和计算开销,因此最优的 m 值便取在 *ratio* 刚开始趋于平稳时的值. 按照图 8(b)中的结果,SK-LSH 的最优 m 值约为 20,O2LSH 则在 $m=10$ 时取得最好 *ratio*. O2LSH 使用一半的哈希函数开销取得了和 SK-LSH 相当甚至更优的查询准确率.

正如之前提到的,产生这种区别的主要原因是来自两种方法曲线特性上的不同. SK-LSH 采用维度优先遍历的线序,容易错失更多的近邻点,需要用更大的桶宽来"捕捉". 然而较大的桶宽会导致桶内混入更多假阳性点,需要增多哈希函数来过滤掉;相应地,O2LSH 中邻域优先遍历的线序能够更早地将更多近邻点索引到,混入的假阳性点很少,因此只需要较少的哈希函数进行过滤.

N_P 的影响. 图 8(c)中横坐标改变的是查询的终止条件,即允许加载的候选页面个数 N_P. 可以看到,加载更多候选点,两类方法的准确率都有显著提升. 因为 N_P 增大意味着局部搜索时的范围增大,能够考察更多候选点,因此有机会召回更多最近邻,提升查询准确率. 不过,N_P 增大也会增加算法的 I/O 开销和时间开销. 实际查询时,需要根据需求进行准确率和效率的权衡.

***L* 的影响**. 同 N_P 类似,更多的哈希表一定程度上也都能提升查询准确率. 因为更多哈希表能相互弥补丢失的近邻,有助于克服假阴性问题. 但是,更多的哈希表意味着更大的空间开销. 因此需要权衡查询精度与效率来选取最合适的 L 值. 本文中,我们一般构建 $L=3$ 个哈希表.

5.3 ANN 查询综合对比

本小节,我们将 O2LSH 与目前几类最先进的 LSH 方法(包括 C2LSH、SK-LSH、SRS 和 QALSH)进行综合的 ANN 查询性能对比. 由于这几类算法都证实了对 LSB-Forest 的优越性,此处不再对 LSB-Forest 进行对比.

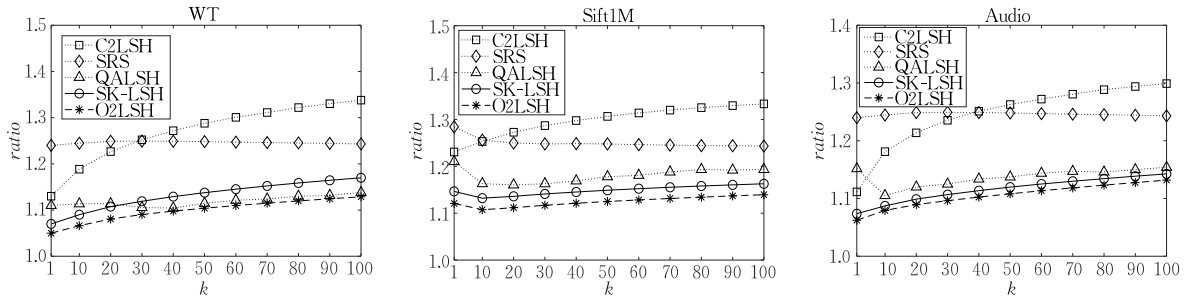
5.3.1 参数设置

各个算法以取得最好的精度与效率平衡为标准选取参数,详细参数设置见表 4. 其中,对于 C2LSH,我们使用其高效率版本(即 $Ct=1$),近似比率 c 统一取 3;QALSH 的近似比率统一取 5 或 6;对于 SRS,参考原论文推荐,近似比率 c 和哈希函数个数 m 分别取 4 和 6. 对于 SK-LSH 和 O2LSH,为深入对比二者的线序索引性能,我们让二者建立同等规模的哈希索引结构,即哈希表数目相同(即 $L=3$)、每个哈希表中复合哈希函数包含的 LSH 函数个数相同(即 $m=10$). W 的变化不会影响索引开销,但是会影响查询精度,因此需要设置到最优值以获取最优精度. 为此,SK-LSH 需要对不同数据集进行调参以确立最优的参数取值;而 O2LSH 对桶宽不敏感,最优值取值范围比较宽,只需要确保哈希映射后的空间被充分划分即可. 为此,我们根据算法 2 统计出各个数据集的平均投影范围 R_H,将 R_H 统一缩小 1000 倍作为最终的 W 值. 各个数据集经过统计得到的 R_H 值和两个方法最终确立的 W 值见表 4. 最后,N_P 兼顾精度和 I/O 开销设置一个合适的值.

