基于导航扩展图的高效近似最近邻搜索

摘要

近似最近邻搜索（ANNS）是数据库和数据挖掘中的一个基本问题。可扩展的 ANNS 算法应该既高效又快速。一些早期的基于图的方法在搜索时间复杂度上表现出了有吸引力的理论保证，但它们都面临着高索引时间复杂度的问题。最近，一些基于图的方法被提出来通过近似传统图来降低索引复杂度；这些方法在百万级数据集上取得了革命性的性能。然而，它们仍然无法扩展到十亿节点的数据库。在本文中，为了进一步提高基于图的方法的搜索效率和可扩展性，我们首先介绍四个方面：(1) 确保图的连通性；(2) 降低图的平均出度以实现快速遍历；(3) 缩短搜索路径；(4) 减小索引大小。然后，我们提出了一种称为单调相对邻域图（MRNG）的新颖图结构，它保证了非常低的搜索复杂度（接近对数时间）。为了进一步降低索引复杂度并使其适用于十亿节点的 ANNS 问题，我们通过近似 MRNG 提出了一种名为导航展开图（NSG）的新颖图结构。NSG 同时考虑这四个方面。大量实验表明 NSG 显着优于所有现有算法。此外，NSG 在淘宝（阿里巴巴集团）的电商搜索场景中表现出优越的性能，并已以十亿节点规模集成到其搜索引擎中。

关键词： 近似最近邻搜索；图

1 引言

近似最近邻搜索（ANNS）几十年来一直是一个热门话题，为数据挖掘、数据库和信息检索中的许多应用提供了基础支持 [2, 6, 7]。对于稀疏离散数据（如文档），可以在高级索引结构（例如倒排索引）上有效地执行最近邻搜索。对于密集连续向量，已经提出了各种解决方案，例如基于树结构的方法 [2, 4, 5, 10]、基于散列的方法 [11, 13]、基于量化的方法 [1, 12] 和基于图的方法 [3, 18, 19]。其中，基于图的方法最近显示出巨大的潜力。一些实验结果表明，基于图的方法在常用的欧几里德空间 [2, 9, 15, 18] 中比其他类型的一些流行算法表现得更好。原因可能是这些方法不能像基于图的方法一样表达邻居关系，并且它们倾向于比基于图的方法检查邻居子空间中更多的点以达到相同的精度。因此，它们的搜索时间复杂度涉及维度上的指数因子，并导致性能较差。通过图进行最近邻搜索的研究已经有数十年了。给定 d 维欧几里德空间中的一组点，图被定义为连接这些点（节点）的一组边。边定义节点之间的邻居关系。在边上提出了各种约束，以使图适合 ANNS 问题。这些图现在被称为邻近图。一些邻近图，如 Delaunay 图和单调搜索网络（MSNET），确保从任何节点到另一个节点存在一条路径，其中中间节点越来越接近目标节点。然而，没有给出找到这样一条路径所需的计算复杂度。其他工作如随机邻域图保证多对数搜索时间复杂度。根据经验，贪婪路由路径的平均长度随着可导航小世界网络上的数据大小呈多对数增长。然而，构建这些图的时间复杂度非常

高（至少 O(n²)），这对于大规模问题来说是不切实际的。最近的一些基于图的方法试图通过设计图的近似值来解决这个问题。例如，GNNS [14]、IEH [16] 和 Efanna [9] 基于 kNN 图，它是 Delaunay 图的近似。NSW [18] 近似于 NSWN，FANNG [15] 近似相对邻域图，并且提出了分层 NSW，以利用 Delaunay 图、NSWN 和 RNG 的属性。此外，HNSW [17] 中使用分层结构来实现图不同层上的多尺度跳跃。这些近似主要基于直觉，一般缺乏严格的理论支持。在我们的实验研究中，我们发现它们对于当今需求量很大的十亿节点应用程序来说仍然不够强大。为了进一步提高基于图的方法的搜索效率和可扩展性，我们从如何在图上执行 ANNS 开始。尽管图索引多种多样，但几乎所有基于图的方法都共享相同的贪婪最佳优先搜索算法。该算法尝试通过贪婪过程到达查询节点（如算法1）。对于给定的查询，我们需要从数据集中检索其最近邻。给定一个起始节点，我们沿着出边到达邻居，并将它们与查询进行比较以选择一个继续。选择原则是最小化到查询的距离，新的迭代从所选节点开始。我们可以看到，改进基于图的搜索的关键是缩短算法形成的搜索路径并减少图的出度（即减少每个节点的选择数量）。为了改进基于图的搜索，我们需要：（1）确保图的连接性，以确保查询（或查询的最近邻居）是可达的；(2) 降低图的平均出度；(3) 缩短搜索路径以降低搜索时间复杂度；(4) 减少索引大小（内存使用）以提高可扩展性。一些方法使用散列、随机 KD 树和多层图来加速搜索。然而，这些可能会导致海量数据库占用大量内存。我们的目标是减少索引大小并同时保持搜索效率。在本文中，我们提出了一种新的图结构，称为单调相对邻域图（MRNG），它保证了较低的平均搜索时间复杂度（非常接近对数时间复杂度）。为了进一步降低索引复杂度，我们提出了导航展开图（NSG），它是 MRNG 的一个很好的近似，其继承了较低的搜索复杂度，并兼顾了四个方面。文章的贡献如下：

1. 首先对 MSNET 图族在近似最近邻搜索（ANNS）中具有吸引力的属性进行了全面的理论分析。在此基础上，我们提出了一种新颖的图结构，称为单调相对邻域图（MRNG），它确保了接近对数的搜索复杂度。

2. 为了进一步提高基于图的 ANNS 方法的效率和可扩展性，我们考虑了图的四个方面：确保连通性、降低平均出度、缩短搜索路径和减小索引大小。受这些启发，我们设计了一种近似的 MRNG，称为导航展开图（NSG），以同时解决这四个方面的问题。与 MRNG 相比，NSG 显著降低了索引复杂度，对于大规模问题非常实用。

3. 大量实验表明，我们的方法在搜索性能方面优于最先进的方法，并且在基于图的方法中内存使用量最小。此外，NSG 算法还在淘宝（阿里巴巴集团）的电商搜索场景上进行了测试。该算法已集成到他们的搜索引擎中，用于十亿节点的搜索。

