



计算机科学

COMPUTER SCIENCE

PIEnum:高效的概率图上路径枚举算法

谢文林, 杜明, 周军锋

引用本文

谢文林, 杜明, 周军锋. [PIEnum:高效的概率图上路径枚举算法](#)[J]. 计算机科学, 2025, 52(12): 115-124.

XIE Wenlin, DU Ming, ZHOU Junfeng. [PIEnum:Efficient Algorithm of Path Enumeration on Large Uncertain Graphs](#) [J]. Computer Science, 2025, 52(12): 115-124.

相似文章推荐 (请使用火狐或 IE 浏览器查看文章)

Similar articles recommended (Please use Firefox or IE to view the article)

[基于超图网络嵌入的蛋白质复合体识别算法](#)

Protein Complex Identification Algorithm Based on Hypergraph Network Embedding

计算机科学, 2025, 52(12): 102-114. <https://doi.org/10.11896/jsjcx.250900062>

[基于知识图谱嵌入的异构图欺诈用户检测](#)

Fraud User Detection Based on Heterogeneous Information Network with Knowledge Graph Eembedding

计算机科学, 2025, 52(11A): 250400085-7. <https://doi.org/10.11896/jsjcx.250400085>

[基于MOBSF_rule的安卓恶意软件检测方法](#)

MOBSF_rule Based Android Malware Detection Method

计算机科学, 2025, 52(11A): 250200120-11. <https://doi.org/10.11896/jsjcx.250200120>

[基于多层级图表征增强的加密应用流量识别方法](#)

Classification of Encrypted Application Traffic Enhanced by Multi-level GraphRepresentation

计算机科学, 2025, 52(11A): 241200126-7. <https://doi.org/10.11896/jsjcx.241200126>

[基于量子安全和脆弱水印的图像篡改检测与自恢复算法](#)

Image Tampering Detection and Self-recovery Algorithm Based on Quantum-secure and FragileWatermarking

计算机科学, 2025, 52(11A): 250900081-8. <https://doi.org/10.11896/jsjcx.250900081>

PIEnum: 高效的概率图上路径枚举算法

谢文林 杜明 周军锋

东华大学计算机科学与技术学院 上海 201620

(2222702@mail.dhu.edu.cn)

摘要 枚举概率图上两个顶点间的路径是分析两点间关系的基本手段。为了解决已有算法存在的剪枝不充分、冗余计算等问题,提出了一种基于剪枝和索引的算法——PIEnum,其任务是在概率图 G 上枚举所有从起点 s 到终点 t ,长度不超过 k 且路径上所有边的概率累积值不低于 γ 的简单路径,其中 k 和 γ 分别为给定的路径长度约束值和概率阈值。对于一个查询,PIEnum首先剔除无效顶点以缩减路径枚举的搜索空间,然后构建一个轻量级的在线索引来避免路径枚举过程中重复的剪枝判断,最后在路径枚举的过程中将无效的搜索分支剪枝。为了进一步提升算法在稠密图上的查询效率,基于Join模式实现了PIEnum⁺。在10个真实数据集上检验了该算法的性能,实验结果表明,PIEnum整体性能比已有算法提升了10倍以上。

关键词: 图; 概率图; 路径枚举; 剪枝; 索引

中图分类号 TP301

PIEnum: Efficient Algorithm of Path Enumeration on Large Uncertain Graphs

XIE Wenlin, DU Ming and ZHOU Junfeng

School of Computer Science and Technology, Donghua University, Shanghai 201620, China

Abstract A basic approach of investigating the relationship between two vertices on the uncertain graph is to enumerate all paths between them. To solve the problem of inadequate pruning and redundant computation in the state-of-the-art algorithms, this paper proposes an pruning and index based algorithm, PIEnum, whose target is to enumerate simple paths from a source vertex s to a target vertex t where the length of each path is no more than a given hop constraint k and the probability of each path is no less than a given probability threshold γ . For an input query, it firstly excludes the unpromising vertices to reduce the search space. Then it builds an online light-weight index to avoid repeated pruning examinations during the enumeration. Finally, it develops an efficient approach to prune invalid search branches during the enumeration. To further improve the performance on dense graphs, it implements PIEnum⁺ based on the Join paradigm. The comprehensive experimental results on 10 real-life graphs show that PIEnum improves the overall performance by at least 10 times compared to the state-of-the-art algorithms.

