更快速的地理社交可达性查询

Rick van der Heijden 埃因霍温理工大学，埃因霍温，荷兰
r.h.j.v.d.heijden@student.tue.nl

Nikolay Yakovets 埃因霍温理工大学，埃因霍温，荷兰
hush@tue.nl

Thekla Hamm 埃因霍温理工大学，埃因霍温，荷兰
t.l.s.hamm@tue.nl

arXiv:2602.05928v1 [cs.DB] 5 Feb 2026

摘要

地理社交可达性查询（RangeReach）用于确定地理社交网络中的给定顶点是否能够到达查询区域内的任意空间顶点。最先进的 3DReach 方法通过区间标记对图可达性进行编码，并使用三维 R 树为空间顶点建立索引，以回答此类查询。我们提出了 2DReach，这是一种更简单的方法，完全避免了区间标记。与 3DReach 一样，2DReach 会将强连通分量（SCC）折叠为有向无环图，但它不计算区间标签，而是直接为每个分量存储一棵覆盖所有可达空间顶点的二维R树。因此，查询只需进行一次二维R树查找。我们还提出了压缩变体，通过排除空间汇点，以及在可达集合相同的分量之间共享R树，来减少存储开销。在四个真实世界数据集上的实验表明，2DReach 的索引构建速度快于 3DReach，且其压缩变体在所有方法中索引大小最小。与不同查询参数下的响应时间相比，2DReach 的查询性能具有竞争力或更优，且响应时间更加稳定。

关键词

地理社交可达性、空间索引、R树、基于位置的社交网络

1 引言

Yelp 和 Foursquare 等基于位置的社交网络（LBSN）将用户之间的社交互动与实体场所中的空间活动结合起来。在这些网络中，用户关注或结交其他用户，并在场所签到、撰写评论或评分。由此形成的图具有地理社交特征：顶点既代表用户，也代表场所；边捕捉社交和空间关系；部分顶点带有地理坐标。这类网络中的一项基础查询是地理社交可达性查询（RangeReach）[6]：给定一个查询顶点 u 和一个空间区域 R，判断 u 能否到达位于 R内的任意空间顶点。RangeReach 可应用于兴趣点推荐、地理广告和流行病学接触者追踪 [2]。

现有的 RangeReach 方法在问题的社交部分（图可达性）与空间部分（范围查询）之间面临权衡。GeoReach [6] 为每个顶点添加空间可达性摘要，并在查询时对图遍历进行剪枝，但当必须探索图中的大部分内容时，其性能会下降。最先进的 3DReach [2] 通过先将强连通分量（SCC）合并为超级顶点，再计算用于编码所得有向无环图中可达性的区间标签，从而消除了图遍历。空间顶点被索引在三维 R 树中，第三维表示用于基于区间的可达性测试的后序编号。尽管这种方法有效，但有两个缺点：区间标记的构建开销较大，而且三维 R 树需要更大的包围盒（每个节点六个浮点数，而非四个），范围查询也更复杂。

我们提出了 2DReach，通过完全避免区间标注来简化索引结构。与 3DReach 一样，2DReach 将强连通分量压缩为超级顶点，从而形成有向无环图。然而，2DReach 不计算区间标签，也不将空间数据提升至三维，而是直接为每个分量预先计算并存储一个二维R树，其中包含从该分量可达的所有空间顶点。因此，RangeReach 查询只需在查询顶点所属分量中进行一次二维R树查找。该设计以增加一些R树存储空间为代价，大幅简化了索引构建和查询执行。

我们进一步引入压缩变体，通过将空间汇点排除在强连通分量分解之外，并在具有相同可达集合的分量之间共享R树，从而减少存储。在四个真实世界位置社交网络数据集上的实验表明，2DReach的索引构建速度快于3DReach，其中压缩程度最高的变体在所有方法中产生最小的索引大小。2DReach提供具有竞争力或更优的查询性能，并且在不同查询参数下响应时间显著更加稳定。

2 预备知识

本节介绍理解地理社交可达性查询所需的必要记号。

地理社交图。 地理社交图 $G=(V,E,\delta)$是扩展了空间信息的有向图。V是顶点集合，$E\subseteq V\times V$是有向边集合。函数 δ: $V\to\mathbb{R}^2\cup\{\perp\}$将顶点映射到二维欧几里得点[1]，若顶点没有空间信息则映射为 $\perp$。最后，$P=\{v\in V\mid\delta(v)\neq\perp\}$表示空间顶点集合。