图搜索算法

2 相关工作

我们用 E^d 表示 l_2 范数下的欧几里得空间。任意两个点 p, q 之间的接近程度定义为它们之间的 l_2 距离 $\delta(p, q)$

Algorithm 1 图搜索 (G, p, q, l)

需要： 图 G，起始节点 p，查询点 q，候选池大小 l
确保： q 的 k 个最近邻

```
i ← 0, 候选池 S ← ∅
S.add(p)
while i < l do
    i ← S 中第一个未检查节点的索引
    将 p_i 标记为已检查
    for 每个邻居 n of p_i in G do
        S.add(n)
    end for
    对 S 根据到 q 的距离进行升序排序
    if S.size() > l then
        S.resize(l)
    end if
end while
返回 S 中的前 k 个节点 =0
```

2.1 问题设定

在信息检索和高维数据管理的各种应用中，可以将高维空间中的最近邻搜索问题抽象化为最近邻搜索问题。最近邻搜索（NNS）问题定义如下 [13]：**定义 1（最近邻搜索）**：给定一个有限点集 S，包含 n 个在空间 E^d 中的点，预处理 S 以有效地返回一个点 $p \in S$，该点与给定查询点 q 最近。这自然推广到 K 最近邻搜索，当我们要求算法返回 K 个与查询点最近的点（$K > 1$）时。最近邻搜索问题的近似版本（ANNS）可以定义如下 [13]：**定义 2（ϵ-最近邻搜索）**：给定一个有限点集 S，包含 n 个在空间 E^d 中的点，预处理 S 以有效地回答查询，返回一个点 $p \in S$，使得 $\delta(p,q) \leq (1+\epsilon)\delta(r,q)$，其中 r 是 S 中与 q 最近的邻居。类似地，当我们要求算法返回 K 个点（$K > 1$），使得 $\forall i = 1, \ldots, K, \delta(p_i, q) \leq (1+\epsilon)\delta(r,q)$ 时，这个问题可以推广到近似 K 最近邻搜索（AKNNS）。由于精确最近邻搜索的内在困难，大多数研究者转向 AKNN。主要动机是在较短的搜索时间内牺牲一点准确性。为了方便建模和评估，我们并不计算 ϵ 的确切值，而是使用另一个指示近似程度的指标是：精度（precision）。假设由给定查询 q 的 AKNNS 算法返回的点集为 R'，而 q 的正确 k 最近邻集为 R，则精度（准确性）定义如下 [15]：

$$\text{precision}(R') = \frac{|R' \cap R|}{|R'|} = \frac{|R' \cap R|}{K}. \tag{1}$$

更高的精度对应于更小的 ϵ，因此，近似程度更高。在本文中，我们使用精度作为评估指标。

2.2 非基于图的 ANNS 算法

非基于图的 ANNS 方法大致包括基于树的方法、基于散列的方法和基于量化的方法。一些众所周知且广泛使用的算法，如 KD 树、R* 树、VA 文件、局部敏感哈希 (LSH) 和乘积量化 (PQ) 都属于以上类别。一些工作侧重于改进算法，而另一些工作则侧重于根据不同平台和场景优化现有方法。在最近一些作品的实验研究中，基于图的方法显着优于一些众所周知

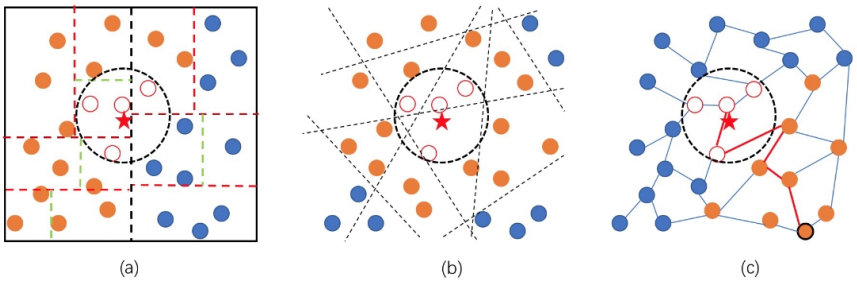

图 1. (a) 是树索引，(b) 是哈希索引，(c) 是图索引。红色星号是查询（不包括在基础数据中）。四个红色圆环是它的最近邻居。树索引和哈希索引将空间划分为多个单元格。每个单元格最多包含三个点。图索引中每个节点的出度也最多为三个。为了检索查询的最近邻居，我们需要回溯并检查树索引的许多叶子节点。我们需要检查哈明半径为 2 的附近桶，以获得哈希索引。至于图索引，如红线所示形成了一个搜索路径。橙色圆圈是在搜索过程中检查的点。基于图的算法需要最少的距离计算次数。

的非基于图的方法（例如，KD-tree，LSH，PQ）。这可能是因为非基于图的方法都试图通过划分空间并对结果子空间进行索引以实现快速检索来解决 ANNS 问题。不幸的是，对子空间进行索引以便有效地扫描相邻区域以定位给定查询的最近邻居并不容易。以图 1为例（图中不包含 PQ 算法，因为从某种角度来看它可以被视为一种哈希方法）。非基于图的方法需要检查许多附近的单元格以实现高精度。检查大量远距离点，这个问题就变得很多随着维度的增加，情况会变得更糟（称为维度灾难）。基于图的方法可能从距离查询较远的位置开始，但它们通常会快速接近查询，因为所有这些方法都基于邻近图，邻近图通常可以更好地表达邻居关系。总之，为了达到相同的精度，非基于图的方法往往比基于图的方法检查更多的点。这将在我们后面的实验中得到证明。

2.3 基于图的 ANNS 算法

给定一个有限点集 S 在 E^d 中，图是一种由一组节点（表示点）和连接一些节点对的边组成的结构。一个节点 p 被称为 q 的邻居，当且仅当 p 和 q 之间存在一条边。基于图的 ANNS 通过上述图索引解决 ANNS 问题。算法 1 通常用于大多数基于图的方法。在过去的几十年中，许多图被设计用于高效的 ANNS。这里我们将介绍几种具有吸引理论性质的图结构。