实验全部在 6 个数据集上实施,考察各个方法 ANN 查询的 *ratio*、I/O 以及空间开销,结果分别见图 9、图 10 和图 11.

表 4 不同 ANN 查询方法在各数据集上的参数选取

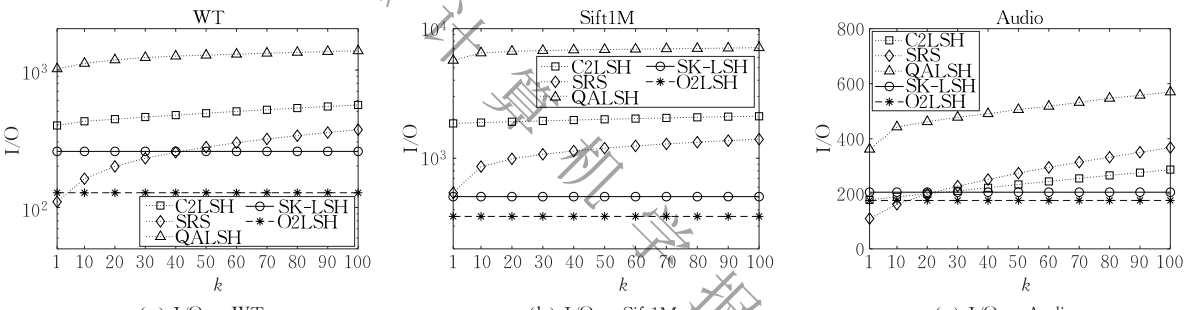
Dataset(R_H)\Method	C2LSH(Ct,c,m)	QALSH(c,m)	SRS(c,m)	SK-LSH(W,m,L,N_P)	O2LSH(W,m,L,N_P)
WT(65.7)	(1,3,230)	(6,19)	(4,6)	(1,10,3,256)	(0.06,10,3,128)
Sift1M(3067)	(1,3,250)	(5,21)	(4,6)	(1000,10,3,500)	(3,10,3,350)
Audio(10.5)	(1,3,204)	(5,16)	(4,6)	(3,10,3,200)	(0.01,10,3,170)
Glove200d(80)	(1,3,252)	(5,21)	(4,6)	(15,10,3,500)	(0.08,10,3,400)
Mnist(15176)	(1,3,208)	(5,17)	(4,6)	(2000,10,3,250)	(15,10,3,200)
Gist1M(23.3)	(1,3,250)	(4,26)	(4,6)	(2.5,10,3,350)	(0.02,10,3,300)