查询。 给定一个有向图 $G=(V,E)$ 和两个顶点$u,\ v\in V$，图可达性查询 $(u?\rightsquigarrow v)$用于判断u 是否能够到达 v，当且仅当存在一条从 u 到 v的路径。我们分别将其表示为$u\rightsquigarrow v$ 或 $u\not\rightsquigarrow v$。

定义 *2.1*（***RangeReach***）。给定一个地理社交图 $G=(V,E,\delta)$、一个查询顶点 u和一个轴对齐矩形 $R\subset\mathbb{R}^2$（空间区域）[2]，地理社交可达性查询（RangeReach）用于判断 u 是否能够通过地理社交路径到达区域 R；当且仅当存在一条从 u到某个顶点 v的路径，使得 $\delta(v)$ 位于 R中。形式化地：

$$\text{RangeReach}(G,u,R)=\begin{cases}TRUE & \text{if } \exists v\in V: u\rightsquigarrow v\wedge\delta(v)\in R\\ FALSE & \text{otherwise}\end{cases}$$

[1]不失一般性，本文假设空间为二维欧几里得空间；对于其他度量空间，解决方案也可以轻松扩展。[2]不失一般性，本文假设查询使用轴对齐矩形。不过，所提出的方法可以轻松扩展至使用其他类型的几何对象进行查询，例如多边形。

强连通分量分解。设 D $=(V_{\mathcal{D}}, E_{\mathcal{D}})$ 是图 G的强连通分量（SCC）分解。 G的每个强连通分量都由$V_{\mathcal{D}}$ 中的一个超顶点表示；如果G中某个强连通分量的顶点与另一个强连通分量的顶点之间存在有向边，则 $E_{\mathcal{D}}$ 中对应的两个超顶点之间也包含一条有向边。根据构造，D 是一个有向无环图（DAG）。

3 相关工作

Sun和Sarwat [6]提出了 RangeReach 查询，并提出 GeoReach，这是首个专门用于地理社交可达性的方法。GeoReach 通过 SPA-Graph，为图中的每个顶点添加预计算的空间可达性信息，并按精度递增将顶点分为三种类型：单个可达性位、可达性最小包围矩形，或一组层次网格单元。给定查询后，GeoReach 会遍历 SPA-Graph，并利用这些标注剪除无法满足空间谓词的路径。尽管效果显著，GeoReach 在查询时仍需进行图遍历；对于答案为否的查询，以及包含许多强连通分量的图，其性能会下降，因为可能需要探索 SPA-Graph 中很大一部分。

Bouros等人 [2] 提出了最先进的 3DReach 方法，该方法完全消除了图遍历。3DReach 首先将强连通分量合并为超级顶点，然后计算基于区间的标注 [1]，为每个分量分配一组后序区间，以编码其可达后代。每个空间顶点都映射为一个三维点（x，y，后序），并由三维 R 树建立索引。RangeReach 查询通过对查询顶点的每个区间标签分别执行一次三维范围查询来回答。其变体 3DReach-Rev 使用反向标号将空间顶点表示为垂直线段，从而将每个查询简化为一次三维范围查询，但代价是需要为更复杂的几何对象建立索引。然而，区间标记的构建成本高昂，而且与二维 R 树相比，三维 R 树需要更大的包围盒，查询也更复杂。

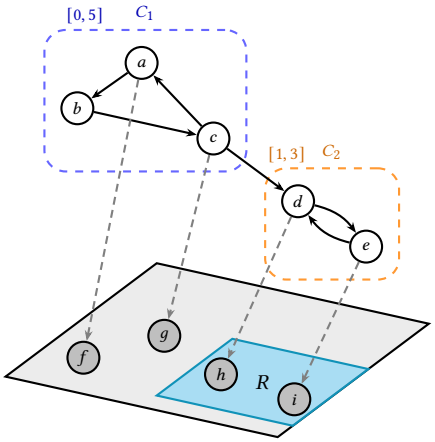
2DReach 与 3DReach 一样，具有在查询时避免图遍历的特性，但采用了更简单的方法。两种方法都会将强连通分量合并为超级顶点，以构成有向无环图。不过，3DReach 随后会计算区间标签（图 1(a)），并使用三维 R 树为各个空间顶点建立索引，因此查询时每个区间都需要执行一次三维范围查询。相比之下，2DReach 完全避免了区间标记：它直接预计算并存储每个分量对应的二维R树，该树涵盖所有可达空间顶点（图 1(b)）。这样，每次查询只需进行一次二维R树查找，且包围盒更小、范围谓词也更简单。