Keywords Graph, Uncertain graph, Path enumeration, Pruning, Index

1 引言

图(Graph)是一种表示实体及其关系的基础模型,它广泛应用于许多领域^[1-5]。概率图(Uncertain Graph)是一种特殊的图结构,它在普通图的每条边上赋予了一个表示该条边存在的概率值。概率图能够很好地表达实体间的不确定性,在许多实际应用,如社交网络、生物网络、道路网络中发挥着重要的作用^[6-15]。图分析的基本任务之一是研究图上两个顶点之间的关系,而枚举两点之间的路径是研究这两个点之间关系的重要手段。近些年,枚举两点间路径的相关工作得到广泛研究^[16-27],主要分为常规图和概率图上的路径枚举问题两类。对于常规图上的路径枚举问题,其任务是枚举所有从

源点 s 到终点 t 的长度不超过一个给定跳数约束值的简单路径(Hop-constrained $s-t$ Simple Path Enumeration, HcPE),其中跳跃约束值即为路径长度约束值,简单路径指不经过重复顶点的路径。概率图上路径枚举问题(Hop-constrained $s-t$ Simple Path Enumeration in Uncertain Graphs, UHcPE)在HcPE问题的基础上增加了概率约束阈值的限制,即任意一条路径上所有边的概率值的乘积必须不低于给定的概率约束阈值。目前,针对该问题的研究仍处于初步阶段,存在较大的探索空间。和HcPE相比,UHcPE利用概率阈值过滤低概率的路径,更适合分析具有不确定性的实体间的关系。因此,UHcPE在现实世界中有着广泛的应用场景^[6-10],以下是几个具有代表性的UHcPE问题应用实例。

到稿日期:2024-11-24 返修日期:2025-02-15

基金项目:国家自然科学基金(62372101,61873337,62272097)

This work was supported by the National Natural Science Foundation of China(62372101,61873337,62272097).

通信作者:杜明(duming@dhu.edu.cn)

1) 社交网络。分析两个用户之间的亲密度是社交网络的一个基本需求^[7-8]。用概率图对社交网络进行建模,用顶点及边表示用户及用户间关系,而用户间的亲密度可以用概率图中边的概率值表示。通过枚举两个用户间满足给定的路径长度约束值和概率阈值的简单路径,可以有效地研究用户间的亲密度,结果路径越多,他们之间的亲密度越高。此外,可以灵活地调节路径长度约束值和概率阈值,以满足实际需求。具体来说,如果需要得到可信度更高的路径,可以增大概率阈值,反之亦然。如果需要扩大搜索范围,可以增大路径长度约束值,反之亦然。

2) 蛋白质网络。蛋白质分子之间存在相互作用关系,而这种关系一般具有不确定性^[9]。用概率图中的顶点和边分别表示蛋白质分子及分子间的作用关系,边上的概率值表示两个蛋白质分子之间存在作用关系的可能性。通过枚举蛋白质分子之间满足给定的路径长度约束值和概率阈值的简单路径,生物学家可以有效地分析蛋白质分子间作用关系的传导链路。

文献[27]是首次研究 UHcPE 问题的,其提出的 UnEnum 算法是目前解决 UHcPE 问题的最优算法。UnEnum 提出了概率约束距离的概念,并利用概率约束距离剔除了部分的无效顶点,以缩减路径枚举的搜索空间。此外,UnEnum 设计了具有一定剪枝能力的路径枚举算法。然而,UnEnum 存在以下几点缺陷:

1) 只能剔除部分无效顶点,导致在路径枚举阶段仍然需要遍历大量无效的顶点和边;

2) 在路径枚举阶段需要重复判断剪枝条件是否成立,导致时间开销过大;

3) 在路径枚举阶段只能剪枝部分无效路径,仍然存在大量的无效路径无法被剪枝。

本文提出了高效的算法来解决 UHcPE 问题,其主要思想是设计高效的剪枝技术剔除无效顶点和无效搜索路径,同时设计高效的索引避免重复的剪枝条件判断。本文的贡献总结如下:

1) 结合路径长度约束值和概率阈值设计了高效的剪枝策略,以剔除无效顶点和无效搜索路径。

2) 设计了一种高效的在线索引 UI(Index on Uncertain Graphs)避免了路径枚举过程中重复的剪枝判断。UI 提供了一种在常数时间复杂度下给出当前点所有有效的邻居顶点的查询方法。

3) 在 10 个真实数据集上进行了全面的实验,结果表明,本文算法综合性能相比 UnEnum 提升了 10 倍以上。

本文第 2 章介绍与研究内容相关的基础知识;第 3 章介绍 HcPE 和 UHcPE 问题的相关研究;第 4 章介绍本文算法 PEnum 的整体思路及其实现过程;第 5 章展示实验的详细数据;最后总结全文并展望未来。

2 预备知识

本章首先介绍与研究内容相关的预备知识,然后介绍本文的问题定义。表 1 总结了本文所用到的数学符号及其表示意义。

表 1 符号及其含义

Table 1 Symbols and their meanings

符号	含义
G, G_r	有向概率图及其反向图
$V, V $	概率图中顶点集合及其包含顶点的数量
$E, E $	概率图中边集合及其包含边的数量
$\delta(e)$	边的概率值
$nbr^+(v), nbr^-(v)$	v 的出邻居和入邻居的集合
$p, p , p[i]$	路径, 路径的长度, 路径中下标为 i 的顶点
$\theta(p)$	路径上所有概率的累积值
s, t	起点和终点
k, γ	路径长度约束值和概率阈值
$UDist(u, v, \gamma)$	u 到 v 的概率约束距离
$MPP(u, v)$	u 到 v 的最大概率累积值

用 $G=(V, E, \delta)$ 表示一个概率图,其中 V 和 E 分别表示所有顶点和边组成的集合, δ 表示为每条边 $e \in E$ 赋予一个概率值 $\delta(e)$ 的函数, $0 < \delta(e) \leq 1$ 。使用 $|V|$ 和 $|E|$ 分别表示 G 中的顶点数和边数, G_r 表示 G 的反向图, G_r 在 G 的基础上将所有的边的方向逆置。对于图 G 上的一个顶点 v ,用 $nbr^+(v)$ 和 $nbr^-(v)$ 分别表示顶点 v 的出邻居和入邻居的集合。图上的一条从点 u 到点 v 的路径 p 由一组顶点序列组成,这组顶点序列上所有相邻顶点之间有边相连。用 $|p|$ 表示路径 p 的长度,即从起点到终点的跳数。 $p[i]$ 表示 p 中第 i 个位置的顶点, $0 \leq i \leq |p|$ 。如果一条路径 p 不存在重复的顶点,则将这样的路径称为简单路径(Simple Path)。给定一个简单路径 p ,概率累积值 $\theta(p)$ 表示这条路径上所有边的概率值的乘积,即 $\theta(p) = \prod_{i=1}^{|p|} \delta(e_i)$,其中 $p = \{e_i | i \in (1, |p|)\}$ 。用 $Dist_G(u, v)$ 表示 G 中顶点 u 到顶点 v 的距离,即 u 到 v 的最短路径的长度。进一步,用 $UDist_G(u, v, \gamma)$ 表示概率约束阈值为 γ 时, u 到 v 的概率约束距离,即 u 到 v 所有路径中满足 $\theta(p) \geq \gamma$ 且长度最短的路径长度。当上下文清晰时,省略下标 G 。下面给出本文的问题定义。