德劳内图（Delaunay Graphs 或 Delaunay Triangulations）被定义为 Voronoi 图的对偶图。它被证明是单调搜索网络，但高维 ANNS 在德劳内图上的复杂性很高。根据 Harwood [15] 等人的研究，德劳内图在高维时迅速变得几乎完全连接。因此，搜索效率大大降低。GNNS 基于（近似）kNN 图，它是德劳内图的近似。IEH [16] 和 Efanna 也基于（近似）kNN 图。他们使用哈希和随机化 KD-树来为 kNN 图上的算法 1 提供更好的起始位置。尽管他们提高了性能，但仍受到大型和复杂索引的影响。

相对邻域图（RNG）最初并不是为 ANNS 问题设计的。然而，RNG 在 ANNS 中展现出了巨大的潜力。RNG 采用了一种有趣的边选择策略，以消除 S 中所有可能三角形的最长边。通过这种策略，RNG 降低了其平均出度一个常数 $C_d + o(1)$，该常数仅与维度 d 相关，且通常非常小。然而，依据 Dearholt [8] 等人的研究，RNG 并没有足够的边来构成单调搜索网络，因为缺乏严格的边选择策略。因此，对于路径长度没有理论保证。Dearholt 等人提出了一种方法，将边添加到 RNG 中，并将其转变为最小边数的单调搜索网络（MSNET）。该算法基

于预构建的 RNG，其索引复杂度为 $O(n^{\frac{2}{1+\epsilon}})$，在一般假设下。构建最小 MSNET 的预处理复杂度为 $O(n^2 \log n + n^3)$。

最小 MSNET 的总索引复杂度对于高维和大规模数据库来说是巨大的。最近的基于图的方法，如 FANNG [15] 和 HNSW [17]，采用了 RNG 的边选择策略，以减少其图的出度并提高搜索性能。然而，他们并没有提供理论分析。

可导航的小世界网络是适合 ANNS 问题的。节点及其邻居的度数均根据特定的概率分布分配。该图上搜索路径的长度随着网络规模的增长而以多项式对数形式增长，$O(A[\log N]^{\nu})$，其中 A 和 ν 是一些常数。这是一个经验估计，尚未被证明。因此，总的经验搜索复杂度为 $O(AD[\log N]^{\nu})$，D 是图的平均度数。图的度数需要仔细选择，这对搜索效率有很大影响。与其他传统图相比，构建这样一个图的时间复杂度约为 $O(n^2)$，这在大量问题中是不切实际的。Yury [18] 等人提出 NSW 图以同时逼近可导航的小世界网络和德劳内图。但他们很快发现图的度数过高，并且在他们的方法中也存在连通性问题。他们后来提出 HNSW 以解决此问题。具体而言，他们将多个 NSW 堆叠成一个层次结构以解决连通性问题。上层的节点通过概率分布进行采样，NSW 的大小从底层缩小到顶层。他们的直觉是，上层使得长距离捷径能够快速定位目标邻域。然后，他们使用 RNG 的边选择策略以减少图的度数。HNSW 是迄今为止最有效的 ANNS 算法，根据一些开源基准测试。

随机邻域图 [3] 是为高维空间中的 ANNS 问题设计的。它以随机方式构建。首先，他们围绕每个节点划分空间，并与一组凸锥相连，然后选择 $O(\log n)$ 个最接近的节点作为其邻居。他们证明了该图的搜索时间复杂度为 $O((\log n)^3)$，这非常有吸引力。然而，其索引复杂度过高。为了降低索引复杂度，他们提出了一种变体，称为 RNG*. RNG* 采用 RNG 的边选择策略，并使用附加结构（KD-树）来提高搜索性能。然而，其索引的时间复杂度仍然高达 $O(n^2)$ [3]。

3 本文方法

3.1 问题相关定义

3.1.1 图的单调性与路径长度

在图上进行 ANNS 的速度主要由两个因素决定：搜索路径的长度和图的平均出度。我们的目标是找到一个具有低出度和短搜索路径的图。我们将首先讨论如何设计一个具有非常短搜索路径的图。在介绍我们的提案之前，我们将首先提供对一类称为单调搜索网络（MSNET）的图的详细分析，该类图在 ANNS 中展现出了巨大的潜力。接下来我们将给出 MSNET 的定义。

3.1.2 定义与符号

给定一个点集 S 在 E^d 空间中，p, q 是 S 中的任意两个点。记 $B(p, r)$ 表示一个开放球体，使得 $B(p, r) = \{x | \delta(x, p) < r\}$。记 $\overrightarrow{pq}$ 表示从 p 到 q 的有向边。首先，我们给出图中单调路径的定义，如下所示：

定义 3（单调路径）。给定一个包含 n 个点的有限点集 S 在空间 E^d 中，p, q 是 S 中的任意两个点，G 是一个定义在 S 上的图。设 $v_1, v_2, \ldots, v_k$，其中 $(v_1, v_k = q)$ 表示在 G 中

从 p 到 q 的路径，即 $\forall i = 1, \ldots, k-1$, $v_i v_{i+1} \in G$。该路径是一个单调路径，如果 $\forall i = 1, \ldots, k-1$, $\delta(v_i, q) > \delta(v_{i+1}, q)$。

定义 4（单调搜索网络）。给定一个包含 n 个点的有限点集 S 在 E^d 空间中，定义在 S 上的图是一个单调搜索网络，当且仅当对于任意两个节点 $p, q \in S$ 存在至少一条从 p 到 q 的单调路径。

3.1.3 单调搜索网络的分析

单调搜索网络（MSNET）是一类能够保证任意两个节点之间存在单调路径的图。MSNET 本质上是强连通的，这确保了在沿单调路径行进时，我们始终朝着每一步的目标前进。在 MSNET 中，Dearholt [8] 等人假设可以使用算法1（通常称为图搜索算法）来确定从起点到目标的单调路径，即不需要回溯。回溯意味着当算法无法找到查询的最近邻（即局部最优）时，我们需要返回已访问的节点并寻找其他替代路径。MSNET 的单调性使得算法1 在图上的搜索行为几乎是确定的且可分析的。然而，Dearholt [8] 等人未能提供该属性的证明。在本节中，我们将给出该属性的具体证明。