(a) *ratio* on WT　(b) *ratio* on Sift1M　(c) *ratio* on Audio

(d) *ratio* on Glove200d　(e) *ratio* on Mnist　(f) *ratio* on Gist1M

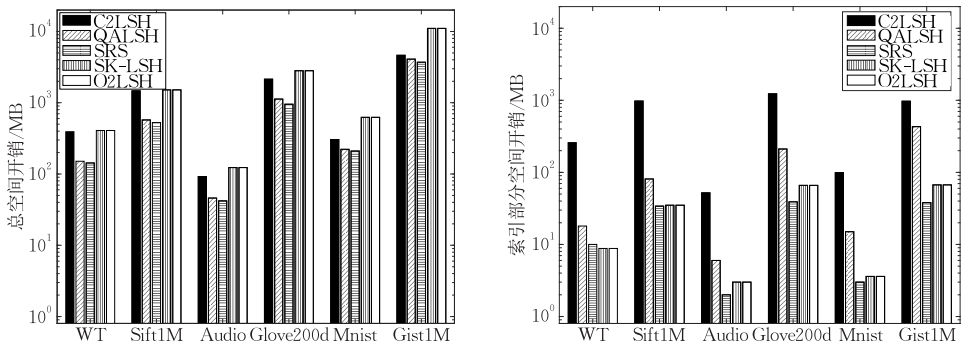
图 9　ANN 查询 *ratio* 对比



(a) I/O on WT　(b) I/O on Sift1M　(c) I/O on Audio

(d) I/O on Glove200d　(e) I/O on Mnist　(f) I/O on Gist1M

图 10　ANN 查询 I/O 开销对比



(a) 总空间开销　(b) 索引结构空间开销

图 11　空间开销对比

5.3.2 综合对比 *ratio* 和 I/O

综合图 9 的 *ratio* 性能和图 10 的 I/O 性能进行观察. 可以发现,SK-LSH 以较少的 I/O 开销取得了较为领先的查询准确率,反映了基于排序的 LSH 思想可以同时取得较好的 ANN 查询精度和 I/O 效率的基本能力;QALSH 的查询精度仅次于 O2LSH 和 SK-LSH,在部分数据集上甚至超越 SK-LSH(WT 数据集),但无一例外地,QALSH 是所有方法中 I/O 开销最大的;SRS 方法的 I/O 性能仅次于 O2LSH 和 SK-LSH,但其查询精度距离二者却有很大差距;C2LSH 的 I/O 性能对维度变化不是特别敏感,但是受数据集规模变化影响较明显:在较小规模的数据集(Audio 和 Mnist)上 I/O 性能可以匹敌甚至超过 O2LSH,但是在大规模数据集(Sift1M、Glove200d 以及 Gist1M)上 I/O 开销迅速增长.

最后,如图 9 和图 10,O2LSH 几乎在所有数据集上都能以最少的 I/O 开销取得最好的查询准确率,领先于基本排序方案和其他先进方案,且领先性不受数据集维度和数据集规模的影响.

此外,我们还统计了所有方法的平均 *ratio* 和平均 I/O,分别见表 5 和表 6,可以看到 O2LSH 能同时取得最好的准确率和 I/O 效率. 其中,相对于基本方案 SK-LSH,O2LSH 在准确率全面提升的情况下,I/O 开销降低了 14%～48%不等. 实验结果证实了本文提出的基于最优线序的索引方案相对于基本排序方案和其他方案的优越性,确实能够有效改善近邻候选点的局部分布,实现在相同(甚至更小)范围内维护更多高质量近邻候选点,最终同时提升 I/O 效率和查询精度.

表 5 ANN 查询平均 *ratio*

ratio	WT	Audio	Sift1M	Glove200d	Mnist	Gist1M
C2LSH	1.268953	1.244570	1.297230	1.241162	1.303211	1.236579
SRS	1.245954	1.245954	1.250638	1.185170	1.245954	1.175362
QALSH	1.118804	1.137593	1.180986	1.200379	1.191545	1.152471
SK-LSH	1.131237	1.115522	1.148983	1.138871	1.149203	1.138134
O2LSH	**1.098780**	**1.104771**	**1.124887**	**1.133074**	**1.129808**	**1.129679**

表 6 ANN 查询平均 IO

IO	WT	Audio	Sift1M	Glove200d	Mnist	Gist1M
C2LSH	483	233	1992	2534	181	648
SRS	263	263	1131	1109	263	752
QALSH	1256	497	6781	11218	646	2776
SK-LSH	262	206	506	506	256	306
O2LSH	**134**	**176**	**356**	**406**	**206**	**256**
(SK-O2)/SK	48.85%	14.56%	29.64%	19.76%	19.53%	16.34%

5.3.3 空间开销

本节我们比较各个近邻查询方法取得以上近邻查询性能所付出的空间开销. 我们使用了两种空间开销指标,一个是查询算法的总空间开销,一个是查询算法索引结构的空间开销. 统计结果分别见图 11(a)和图 11(b).