问题定义:给定一个概率图 G ,起点 s 和终点 t ,路径长度约束值 k 和一个概率约束阈值 γ ,枚举出所有从 s 到 t 的满足 $|p| \leq k$ 并且 $\theta(p) \geq \gamma$ 的简单路径。用 $q(s, t, k, \gamma)$ 表示上述查询并用 $P(q)$ 表示 q 输出路径的集合。下面为示例。

例 1 给定图 1 中的概率图 G_0 ,起点为 s ,终点为 t ,若 $k=5, \gamma=0.5$,只有一条路径 $p = \{s, v_3, v_2, v_5, t\}$ (图中红箭头所示)满足查询 $q(s, t, 5, 0.5)$ 。

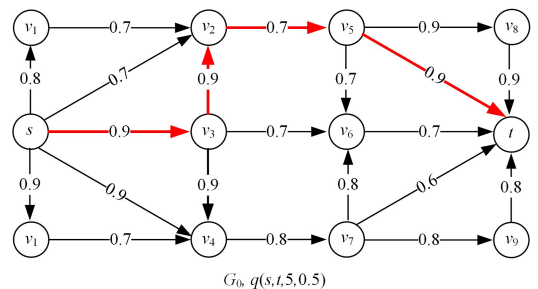


图 1 问题定义举例(电子版为彩图)

Fig. 1 Example of the problem definition

3 相关工作

解决常规图和概率图上的路径枚举(HcPE 和 UHcPE)

问题的算法大致分为两个阶段。

- 1) 路径枚举前。该阶段设计的剪枝策略将不可能出现在路径结果集中的顶点提前剔除,以缩减路径枚举的搜索空间。
- 2) 路径枚举。该阶段执行 DFS 枚举符合条件的路径。在搜索过程中,使用剪枝策略避免无效的搜索。

3.1 HcPE 问题

HcPE 一直是备受关注的问题,近几年,不同应用场景的 HcPE 算法被相继提出。文献[19]和文献[20]提出的 T-DFS (DFS-like Traversal)和 T-DFS2 算法通过保证每条搜索分支都至少有一条结果来降低枚举过程中的搜索开销。文献[21]提出的 Bc-DFS(Barrier-constrained DFS)算法设计了一种基于障碍值的剪枝策略,通过动态地维护障碍值来避免无效路径的搜索。同时,文献[21]还提出了一种基于 Join 模式的算法 Bc-Join,它将搜索过程一分为二,对于每个部分,应用 Bc-DFS 算法求得中间路径,最后将路径进行拼接得到最终路径。Bc-Join 能有效降低稠密图路径枚举的时间开销。文献[22]提出的 HP-Index(Hot Point Index)算法通过存储图中的稠密点之间的路径,以加速稠密点之间的路径枚举。文献[23]提出的 PEPF(Path Enumeration on FPGA)算法以 Bc-DFS 为基础,利用 FPGA(Field Programmable Gate Array)技术对简单路径枚举进行了加速。文献[24]提出的 HybridEnum 设计了一种在分布式系统上进行简单路径枚举的高效算法,利用分布式系统的算力加速了查询。文献[25]提出了一种基于在线索引的算法 PathEnum,它设计了一种高效的在线索引,缩小了路径枚举的搜索空间并提供了一种在枚举阶段快速查询有效顶点的方法,加速了枚举过程。PathEnum 是当前解决 HcPE 问题的最优算法。文献[26]首次研究了动态图上的 HcPE 问题,提出了一种基于部分路径的索引和高效的路径枚举算法。为了应对动态图的变化,文献[26]还设计了一种高效的索引维护算法。