4 2DReach

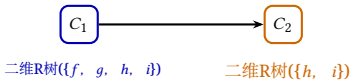
本节将介绍 2DReach。首先，我们解释基线方法，然后介绍内存压缩技术。

与 3DReach 类似，2DReach 会对输入图 G 进行强连通分量分解 D，以处理环。不过，2DReach 不会计算区间标签并将空间顶点索引到三维 R 树中，而是直接为 D 中的每个节点 n 存储一棵二维R树[3]，其中包含从 n可达的所有空间顶点。

[3]对于高维度量空间，可以使用其他空间索引，例如更高维的R树。



(a)



(b)

图1：运行示例。(a) 地理社交图 G，包含强连通分量C_1={a, b, c}和 C_2={d, e}；二维平面上的空间顶点（灰色）；查询区域 R （青色）。3DReach 使用区间标签（按分量显示）进行可达性编码。(b) 2DReach索引：强连通分量有向无环图，其中每个分量配有一棵二维R树，用于存储可达空间顶点。

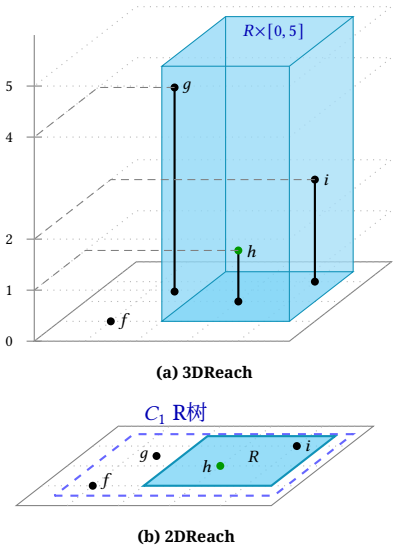
为了构建索引，2DReach 首先计算 D 中每个节点可达的空间顶点。随后，它按逆拓扑序遍历 D。遍历到某个节点时，将其直接子节点的可达空间顶点与该节点自身的可达空间顶点合并，从而得到该节点的可达空间顶点。最后，为每个节点的可达空间顶点构建一棵 R树。算法 1 概述了这一过程。

要查询这一结构，2DReach 以查询顶点 u和查询区域 R作为输入。首先，它会在 D 中确定包含 u的强连通分量 c。随后，2DReach 会查询属于 c的 R树，以确定其中是否包含查询区域 R 内的任何点。最后，它返回此查询的结果。第5节介绍的标准 2DReach 会为每个节点设置指向 c的 R树的指针，以缩短查询时间。

举例来说，考虑图1(a)中的地理社交图，其中强连通分量为 C_1 ={a, b, c}和 C_2 ={d, e}。按照拓扑逆序处理 D，2DReach 首先为 C_2在 {h, i} 上建立一棵 R树，然后为 C_1在 {f, g} ∪{h, i} 上建立一棵 R树（图1(b)）。为回答 RangeReach(G, a, R)，2DReach 将 a映射到 C_1，并使用区域 R查询其 R树，由于 $h \in R$，因此返回 TRUE。

4.1 压缩版2DReach

在讨论了基线 2DReach 后，本节将探讨减少 2DReach 空间占用的策略。首先请注意，基于位置的社交网络通常包含没有出边的空间节点。因此，为了减少空间需求，强连通分量分解



(a) 3DReach

(b) 2DReach

图 2：RangeReach（G，a，R）的查询执行。(a) 3DReach 使用区间 $[0,5]$将区域 R 扩展为三维长方体。(b) 2DReach 直接查询 C_1 的二维R树。两者都能找到$h \in R$（高亮显示），但 2DReach 避免了第三维。