定理 1。给定一个包含 n 个点的有限点集 S，在 E^d 空间中随机分布，并且构建在 S 上的单调搜索网络 G，在 G 中任意两个节点 p, q 之间的单调路径可以通过算法1 找到，且无需回溯。

根据定理 1，我们知道可以在给定的 MSNET 上使用算法1 找到查询 $q \in S$，而无需回溯。因此，迭代次数的期望与 MSNET 中单调路径的长度期望相同。在讨论 MSNET 中长度期望之前，我们先讨论长度期望的问题。

引理 1。给定一个图 G，其点集为 S，包含 n 个点在 E^d 中，G 是一个 MSNET 当且仅当对于任意两个节点 p, q，存在至少一条边 $\overrightarrow{pq}$ 使得 $r \in B(q, \delta(p, q))$。

根据引理 1，我们可以计算 MSNET 中单调路径的长度期望，如下所示。

定理 2。设 S 为一个有限点集，包含 n 个点，均匀分布在 E^d 的一个有限子空间中。假设包含 S 的最小凸壳的体积为 V_S。任意两个点之间的最大距离为 R。我们对 V_S 施加约束，当维度 d 固定时，$\exists k, k \leq V_S \geq V_B(R)$，其中 κ 是一个与 n 无关的常数，$V_B(R)$ 是半径为 R 的球体的体积。我们定义 $\Delta r = \min\{|\delta(a,b)| - \delta(a,c)|, |\delta(a,b) - \delta(b,c)|, |\delta(a,c) - \delta(b,c)|\}$，对于 S 上所有可能的非等边三角形 abc，Δr 是 n 的递减函数。

对于给定的 MSNET 定义在 S 上，从 p 到 q 的单调路径的长度期望，对于任意 $p, q \in S$，为 $O(n^{1/d} \log(n^{1/d})/\Delta r)$。

定理 2 是适用于各种 MSNET 的一般性质。函数 Δr 没有明确的表达式，因为它涉及随机性。我们观察到，在实践中，Δr 随着 n 的增加而非常缓慢地减少。在实验中，我们根据本文提出的图对不同公共数据集上的 Δr 函数进行估计。我们发现 Δr 主要受数据分布和数据密度的影响。由于 $O(n^{1/d})$ 随着 n 的增加而非常缓慢地增长，MSNET 中单调路径的长度期望 $O(n^{1/d} \log(n^{1/d})/\Delta r)$ 将具有非常接近 $O(\log n)$ 的增长率。这在我们的实验中也得到了验证。同样，我们可以看到单调路径的长度期望的增长率在维度 d 较高时较低。

在定理 2 中，对包含数据点的最小凸壳的体积的假设实际上是对数据分布的约束。我们尝试避免数据分布的特殊形状（例如，所有点都在一条直线上），以影响结论。举例来说，如果数据点均匀分布在一个直线上，单调路径的长度期望在这样的数据集上将几乎线性增长。

除了搜索路径的长度期望外，另一个影响搜索复杂性的关键因素是图的平均出度。一些

MSNET（如 Delaunay 图）的度数随着 n 的增加而增长。对于 MSNET，没有统一的几何描述。

Dearholt [8] 等人声称他们找到了一种构建具有最小出度的 MSNET 的方法。然而，他们的方法存在两个问题。首先，他们没有提供关于其 MSNET 随 n 的变化的度数分析。这主要是因为他们提出的 MSNET 是通过向 RNG 添加边来构建的，缺乏几何描述。其次，所提出的 MSNET 构建方法具有非常高的时间复杂度（至少 $O(n^{2-\frac{2}{d+\epsilon}}+n^2\log n+n^3)$），在实际的大规模场景中并不可行。下面我们将提出一种新类型的 MSNET，具有较低的索引复杂性和恒定的出度期望（与 n 无关）。简单来说，这种图的搜索复杂性与 n 的增长速率与单调路径的长度期望相同。

3.1.4 单调相对邻域图

在这一节中，我们描述一种新的图结构，称为 MRNG，它属于 MSNET 家族。为了使图变得稀疏，HNSW 和 FANNG 转向 RNG，但有研究证明 RNG 并没有足够的边成为一个 MSNET。因此，在 RNG 中没有理论上的搜索路径长度保证，且在 RNG 上的搜索可能会遭遇较长的绕行。

考虑以下示例。设 $lune_{pq}$ 表示一个区域，使得 $lune_{pq}=B(p,\delta(p,q))\cap B(q,\delta(p,q))$。给定一个有限点集 S，其包含 n 个点在 E^d 中，对于任意两个节点 $p,q\in S$，边 $pq\in RNG$ 当且仅当 $lune_{pq}\cap S=\emptyset$。

我们发现这个问题主要是由于 RNG 的边选择策略。Dearholt [8] 等人尝试向 RNG 添加边以产生具有最少边的 MSNET，但这种方法非常耗时。受到 RNG 的启发，我们提出了一种新的边选择策略来构建单调图。结果图可能不是最小 MSNET，但非常稀疏。基于新策略，我们提出了一种新图结构，称为单调相对邻域图（MRNG）。正式地，MRNG 可以定义如下：

定义 5（MRNG）。给定一个有限点集 S，包含 n 个点在 E^d 中，MRNG 是一个有向图，其边集满足以下属性：对于任意边 $\overrightarrow{pq}$，$p\overrightarrow{q}\in EMRNG$ 当且仅当 $lune_{pq}\cap S=\emptyset$ 或 $\forall r\in(lune_{pq}\cap S)$，$\overrightarrow{pr}\notin MRNG$。

我们在以下情况下避免歧义，当等腰三角形出现时。如果 $\delta(p,q)=\delta(p,r)$，且 qr 是三角形 pqr 中的最短边，我们根据预先定义的索引选择边，即，如果 $index(q)<index(r)$，我们选择 $\overrightarrow{pq}$。可以看出，MRNG 是以递归方式定义的。换句话说，定义 5 意味着对于任意节点 p，我们应该选择与其最近的邻居。