3.2 UHcPE 问题

概率图上的简单路径枚举算法目前还未得到广泛研究。文献[27]提出的 UnEnum 算法是目前解决 UHcPE 问题的

最优算法。在路径枚举前,UnEnum 使用 BFS 求得图上任意一点到源点 s 和终点 t 的概率约束距离,并根据概率约束距离将无效顶点及其对应的边提前剔除,缩小了搜索空间。具体地,对于图上的任意一点 v ,如果 $UDist(s, v, \gamma) + UDist(t, v, \gamma) > k$,那么 v 属于无效顶点,于是将 v 及其对应的边剔除,从而缩小路径枚举过程的搜索空间。在路径枚举阶段,UnEnum 根据当前路径的长度和当前点的邻居节点到终点 t 的概率约束距离将无效的搜索分支剪枝,加速了枚举过程。具体地,如果当前搜索分支 p' 满足 $|p'| + UDist(t, p[|p'|], \gamma) > k$,那么 p 不可能到达终点 t ,DFS 将结束当前分支的搜索。

4 本文算法

本章首先分析已有算法 UnEnum 存在的问题并给出本文算法的主体思路,然后详细介绍本文算法的具体实现。

4.1 总体思路

UnEnum 存在以下 3 个缺点。

1) UnEnum 在路径枚举前剔除部分无效顶点后仍然可以进一步剔除无效顶点。以图 2 说明这一问题,其中 $k=5, \gamma=0.5$,虚线部分表示被剔除的顶点和边。UnEnum 在路径枚举前剔除了无效顶点 v_0 和 v_1 以及对应的边(G_1 中虚线部分)。然而,点 v_4, v_6, v_7, v_8, v_9 也属于无效顶点,因为结果集中不存在经过这些点的路径,因此,在路径枚举前应该剔除这些点及其对应的边(G_2 中虚线部分)。对比 G_1 和 G_2, G_1 中有 8 个顶点($v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9$)在路径枚举过程中被访问,而 G_2 中仅有 3 个顶点(v_2, v_3, v_5)被访问。下面分析造成上述问题的原因。以图中的顶点 v_4 为例,对于任意一条从 s 到 t 经过 v_4 的路径 $p, \theta(p) < \gamma$ 始终成立,因此 v_4 属于无效顶点,应该被剔除。然而, $UDist(s, v_4, \gamma) + UDist(t, v_4, \gamma) < 5$ 不满足 UnEnum 的剪枝条件,因此 v_4 无法被 UnEnum 剔除。由此可见,若一个顶点 v 满足 $UDist(s, v, \gamma) + UDist(t, v, \gamma) \leq k$,不能保证至少存在一条经过 v 的有效路径,因此 v 可能是无效顶点。

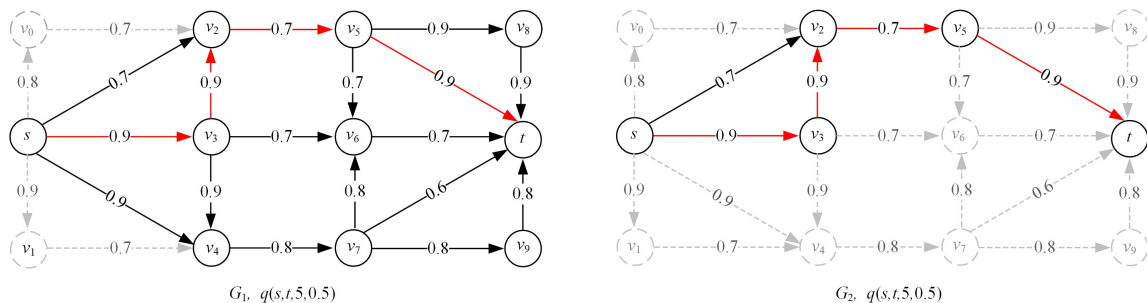


图 2 UnEnum 存在问题的示例

Fig. 2 Observation of the problem of UnEnum

2) UnEnum 在路径枚举过程中无法避免重复的剪枝判断。当 DFS 遍历任意顶点 v 时,需要根据剪枝条件判断其是否能被剪枝,当再次遍历到 v 时,仍然需要重复判断剪枝条件,加剧了路径枚举的时间开销。