算法 12DReach 构建

输入： 地理社交图 $G=(V,\ E,\ \delta)$，强连通分量分解 D
输出： 2DReach索引

令 S为从 D 的节点到 V的节点子集的映射。
令 R 为从 D 的节点到二维 R 树的映射。
对于所有节点 $n \in \mathcal{D}$ **执行**
将所有位于 n中的空间节点存储到 $S[n]$ 中。
结束循环
令 π 为 D 上节点的逆拓扑序。
对于 $n \in \pi$ **执行**
C ← 的子节点（n）。
$S[n] \leftarrow S[n] \cup \bigcup_{c \in C} S[c]$。
$R[n] \leftarrow$ 2上的 D-R 树。
结束循环
返回 R

仅在由非空间顶点诱导的子图（社交子图）上进行。因此，这一修改需要将算法 1 的第 4 行改为纳入节点 n 的所有空间直接邻居。在查询执行期间，2DReach 会验证每个空间顶点是否位于查询区域内；相应流程见算法 2。

这一扩展还可以进一步推广，以涵盖社交子图中任何没有可达空间顶点的强连通分量。此方法还假设空间顶点没有出边，尽管这并非数据模型中的严格约束。因此，在其他数据模型中，压缩过程应仅限于没有出边的空间顶点。因此，算法 2 的第 1 行也应检查 q 是否有出边。

此外，按拓扑序遍历时，父节点的可达集可能与其某个子节点的可达空间顶点集合相同。为了进一步压缩 2DReach索引，任何此类父节点都将与该子节点共用其R树。最后，2DReach 将R树指针绑定到图节点，以提升查询性能；第5节中的一种方法将探讨如何按照算法 2 中概述的方式，在强连通分量级别使用指针来压缩索引。

算法 22DReach压缩查询

要求： 2DReach索引 R，节点到强连通分量分解的映射节点 M
输入： 地理社交图 $G=(V,E,\delta)$，查询顶点 q，查询区域 R
输出： RangeReach 的布尔答案

如果 q 是空间顶点**则**
返回 $\delta(q) \in R$
否则
$c \leftarrow \mathrm{M}[q]$
返回 $\mathcal{R}[c][R]$
结束 if

4.2 复杂度分析

本节讨论所介绍的 2DReach 方法在理论上的最坏情况性能。不过，介绍 2DReach 时侧重于实际性能；因此，关于 2DReach 实际性能的实验评估，请参见第5节。

设 $G=(V,\ E,\ \delta)$ 为一个地理社交图，其空间顶点集合为 P，并设 $\mathrm{D}=(V_{\mathcal{D}},\ E_{\mathcal{D}})$ 为 G 的强连通分量分解。然后，我们将基数定义如下：$d=|V_{\mathcal{D}}|$、$e=|E_{\mathcal{D}}|$、$n=|V|$ $m=|E|$ $p=|P|$ 、和。

T定理 4.1. 存储 *2DReach* 索引需要 $O(d \cdot p)$ 空间。

证明。一个含有n个元素的二维R树占用 O(n) 空间。2DReach 为每个强连通分量节点存储一棵二维R树，每棵树最多有 p个节点。最后，每个强连通分量节点最多有一个指针。□

备注。 压缩版*2DReach* 的最坏情况空间复杂度仍然相同。

定理 4 2 2 $n+$。构建 *DReach* 索引需要$O(d \cdot p \cdot (\log(p)))$ 时间。

证明。在 S中存储空间节点，需要检查每个节点是否为空间节点，耗时 O(n)。此外，强连通分量分解的拓扑序计算需要线性时间 O($d+e$)。显然，$e \in \mathrm{O}(d^2)$。此外，对于 D 中的每个节点，计算 S 可能需要进行 O(d) 个子节点对应的 O(p) 次集合插入操作。最后，为 D 中的每个节点构建一棵二维R树；由于每棵树最多包含 p个元素，因此需要 O($p \cdot \log(p)$) 时间。□

备注。 在压缩版*2DReach*的最坏情况时间复杂度中，如果所有空间顶点都是汇点，则可以用 $d-p$ 替换 d。但是，如果不作此假设，最坏情况时间复杂度则不会改变。

定理 4.3。 *2DReach*索引的查询时间为 $O(p)$。

证明。等价于 R树查询时间。 □

注释。 对于包含 n 个元素的 R树，其平均查询时间为 $O(\log_M(n))$，其中 M 为每页最大条目数。随后，*2DReach* 的平均查询时间也等价。