MRNG 的边选择策略与 RNG 的不同之处在于，对于任何边 $pq\in MRNG$，$lune_{pq}\cap S$ 不一定为空。这个差异可以在图 3 中清晰地看到。这里我们证明 MRNG 是一个 MSNET。

定理 3。给定一个有限点集 S，包含 n 个点。定义在 S 上的 MRNG 是一个 MSNET。

虽然在结构上有所不同，但 MRNG 和 RNG 共享一些公共边。我们将首先定义最近邻图（NNG）如下：

定义 6（NNG）。给定一个有限点集 S，包含 n 个点在 E^d 中，NNG 是边的集合，对于任何边 $\overrightarrow{pq}$，$\overrightarrow{pq}\in NNG$ 当且仅当 q 是 p 在 S 中的最近邻。同样，我们可以通过为每个节点分配一个唯一的索引，并将节点链接到其最近邻居（具有最小索引）来消除 NNG 中的歧义。显然，我们有 $MRNG\supset RNG$（如果节点 q 是节点 p 的最近邻，则 $lune_{pq}\cap S=\emptyset$）。这是 MRNG 单调性的必要条件。图 4 显示了一个示例，说明当我们应用 MRNG 的边选择策略时，在某个图 G 上的非单调路径。图 4 中的边满足 MRNG 的选择策略，除了 q 被强制不与其最

近邻 t 连接。因为 t 是 q 的最近邻，所以 $\delta(q,r) > \delta(q,t)$。因为 qt 是三角形 qtr 中的最短边，而 q,r 是连接的，因此 rt 必须是根据边选择策略选择的三角形 qtr 中的最长边。因此，r,t 不能连接，我们只能通过节点 s 到达 t。类似地，当 r 是三角形 rst 中的最长边时，边 rs 可以共存于此图中。因此，当我们从 p 移动到 t 时，需要至少绕行节点 r,s。因为 $\delta(q,r) > \delta(q,t)$，所以它是从 p 到 t 的非单调路径。如果我们不保证 $NNG \subset MRNG$，绕行是不可避免的，这在实践中可能更糟。很容易验证，如果我们在图 G 上执行 MRNG 的边选择策略，则类似的绕行问题也会出现，但不保证 $NNG \subset G$。

在这里我们将讨论 MRNG 的平均出度。MRNG 的边数比 RNG 多，但仍然非常稀疏，因为共享同一节点的两条边之间的夹角至少为 60°（根据 MRNG 的定义，对于任何两条边 $pq, pr \in MRNG$，$qr \notin MRNG$）。

引理 2。给定一个 E^d 中的 MRNG，MRNG 的最大度数是一个常数，与 n 无关。

现在根据引理 2、定理 1、定理 2 和定理 3,我们有搜索期望的复杂度为 $O(cn^{1/d}\log n^{1/d}/\Delta r)$，其中 c 是 MRNG 的平均出度，与 n 无关，Δr 是 n 的函数，随着 n 的增加而非常缓慢地减少。

3.2 NSG：一种 MRNG 的近似实践

3.2.1 MRNG

MRNG 可以通过在每个节点上应用我们的边选择策略来简单构建。具体来说，对于每个节点 p，我们将剩余节点集表示为 $R = S \setminus \{p\}$。我们计算 R 中每个节点与 p 之间的距离，然后按距离升序排列。我们将选定的节点集记为 L。我们将 R 中与 L 最近的节点添加到 L 中，以确保 $NNG \subset MRNG$。接下来，我们从 R 中获取一个节点 q，从 L 中获取一个节点 r，以检查边 pq 是否是三角形 pqr 中的最长边。如果 pq 不是三角形 pqr 中的最长边，则对于 L 中的每个节点 r，我们将 q 添加到 L。我们重复这个过程，直到检查完 R 中的所有节点。这个简单的构建过程运行时间为 $O(n^2\log n + n^2c)$，其中 c 是 MRNG 的平均出度，远小于文献中提出的 MSNET 索引方法的出度，至少为 $O(n^{2-\frac{2}{1+d+\epsilon}} + n^2\log n + n^3)$，在一般位置假设下。

3.3 NSG 的构建

尽管 MRNG 可以保证快速的搜索时间，但其高索引时间在大规模问题中仍然不够实用。我们将提出一种实际的方法，通过近似我们的 MRNG，并从四个标准出发来设计一个适合 ANNS 的良好图。我们将其命名为导航扩展图（NSG）。我们首先呈现 NSG 构建算法（算法 2）如下：

1. 我们使用当前最先进的方法构建一个近似的 kNN 图。
2. 我们找到数据集的近似重心。这可以通过以下步骤实现：（1）计算数据集的重心；（2）将重心视为查询，使用算法1 在 kNN 图上进行搜索，并将返回的最近邻作为近似重心。这个节点被命名为导航节点，因为所有搜索都将从这个固定节点开始。
3. 对于每个节点，我们生成一个候选邻居集，并从候选集中选择邻居。这可以通过以下步骤实现。对于给定节点 p，（1）我们将其视为查询，并从导航节点开始在预构建的 kNN

图上执行算法1。(2) 在搜索过程中，每个访问的节点 q（即计算 p 和 q 之间的距离）将被添加到候选集中（距离也被记录）。从候选集中选择最多 m 个邻居作为 p 的邻居，采用 MRNG 的边选择策略。

4. 我们在之前步骤中产生的图上展开深度优先搜索树。我们将导航节点视为根节点。当 DFS 结束时，有一些节点未与树连接，我们将它们与其近似最近邻（来自算法1）连接，并继续 DFS。

以下是 NSG 构建算法的动机。最终目标是构建一个低索引时间复杂度的 MRNG 的近似。

1. MRNG 确保在任意两个节点之间至少存在一条单调路径，但这并不是一项简单的任务。相反，我们只需选择一个节点并尝试保证从该节点到所有其他节点的单调路径的存在。我们将此节点命名为导航节点。当我们执行搜索时，我们始终从导航节点开始，这使得在 NSG 上的搜索几乎与在 MRNG 上的搜索一样高效。