从图 11(a)可以看到,SRS 总的空间开销最少,其次是 QALSH 和 C2LSH,SK-LSH 和 O2LSH 产生了最大的空间开销. 为便于分析,我们将原始空间开销值进行了转换,转换成一种相对空间开销比例,即拿算法实际的空间开销比上原始数据集的物理大小得出一个比例值,结果见表 7 和表 8.

从表 7 可以明显地看到,对所有这些 LSH 外存索引方法来说,空间开销中最主要的成分是原始数据的存储. 在这一点上,SRS 最为明显,其次是 QALSH.

表 7 总空间开销相对比例

数据集(大小/MB)	C2LSH	QALSH	SRS	SK-LSH	O2LSH
WT(133)	2.95	1.14	**1.08**	3.07	3.07
Sift1M(493)	2.98	1.16	**1.07**	3.07	3.07
Audio(40)	2.30	1.15	**1.05**	3.08	3.08
Glove200d(915)	2.34	1.23	**1.04**	3.07	3.07
Mnist(207)	1.48	1.07	**1.01**	3.02	3.02
Gist1M(3665)	1.27	1.12	**1.01**	3.02	3.02

表 8 索引结构空间开销相对比例

数据集(大小/MB)	C2LSH	QALSH	SRS	SK-LSH	O2LSH
WT(133)	1.95	0.14	**0.08**	**0.07**	**0.07**
Sift1M(493)	1.98	0.16	**0.07**	**0.07**	**0.07**
Audio(40)	1.30	0.15	**0.05**	0.08	0.08
Glove200d(915)	1.34	0.23	**0.04**	0.07	0.07
Mnist(207)	0.48	0.07	**0.01**	**0.02**	**0.02**
Gist1M(3665)	0.27	0.12	**0.01**	**0.02**	**0.02**

这两种方法都只存储一份原始数据，因此二者的相对空间开销比例都是稍稍大于 1 的值. C2LSH 在不同数据集上则出现了一定的波动，虽然也是只用存储一份原始数据，但是 C2LSH 会生成比较多的哈希函数，其个数有时会接近甚至超过原始数据集维度，造成其空间开销也比较高. 不过由于 C2LSH 对维度较好的可扩展性，随着维度的升高，哈希表所占的空间比例相对数据集大小会下降，此时 C2LSH 的相对空间开销会降下来.

而 SK-LSH 和 O2LSH 都是多哈希表结构，需要存储多份原始数据，因此总的空间开销会比较大. 从表 7 中可以看到，二者的相对空间开销值都是大于 3 的数值，这和设置的哈希表个数 L 的取值有关系：O2LSH 和 SK-LSH 在索引构建时所设置的哈希表个数 L 都是 3. 另外可以看到，由于 O2LSH 与 SK-LSH 相比最主要的区别在于改善了线序，二者使用的索引结构在架构上是类似的，因此二者几乎具有相同规模的空间开销.

不过，如果从总的空间开销中减去原始数据，只观察索引部分的大小，如图 11(b)和表 8，会发现 O2LSH 和 SK-LSH 的索引(即 B^+ 树部分)开销显著减少，超越 C2LSH 和 QALSH，仅次于 tiny 级的 SRS 方法.

6　结束语

基于排序的 LSH 索引有助于克服外存环境下海量高维 ANN 查询面临的“维度灾难”和 I/O 性能瓶颈问题. 本文对典型空间填充曲线上近邻候选点的局部分布展开量化分析，发现：(1) 基本排序方案使用的 row-wise 曲线具有 DFT 特性，会给 ANN 查询带来多个局限；(2) NFT 特性的曲线能够产生更好的局部分布，且性能更稳定. 因此，本文选取一种排列效果最好的 NFT 曲线构造了一种最优排序的 LSH 索引，O2LSH，最大程度地改善了近邻候选点的局部分布. 实验结果表明，本文提出的新的索引方案和近邻查询算法能够进一步提升 ANN 查询的 I/O 效率和精度，性能优于基本排序方案和其他先进 LSH 索引. 其中，相对于基本排序方案，O2LSH 能够降低对关键参数变化的敏感性，是一种更加高效、实用的索引方法.