5 实验

本节讨论2DReach的实验评估。首先讨论实验设置，随后对实验结果进行评估。所有实验均在一台配备64GB内存、主频为3.80 GHz的Intel(R) Core(TM) Ultra 9 275HX CPU以及Ubuntu 24.04 LTS的笔记本电脑上完成。此外，代码使用C++17编写，并使用启用了-O3优化标志的gcc（v13.3.0）编译，可在GitHub[4]上获取。

5.1 实验设置

在实验中，我们使用了四个真实世界数据集：Yelp[5]，Foursquare [3, 5][6]，Gowalla [4][7]，和Weeplaces[7]。这些数据集是基于位置的社交网络（LBSNs），由用户（不含空间信息的节点）的社交图和场所（含空间信息的节点）组成。此外，对于这类图，场所没有出边，因为它们只能被访问、签到或评价。表2给出了有关这些数据集的统计信息。这些数据集也被用于3DReach评估 [2]。

在实验中，我们评估以下方法：

- 3DReach：来自 [2] 的最先进的 3DReach 索引。我们使用作者提供的源代码[8]。
- 3DReach-Rev：一种 3DReach 变体，使用反向区间标记将空间顶点表示为垂直线段，从而将每个查询简化为单个三维范围查询 [2]。我们使用作者提供的源代码[8]。
- 2DReach：第 4 节所述的标准 2DReach，不含压缩机制。- 2DReach-Comp：第 4.1 节所述的 2DReach，采用了各种压缩机制。
- 2DReach-Pointer：在 2DReach-Comp 的基础上，强连通分量分解节点的指针仅为非空间分量存储，而非为节点存储，并使用位向量检索正确的 R树。

我们针对不同参数开展了一系列实验，以评估这些方法的性能。对于每个参数的每个取值，执行 1000 次查询。每项实验重复十次，并将中位数作为最终结果。实验采用的不同参数如下：

区域范围比 区域范围比是查询区域大小占整个空间区域的百分比。

顶点度 顶点度是查询顶点在指定范围内的出度。

空间选择性比率 空间选择性比率是指查询区域内的空间节点数占地理社交图中节点总数的百分比。

[4]https://github.com/RickHeijden/rangereach [5]www.yelp.com/dataset [6] https://academictorrents.com/details/b24c73949308b3f6bdd8fea1a485534392eef338[7] Gowalla 与 Weeplaces 数据集是从 3DReach 的下载网址下载的 [2]: https://seafile.rlp.net/d/6fa3703e57234b6d9831/。[8] https://github.com/pbour/rangereach

表1列出了各参数在实验中采用的取值。除空间选择性比率外，其他参数的默认值均在针对其他参数进行实验时予以涵盖。进行其他参数的实验时，空间选择性比率可以取任意值。我们遵循 3DReach [2] 确立的实验方法，以便进行直接比较。

5.2 评估

本节根据所述实验评估不同方法。首先讨论初始化过程和存储开销。然后评估不同方法之间的查询性能。

5.2.1 初始化与存储开销。 所有方法都需要一个初始化阶段，以便为地理社交图构建索引并将其存储以供查询执行。每种方法都需要进行强连通分量分解。此外，3DReach 和 3DReach-Rev 方法还会创建区间标注方案，其中前者会存储该方案。另外，2DReach 方法需要构建并存储从节点到 R树的指针。此外，2DReach-Pointer 还会存储额外的位向量。最后，所有方法都需要构建并存储其 R树。

表 3 展示了各方法初始化阶段所需的时间（秒）。结果表明，所有 2DReach 方法的构建时间都优于 3DReach 和 3DReach-Rev。3DReach 和 3DReach-Rev 的构建时间主要受其标注过程影响，这也造成了它们与其他方法之间明显的差异。

表4 概述了索引大小。结果表明，标准 2DReach 的存储开销最多仅增加 40%。然而，当社交复杂度显著提高时（Yelp 和 Foursquare），该方法的竞争力会减弱。这是因为社交SCC数量增加，需要使用更复杂的二维R树。2DReach-Pointer 在所有情况下都有显著改进，因为标准 2DReach 的存储开销会随着社交组件的增加而上升，而非空间组件的数量则远少于节点数。