2. MRNG 的边选择策略将其他节点视为当前节点的候选邻居，这导致了高复杂度。为了加速这个过程，我们希望为每个节点生成一个子集的候选节点。这些候选节点包含两个部分：(1) 如上所述，NNG 对单调性至关重要。因为获取确切的 NNG 是非常耗时的，所以我们返回近似的 kNN 图。高质量的近似 kNN 图通常包含高质量的近似 NNG。当只有少数节点未与其最近邻连接时，这是可以接受的。(2) 因为在 NSG 上的搜索始终从导航节点 p_n 开始，对于给定节点 p，我们只需考虑与 p 在搜索路径上的那些节点。因此，我们将 p 视为查询，并在预构建的 kNN 图上执行算法1。被搜索和访问的节点 p 的最近邻在近似 NNG 中被记录为候选节点。形成从导航节点到 p 的单调路径的节点在候选集中很可能被包含。当我们执行 MRNG 的边选择策略时，对于这些候选节点，NSG 很可能继承了从导航节点到 p 的 MRNG 中的单调路径。

3. 上述方法中可能存在的问题是某些节点的度爆炸问题。特别地，导航节点和密集区域中的节点将充当“交通枢纽”，并具有高出度。这个问题在 HNSW 中也有讨论。他们引入了一种多层图结构来解决这个问题，但他们的解决方案显著增加了内存使用量。我们的解决方案是将所有节点的出度限制为一个小值 $m \ll n$，而放弃较长的边。结果是由于边的消除，图的连通性不再得到保证。

为了应对连通性问题，我们引入了一种基于 DFS 跨越树的新方法，如上所述。在此过程中，所有节点都保证至少有一条从导航节点扩展出去的路径。尽管所提议的方法在最坏情况下会牺牲一些性能，但如果我们构建高质量的近似 kNN 图并选择适当的度限制 m，则 NSG 中的绕行将会最小化。

通过近似 MRNG，NSG 可以继承与 MRNG 相似的低搜索复杂度。同时，度上限使得图变得非常稀疏，而树跨越操作保证了 NSG 的连通性。NSG 的索引仅包含稀疏图和无辅助结构。我们的方法在与之前的工作相比的所有四个方面都有所进展。这些改进在我们的实验中也得到了验证。

3.3.1 NSG 的索引

NSG 的总索引复杂度包含两个部分，kNN 图构建的复杂度和 NSG 预处理步骤的复杂度。在百万规模的实验中，我们使用 nn-descent 算法构建近似 kNN 图。在 DEEP100M 实验中，我们使用 Faiss 在 GPU 上构建它，因为 nn-descent 在大型数据集上消耗的内存量急剧增加。我们关注于此节中算法 2 的复杂度。

NSG 的预处理步骤包括搜索-收集-选择操作和树跨越。由于 kNN 图是 MSNET 的近似，搜索复杂度大约为 $O(kn^{\frac{1}{d}}\log\frac{1}{\Delta r})$。对所有节点搜索，因此总复杂度大约为 $O(kn^{\frac{1+d}{d}}\log\frac{1}{\Delta r})$。边选择的复杂度为 $O(nlc)$，其中 l 是搜索生成的候选者数量，c 是我们为图设置的最大度数。由于 c 和 l 通常在实践中非常小（即 $c \ll n, l \ll n$），因此这个过程非常快。最后的过程是树跨越。这个过程非常快，因为强连通分量的数量通常小于我们需要添加到图中的边数。我们可以看到，最耗时的部分是“搜索-收集”部分。因此，这些过程的总复杂度大约为 $O(kn^{\frac{1+d}{d}}\log\frac{1}{\Delta r})$，这一点在后续的实验评估中得到了验证。

在本文的实现中，NSG 的整体经验索引复杂度为 $O(kn^{\frac{1+d}{d}}\log n^d + f(n))$，其中 $f(n) = n^{1.16}$ 与 nn-descent 相关，$f(n) = n\log n$ 与 Faiss 相关，这远低于 MRNG 的 $O(n^2\log n + cn^2)$。

3.3.2 NSG 的搜索

我们使用算法1 在 NSG 上进行搜索，并始终从导航节点开始搜索。由于 NSG 是对 MRNG 的精心设计的近似，NSG 上的搜索复杂度大约为 $O(cn^{\frac{1}{d}}\log n^{\frac{1}{d}}/\Delta r)$，其中 c 是 NSG 的最大度，d 是维度。在我们的实验中，Δr 大约为 $O(n^{-\frac{\epsilon}{d}})$，其中 $0 < \epsilon \ll d$。因此，经验平均搜索复杂度为 $O(cn^{\frac{1+\epsilon}{d}}\log n^{\frac{1}{d}})$。因为 $1 + \epsilon \ll d$，所以复杂度非常接近 $O(\log n)$，这一点在后续的实验评估中得到了验证。

4 复现细节

4.1 与已有开源代码对比

在本研究中，我们对 NSG 的源代码进行了实验环境的改写，旨在探讨在更大内存存储条件下，NSG 的性能表现是否会有所提升。随着数据规模的不断扩大，传统的内存管理方式可能无法满足高效处理的需求，因此优化实验环境以适应更大的内存空间显得尤为重要。我对原有的近似最近邻搜索算法进行了一系列优化，以充分利用非统一内存访问（NUMA）架构的特性，提高算法在多处理器系统上的性能。

4.1.1 数据分布优化

改进实现了一个简化版的 k-means 聚类算法，用于将数据点智能地分组到不同的 NUMA 节点。这种方法可以确保数据的局部性，减少跨 NUMA 节点的内存访问，从而降低访问延迟。具体实现如下：

4.1.2 NUMA 感知的数据加载

对数据加载函数进行了重构，使其具备 NUMA 感知能力。新的 `load_data_numa` 函数不仅负责数据的读取，还会根据聚类结果将数据分配到相应的 NUMA 节点。这种方法可以在数

Algorithm 2 k-means 聚类用于 NUMA 数据分布

```
0: function KMEANS_CLUSTERING(data, num_points, dim, k, max_iter)
0:     初始化 k 个随机中心点
0:     for iter = 1 to max_iter do
0:         for 每个数据点 do
0:             计算到每个中心点的距离
0:             分配到最近的中心点
0:         end for
0:         更新中心点位置
0:     end for
0:     return 数据点的聚类结果
0: end function=0
```