致　谢　感谢焦文菲同学在论文早期阶段做出的宝贵研究工作，感谢各位评阅专家的耐心审阅和提出的珍贵意见！

参　考　文　献

[1] Yuan Pei-Sen, Sha Chao-Feng, Wang Xiao-Ling, Zhou Ao-Ying. c-Approximate nearest neighbor query algorithm based on learning for high-dimensional data. Journal of Software, 2012, 23(8): 2018-2031(in Chinese)
(袁培森，沙朝锋，王晓玲，周傲英. 一种基于学习的高维数据 c-近似最近邻查询算法. 软件学报，2012，23(8)：2018-2031)

[2] Vitter J S. Algorithms and data structures for external memory. Foundations and Trends in Theoretical Computer Science, 2008, 2(4): 305-474

[3] Indyk P, Motwani R. Approximate nearest neighbors: Towards removing the curse of dimensionality//Proceedings of the 30th Annual ACM Symposium on Theory of Computing. Texas, USA, 1998: 604-613

[4] Datar M, Immorlica N, Indyk P, et al. Locality-sensitive hashing scheme based on p-stable distributions//Proceedings of the 20th Annual Symposium on Computational Geometry. New York, USA, 2004: 253-262

[5] Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via hashing//Proceedings of the 25th International Conference on Very Large Data Bases. Edinburgh, Scotland, 1999, 99(6): 518-529

[6] Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(1): 117-128

[7] Wang Jianfeng, Wang Jingdong, Song Jingkuan, et al. Optimized Cartesian k means. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(1): 180-192

[8] Hajebi K, Abbasi-Yadkori Y, Shahbazi H, et al. Fast approximate nearest-neighbor search with k-nearest neighbor graph//Proceedings of the International Joint Conference on Artificial Intelligence. Barcelona, Spain, 2011, 22(1): 1312

[9] Malkov Y, Ponomarenko A, Logvinov A, et al. Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems, 2014, 45: 61-68

[10] Dong W, Moses C, Li K. Efficient k-nearest neighbor graph construction for generic similarity measures//Proceedings of the 20th International Conference on World Wide Web. Hyderabad, India, 2011: 577-586

[11] Tao Yufei, Yi Ke, Sheng Cheng, et al. Quality and efficiency in high dimensional nearest neighbor search//Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. Rhode Island, USA, 2009: 563-576

[12] Gan Junhao, Feng Jianlin, Fang Qiong, et al. Locality-sensitive hashing scheme based on dynamic collision counting//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Arizona, USA, 2012: 541-552

[13] Sun Y, Wang W, Qin J, et al. SRS: Solving c-approximate nearest neighbor queries in high dimensional Euclidean space

with a tiny index. Proceedings of the VLDB Endowment, 2014, 8(1): 1-12

[14] Huang Q, Feng J, Zhang Y, et al. Query-aware locality-sensitive hashing for approximate nearest neighbor search. Proceedings of the VLDB Endowment, 2015, 9(1): 1-12

[15] Liu Yingfan, Cui Jiangtao, Huang Zi, et al. SK-LSH: An efficient index structure for approximate nearest neighbor search. Proceedings of the VLDB Endowment, 2014, 7(9): 745-756

[16] Gaede V, Günther O. Multidimensional access methods. ACM Computing Surveys, 1998, 30(2): 170-231

[17] Panigrahy R. Entropy based nearest neighbor search in high dimensions//Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms. Miami, USA, 2005: 1186-1195

[18] Lv Q, Josephson W, Wang Z, et al. Multi-probe LSH: Efficient indexing for high-dimensional similarity search//Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna, Austria, 2007: 950-961

[19] Tang B, Yiu M L, Hua K A. Exploit every bit: Effective caching for high-dimensional nearest neighbor search. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(5): 1175-1188

[20] Asano T, Ranjan D, Roos T, et al. Space-filling curves and their use in the design of geometric data structures. Theoretical Computer Science, 1997, 181(1): 3-15