随着空间节点数量的增加，2DReach-Comp 相对于 2DReach（Gowalla 和 Weeplaces）有所改进。这是由于空间汇点所需的 R 树数量减少。表2进一步表明，复用子节点的 R 树效果甚微，最多只能减少两个。然而，在 Weeplaces 等低SCC情况下，这会产生更显著的影响。最后，由于压缩和指针减少，2DReach-Pointer 在存储开销方面继续优于所有其他方法。

值得注意的是，尽管存储了多棵R树，2DReach-Pointer 的索引大小仍小于两种 3DReach 变体。这是因为 3DReach 必须为所有超顶点存储区间标签，在大型图中其数量可达数百万个区间。相比之下，2DReach 直接利用社交SCC成员关系，放弃为空间汇点构建二维R树结构，并完全避免区间标记。此外，2DReach-Pointer 在可能时在父节点和子节点之间共享R树，且其二维R树节点所需的包围盒更小（4个浮点数），而三维R树节点需要6个浮点数。R树共享、将空间汇点排除在分解之外以及消除区间标签的组合，解释了为何

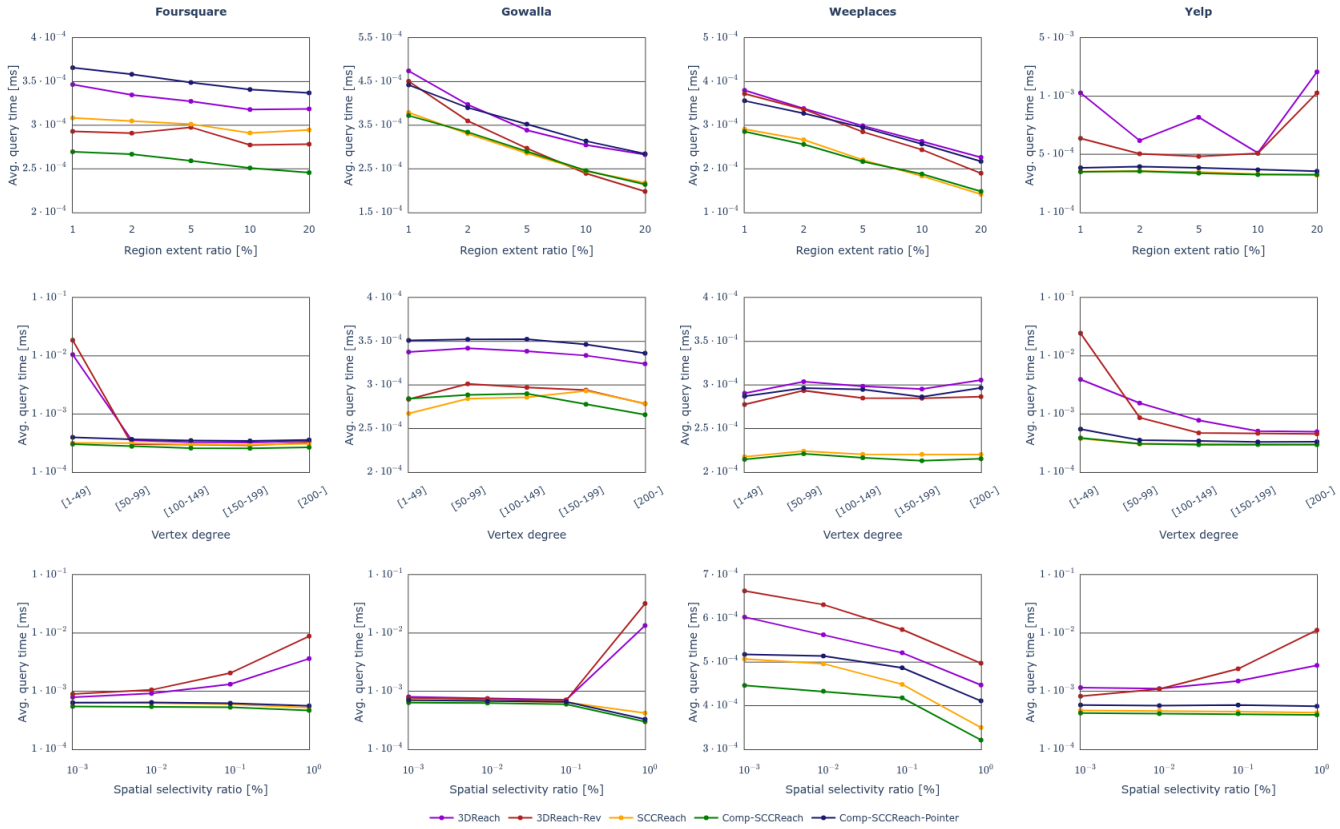


图3：以折线图展示的实验结果。各列分别代表四个数据集，各行分别代表三个实验参数。Yelp（全部）、Foursquare（顶点度数和空间选择性）以及 Gowalla（空间选择性）的结果采用对数坐标轴显示；其他所有图均采用线性坐标轴。