据加载阶段就优化内存布局，为后续的搜索操作奠定基础。

$$\text{load\_data\_numa} : \text{file} \rightarrow \{\text{data}_1, \text{data}_2, ..., \text{data}_n\} \tag{2}$$

其中，n 为 NUMA 节点数，data_i 表示分配到第 i 个 NUMA 节点的数据。

4.1.3 并行化搜索策略

在主循环中，我们采用了跨 NUMA 节点的并行搜索策略。对于每个查询 q，算法会同时在所有 NUMA 节点上执行搜索操作，然后合并各节点的结果：

$$\text{Result}(q) = \bigcup_{i=1}^{n} \text{Search}(q, \text{data}_i) \tag{3}$$

这种方法可以充分利用多处理器系统的计算资源，显著提高搜索效率。

4.1.4 灵活的 NUMA 配置

为了适应不同的硬件环境，我们在命令行参数中添加了 NUMA 节点数量的配置选项：

```
./program <data_file> <query_file> <nsg_path> <L> <K> <result_path> <numa_nodes>
```

这使得算法可以根据实际的硬件拓扑结构进行优化，提高了系统的可扩展性和适应性。

4.1.5 NUMA 感知的内存管理

实现了 NUMA 感知的内存释放函数 `free_data_numa`，确保了内存资源的正确回收：

$$\text{free\_data\_numa} : \{\text{data}_1, \text{data}_2, ..., \text{data}_n\} \rightarrow \emptyset \tag{4}$$

这不仅优化了资源利用，还有助于维护系统的长期稳定性。

4.2 实验环境搭建

本实验的硬件环境如表 1 所示。该环境配备了高性能的 INTEL(R) XEON(R) PLATINUM 8581C 处理器，具有 240 个 CPU 核心，总内存达 503 Gi。系统配置了 4 个 NUMA 节点为 NUMA 优化策略提供了理想的测试平台。

表 1. 实验环境详情

操作系统	
发行版	Ubuntu 22.04.5 LTS (Jammy Jellyfish)
版本	22.04
代号	jammy
处理器	
型号	INTEL(R) XEON(R) PLATINUM 8581C
架构	x86_64
CPU 数量	240
每个座的核数	60
座数	2
每个核的线程数	2
CPU 最大频率	4000.0000 MHz
CPU 最小频率	800.0000 MHz
内存	
总内存	503 Gi
已用内存	6.6 Gi
可用内存	493 Gi
交换空间	2.0 Gi
NUMA 配置	
NUMA 节点数	4
NUMA 节点 0 CPU	0-29,120-149
NUMA 节点 1 CPU	30-59,150-179
NUMA 节点 2 CPU	60-89,180-209
NUMA 节点 3 CPU	90-119,210-239
缓存	
L1d	5.6 MiB (120 实例)
L1i	3.8 MiB (120 实例)
L2	240 MiB (120 实例)
L3	600 MiB (2 实例)

4.3 实验结果分析

4.3.1 数据集

由于并非所有最新的最先进算法都能扩展到十亿点数据集，因此本实验是在四个百万规模的数据集上进行的。SIFT1M 和 GIST1M 是 BIGANN 数据集中广泛使用的相关文献中的两个数据集。

4.3.2 比较算法

我们采用了几种比较方法。首先，进行串行扫描以获取测试点的准确最近邻。基于树的方法中，Flann 是一个知名的 ANNS 库，基于随机化的 KD 树、K 均值树和复合树算法，而 Annoy 则基于二叉搜索森林。基于哈希的方法中，FALCONN 是一个知名的 ANNS 库，利用多探测局部敏感哈希。基于量化的方法，Faiss 最近由 Facebook 发布，包含了针对最先进 PQ 方法的良好实现代码，适用于 CPU 和 GPU。基于图的方法中，KGraph 基于 kNN 图，Efanna 则基于随机化 KD 树和 kNN 图的复合索引，FANNG 则基于文献中提出的某种图结构。HNSW 是一种层次图结构，DPG 则是从 kNN 图中选择边的无向图。根据开源基准，HNSW 是目前 CPU 上最快的 ANNS 算法。NSG 是本文提出的方法，仅包含一个带有导航节点的图，搜索始终从该节点开始。NSG-Naive 是一个设计基线，用于展示 NSG 的搜索 - 收集 - 选择操作的必要性以及图连接性的保证。由于没有导航节点，我们使用带有随机初始化的算法1 进行 NSG-Naive。从图 2和图 3中可以看出，NSG（Navigating Graph）算法在高精度