[21] Liao S, Lopez M A, Leutenegger S T. High dimensional similarity search with space filling curves//Proceedings of the 17th International Conference on Data Engineering. Heidelberg, Germany, 2001: 615-622

[22] Valle E, Cord M, Philipp-Foliguet S. High-dimensional descriptor indexing for large multimedia databases//Proceedings of the 17th ACM Conference on Information and Knowledge Management. Napa Valley, USA, 2008: 739-748

[23] Amsaleg L, Chelly O, Furon T, et al. Estimating local intrinsic dimensionality//Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Sydney, Australia, 2015: 29-38

[24] Skilling J. Programming the Hilbert curve///Proceedings of the 23rd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering. Jackson Hole, USA, 2004, 707(1): 381-387

FENG Xiao-Kang, Ph. D. candidate. His research interests include high-dimensional indexing and approximate query processing.

PENG Yan-Guo, Ph. D., lecturer. His research interests include secure issues in data management, privacy protection and cloud security.

CUI Jiang-Tao, Ph. D., professor. His main research interests include data and knowledge engineering, large-scale data management, data security and privacy protection.

LIU Ying-Fan, Ph. D. candidate. His main research interests include management of large-scale complex data and similarity search for high-dimensional data.

LI Hui, Ph. D., professor. His research interests include social computing, knowledge discovery, graph mining and data privacy in big data.

Background

Nearest neighbor (NN) search in high-dimensional space is a fundamental paradigm in a wide range of applications, such as text information retrieval, search engine, content-based information query, duplication detection, etc. In these areas, data are usually modeled as high-dimensional features. On the one hand, due to the notorious "curse of dimensionality", exact NNs are usually infeasible to achieve. A lot of research efforts have been devoted to finding approximate nearest neighbors (ANN) which are close enough to the query to achieve a satisfying trade-off between accuracy and efficiency. On the other hand, large scale feature datasets are often too massive to fit into the computer's internal memory, external memory (usually disks) becomes a reasonable choice. However, due to the huge speed gap between internal memory and external memory, the resulting input/output communication becomes very expensive, too much NN candidates or improper loading manner will make the NN candidates loading the most time-consuming part of the entire NN search, called the I/O performance bottleneck.

Locality-sensitive hashing (LSH) is an attractive approach for disk-based ANN search as it enables fast and accurate irrelevant points filtration and offers c-ANN results at a sub-linear time complexity. Plenty of LSH based methods have been developed to further boost the ANN performance.

However, most of them access candidate objects through significant random I/O operations which tend to incur the I/O performance bottleneck.

In this work, the authors propose a novel LSH index O2LSH to address the above issues. They conduct a thorough quantitative analysis on typical space-filling curves and find that NFT curves can produce better local distribution of NN candidate points than DFT curves. By comparison, they finally choose a best NFT curve as the linear order to maintain as many as NN candidates within same local disk pages, which not only enhances the I/O efficiency but also improves the ANN search accuracy. Empirical experiments demonstrate the superior accuracy and I/O efficiency of O2LSH in ANN search, compared with state-of-the-art methods. Besides, O2LSH eliminates the sensitiveness to a key parameter of LSH function, which further enhances the algorithm practicality.

This research is supported by the National Natural Science Foundation of China under Grant Nos. 61976168, 61702403, 61672408 and 61972309, the China 111 Project under Grant No. B16037, Director Fund project of the National Engineering Laboratory for Public Security Risk Perception and Control by Big Data, CCF-Huawei Database Innovation Research Program under Grant No. CCF-HuaweiDBIR008B, China Postdoctoral Science Foundation No. 2018M633473, the Natural Science Basic Research Plan in Shaanxi Province of China under Grant Nos. 2015JQ6227, 2018JM6073 and 2019CGXNG-023, the Jiangxi Key Research and Development Plan under Grant No. 20181ACE50029, the Fundamental Research Funds for the Central Universities under Grant Nos. XJS190305 and JB181505, and the Innovation Fund of Xidian University.