参数	（实验）取值	默认值
区域范围比	1%，2%，5%，10%，20%	5%
顶点度	[1 − 49]，[50 − 99]，[100 − 149]，[150 − 199]，[200−]	[100 − 149]
空间选择性比率	0.001% 0.01% 0.1%，1%	N/A

表1：三个参数 fo　　用于评估实验的 r。对每个参数，均采用（实验）取值执行实验。
此外，在其他实验中，　　采用默认值。

数据集	用户	场所	节点	边	SCCs	用户强连通分量	不同的 R 树
Foursquare	2,119,987（65.2%）	1,132,617（34.8%）	3,252,604	19,685,786	1,400,154	267,537（19.1%）	267,535
Gowalla	407,533 (13.0%)	2,723,102 (87.0%)	3,130,635	23,778,362	2,723,103	1 (0.0%)	1
Weeplaces	16,022 (1.6%)	971,309 (98.4%)	987,331	2,758,946	971,311	2 (0.0%)	1
Yelp	1,987,693 (93.0%)	150,310 (7.0%)	2,138,003	21,357,271	1,238,535	1,088,225 (87.9%)	1,088,225

表2：各数据集的图和强连通分量统计信息。用户和场所分别是非空间节点和空间节点。用户/场所占节点的百分比。用户强连通分量是不包含空间信息的强连通分量，其占比以强连通分量为基准。不同的 R 树是指与其子节点相比具有不同空间可达性的 R 树。

2DReach 的理论最坏情况空间开销在实践中并未出现。

数据集	3DReach	3DReach-Rev	2DReach	2DReach-Comp	2DReach-Pointer
Foursquare	4.841	5.928	3.529	3.257	**3.241**
Gowalla	7.909	8.764	5.400	4.656	**4.639**
Weeplaces	1.225	1.745	0.803	0.681	**0.676**
Yelp	4.372	5.427	2.663	2.669	**2.661**

表 3：索引构建时间 [秒]：构建每种方法的强连通分量分解。此外，还构建了 3DReach 和 3DReach-Rev 的（反向）区间标记和三维 R 树、 2DReach 方法的 R 树和指针（以及位向量）。

数据集	3DReach	3DReach-Rev	2DReach	2DReach-Comp	2DReach-Pointer
Foursquare	35.1 (20.5 / 14.6)	78.5	49.3 (23.3 / 26.0)	40.3 (14.2 / 26.0)	**16.6** (14.2 / 2.4)
Gowalla	71.2（49.4 / 21.8）	164.1	73.0（47.9 / 25.0）	51.2 (26.1 / 25.0)	**26.7** (26.1 / 0.5)
Weeplaces	25.4 (17.6 / 7.8)	58.5	25.0 (17.1 / 7.9)	17.2 (9.3 / 7.9)	**9.5** (9.3 / 0.2)
Yelp	31.4 (2.7 / 28.7)	79.8*	39.0（21.9 / 17.1）	37.8（20.7 / 17.1）	**29.7** (20.7 / 8.9)

表4：索引大小 [MB。括号中的数值分别表示 3DReach 的（R树存储 / 标记存储）以及 2DReach 变体的（R树存储 / 指针存储）。* 大于 [2] 中报告的数值（44.8MB），可能是由于 Boost库版本的差异所致。

5.2.2 查询时间。 最后，我们分析不同方法的查询性能。这些实验的结果如图3所示。所有 Yelp 实验、Foursquare 顶点度和空间选择性比率实验，以及 Gowalla 空间选择性比率实验均采用对数刻度展示。其他所有实验均采用线性刻度，以适应查询时间上的差异。

首先，在除 Yelp 之外的几乎所有区域范围比取值下，两个 3DReach(-Rev) 均实现了最佳性能。这同样适用于 Weeplaces 实验和 Gowalla 顶点度实验。在这些情况下，2DReach 方法的性能仍然最高可提升 30%。造成这一差异的原因是，前者执行的是更简单的二维查询，而不是三维查询。