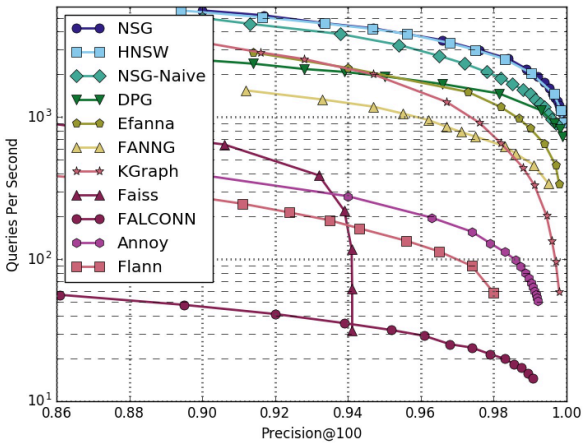

图 2. SIFT1M 所有算法的每秒查询比较

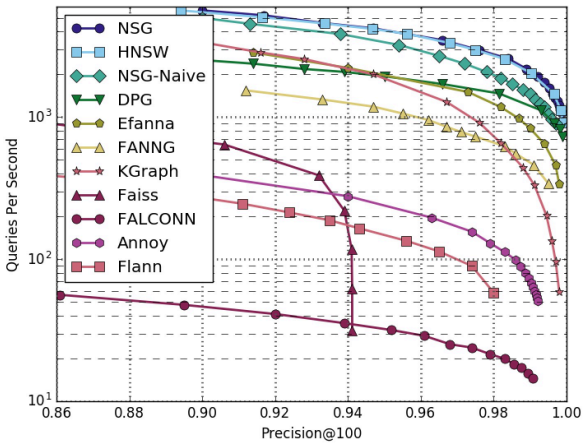

图 3. SIFT1M 所有图算法每秒查询比较

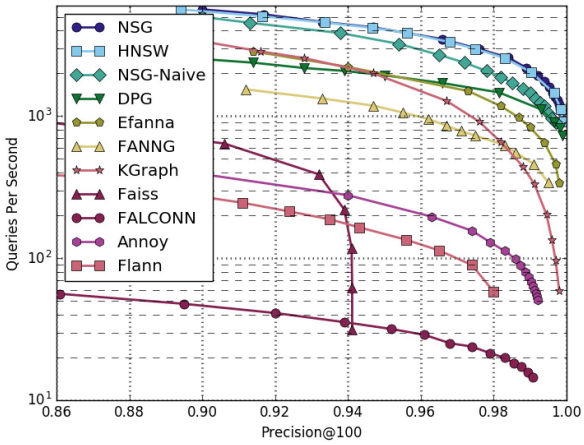

图 4. GIST1M 所有算法的每秒查询比较

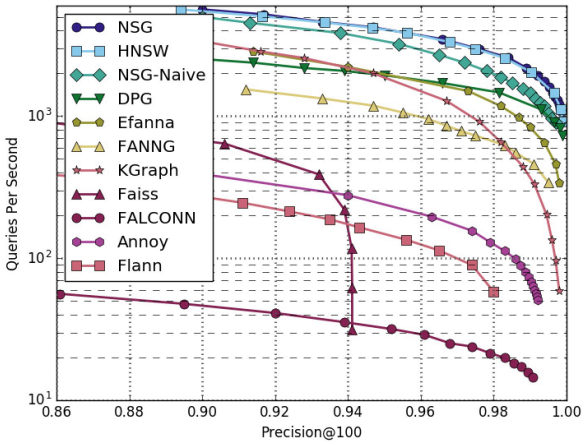

图 5. GIST1M 所有图算法每秒查询比较

区域（接近 1.0）表现出色，能够达到较高的查询速度，优于其他算法。HNSW（Hierarchical Navigable Small World）算法紧随其后，在精度和查询速度之间取得了良好的平衡。相较而言，NSG-Naive 和 DPG 的性能稍逊，但仍然保持了合理的查询速度。Efanna、FANNG 和 KGraph 等算法的表现也较为稳定，能够在一定精度下提供可接受的查询速度。然而，Faiss、

FALCONN、Annoy 和 Flann 的性能相对较低，尤其是在高精度要求下，查询速度显著下降。如图 4和图 5所示，GIST1M 数据集上不同算法的性能比较。

对于 numa 节点改进实验结果如图 5。

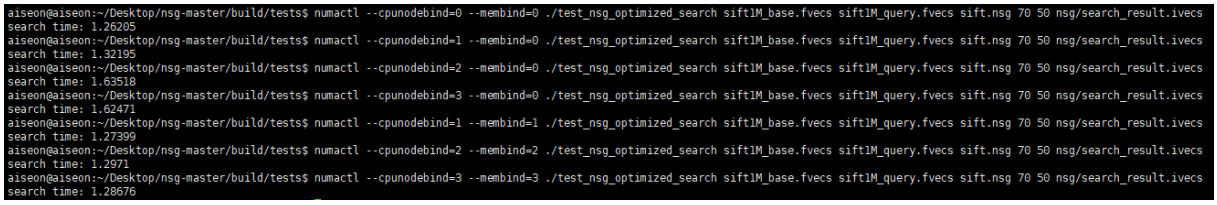

图 6. 实验结果示意

5 总结与展望

本文提出了一种新的单调搜索网络 MRNG，确保了近似对数级的搜索复杂度。作者提出了四个方面（确保连通性、降低平均出度、缩短搜索路径和减少索引大小）来设计更好的图结构以解决大规模问题。基于这四个方面，文章提出了 NSG，它是 MRNG 的一个实用近似，并同时考虑这四个方面。大量实验表明，NSG 在不同方面显著优于其他最先进的算法。此外，对源代码进行了改进，根据数据的近距离关系来分配到不同的 NUMA 节点，以优化访问性能。

参考文献

[1] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan. In *42nd International Conference on Very Large Data Bases*, volume 9, page 12, 2016.

[2] Akhil Arora, Sakshi Sinha, Piyush Kumar, and Arnab Bhattacharya. Hd-index: Pushing the scalability-accuracy boundary for approximate knn search in high-dimensional spaces. *arXiv preprint arXiv:1804.06829*, 2018.

[3] Sunil Arya and David M Mount. Approximate nearest neighbor queries in fixed dimensions. In *SODA*, volume 93, pages 271–280. Citeseer, 1993.

[4] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 322–331, 1990.

[5] JL Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):119–139, 1997.

[6] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, 2005.

[7] Arjen P de Vries, Nikos Mamoulis, Niels Nes, and Martin Kersten. Efficient k-nn search on vertically decomposed data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 322–333, 2002.

[8] DW Dearholt, N Gonzales, and G Kurup. Monotonic search networks for computer vision databases. In *Twenty-Second Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 548–553. IEEE, 1988.

[9] Cong Fu and Deng Cai. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. *arXiv preprint arXiv:1609.07228*, 2016.

[10] Keinosuke Fukunaga and Patrenahalli M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE transactions on computers*, 100(7):750–753, 1975.

[11] Jinyang Gao, Hosagrahar Visvesvaraya Jagadish, Wei Lu, and Beng Chin Ooi. Dsh: data sensitive hashing for high-dimensional k-nnsearch. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1127–1138, 2014.

[12] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2946–2953, 2013.

[13] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[14] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[15] Ben Harwood and Tom Drummond. Fanng: Fast approximate nearest neighbour graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5713–5722, 2016.

[16] Zhongming Jin, Debing Zhang, Yao Hu, Shiding Lin, Deng Cai, and Xiaofei He. Fast and accurate hashing via iterative nearest neighbors expansion. *IEEE transactions on cybernetics*, 44(11):2167–2177, 2014.

[17] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[18] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014.

[19] Yubao Wu, Ruoming Jin, and Xiang Zhang. Fast and unified local search for random walk based k-nearest-neighbor query in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of Data*, pages 1139–1150, 2014.