其他实验显示，3DReach(-Rev)呈现出更明显的指数级增长。在 Foursquare 顶点度实验和 Gowalla 空间选择性实验中，两种 3DReach 方法的表现均与 2DReach 不相上下。然而，在某些特定参数值下，它们的查询性能会慢几个数量级。相比之下，2DReach 缓解了这种指数级增长，无论参数值如何变化，都能保持相对稳定的查询性能。

在所有 Yelp 实验以及 Foursquare 选择性实验中，3DReach(-Rev)通常比 2DReach 的各个变体慢一个数量级。一个可能的原因是，无论图的结构如何，2DReach 查询都只需使用一个二维R树。2DReach 最显著的优势在于能够稳定并优化所有实验中的查询性能，避免查询性能大幅下降。

最后需要指出的是，2DReach 与 2DReach-Comp 的查询性能相近，尽管 2DReach-Pointer 始终比二者更慢。这一差异最高可达 30%，并且与其他 2DReach 变体不同，在高性能实验中，相较于 3DReach(-Rev)，2DReach-Pointer 失去了优势。然而，当 3DReach(-Rev) 的性能呈指数级下降时，2DReach-Pointer 的表现仍与 2DReach 和 2DReach-Comp 相当。显然，查询性能的损失是由于需要通过强连通分量节点和位向量来获取 R树指针。

6 结论

我们提出了 2DReach，它是最先进的 3DReach [2] 在地理社交可达性查询方面的一种更简单、更快速且更小巧的替代方法。通过消除区间标记，2DReach 以每个分量的二维R树取代 3DReach 的三维 R 树，从而简化索引构建和查询执行。压缩变体通过排除空间汇点，并在可达集合相同的分量之间共享 R 树，进一步减少存储需求。在四个真实世界数据集上的实验表明，2DReach 能够更快地构建索引；其中，2DReach-Pointer 在所有方法中索引大小最小，同时保持稳定的查询性能，在社交复杂度较高的数据集上，其速度比 3DReach 最多快一个数量级。

未来的工作可以探索 2DReach 在其他空间图领域的适用性，并研究持久化 R 树，以改进不同分量之间的 R 树共享。

制品可用性

源代码和实验脚本可在 https: // github.com/RickHeijden/rangereach 获取。

参考资料

[1] Rakesh Agrawal、Alexander Borgida 和 H. V. Jagadish。1989 年。在大型数据与知识库中高效管理传递关系。载于*1989 年 ACM SIGMOD* 数据管理国际会议论文集。ACM 出版社，美国俄勒冈州波特兰，253 262 10 1145 67544 66950 –。doi: 。/。

[2] Panagiotis Bouros、Theodoros Chondrogiannis 和 Daniel Kowalski。2025。。快速地理社交可达性查询。doi: 10 48786 2025 03 。/EDBT。。Justin J.

[3] Levandoski、Mohamed Sarwat、Ahmed Eldawy 和 Mohamed F. Mokbel。2012 年。LARS: 一种位置感知推荐系统。载于*2012 年 IEEE 第 28 届*国际数据工程会议（*ICDE '12*）论文集。IEEE 计算机学会，美国，450 461 10 1109 2012 54 –。doi: 。/ICDE。。

[4] Yong Liu、Wei Wei、Aixin Sun 和 Chunyan Miao。2014 年。利用地理邻域特征进行位置推荐。载于第 *23 届 ACM* 信息与知识管理国际会议论文集

管理（*CIKM '14*）。计算机协会，美国纽约州纽约，739 748 10 1145 2661829 2662002 –。doi：。/。Mohamed Sarwat、Justin J. Levandoski、Ahmed Eldawy 和 Mohamed F. Mokbel。2014 年。LARS*：一种高效且可扩展的位置感知推荐系统。*IEEE* 知识与数据工程汇刊 第 26 卷，第 6 期（2014 年 6 月），1384–1399。[5]

doi:10.1109/TKDE.2013.29[6] Yuhan Sun 和 Mohamed Sarwat。2016。GeoReach：一种利用空间范围谓词评估图可达性查询的高效方法。doi:10.48550/arXiv.1603.05355 arXiv:1603.05355 [cs]。