3) UnEnum 在路径枚举阶段只能部分剪枝无效搜索分支。以图 2 中的 G_2 来说明这个问题。 G_2 是剔除无效顶点后

得到的子图。在 DFS 遍历过程中,当搜索分支为 $p = \{s, v_2\}$ 时, $|p| + UDist(t, v_2, \gamma) = 1 + 2 < k$,不满足剪枝条件,因此 DFS 将继续向下搜索。然而,这条搜索分支向下搜索并无结果。

通过以上分析,本文的目标是通过进一步缩小小路径枚举的搜索空间和避免冗余的计算来提高路径枚举的整体性能。

为了达到这一目标,首先设计一种通过充分利用路径长度约束值和概率阈值来剔除无效顶点的剪枝算法,然后构建一个轻量级的在线索引来避免重复的剪枝判断,最后设计高效的路径枚举算法进一步对无效搜索分支进行剪枝。

首先给出本文算法 PIEnum (Pruning and Index based Enumeration) 的总体概览。如算法 1 所示,对于一个查询 $q(s, t, k, \gamma)$, PIEnum 首先通过调用算法 ExcludeVertices 来剔除无效顶点,同时将余下的顶点存储在集合 S 中(第 1 行)。然后, PIEnum 利用 S 中顶点构建索引 UI(第 2 行)。最后, PIEnum 调用路径枚举算法 PathEnum 输出所有符合条件的路径(第 3 行),在路径枚举过程中,无效搜索分支会被剪枝。

算法 1 PIEnum

输入: $G, G_r, q(s, t, k, \gamma)$

输出: $P(q)$

1. $S \leftarrow \text{ExcludeVertices}(G, G_r, q)$;
2. $UI \leftarrow \text{BuildIndex}(G, G_r, q, S)$;
3. $P(q) \leftarrow \text{PathEnum}(G, q, UI)$;

4.2 无效顶点的剔除

最大概率累积值的定义如下。

定义 1 (最大概率累积值, Maximal Product of Probabilities, MPP) 给定概率图 G 、顶点 u 和 v , 对于从 u 到 v 的所有路径, $MPP_G(u, v)$ 表示这些路径的概率累积值的最大值。

当上下文清晰时,省去下标 G 。以 G_2 中顶点 s 和 v_2 为例, s 到 v_2 有两条路径 $p_1 = \{s, v_2\}$ 和 $p_2 = \{s, v_3, v_2\}$, 由于 $\theta(p_1) = 0.7 < \theta(p_2) = 0.81$, 因此 $MPP(s, v_2) = 0.81$ 。

基于 MPP 的定义, 有如下定理。

定理 1 给定概率图 G 、查询 $q(s, t, k, \gamma)$ 和一个顶点 v , 如果 $MPP(s, v) * MPP(t, v) < \gamma$, 那么不存在一条路径 p 同时满足 $p \in P(q)$ 并且 $v \in p$ 。

证明: 假设存在一条路径 p 同时满足 $p \in P(q)$ 并且 $v \in p$, 将 p 从 v 分为左右两部分, 得到 p_1 和 p_2 , 则有 $\theta(p) = \theta(p_1) \times \theta(p_2) \geq \gamma$ 。根据 MPP 的定义, 有 $\theta(p_1) \leq MPP(s, v)$, $\theta(p_2) \leq MPP(t, v)$ 。又已知 $MPP(s, v) \times MPP(t, v) < \gamma$, 因此 $\theta(p_1) \times \theta(p_2) < \gamma$, 即 $\theta(p) < \gamma$, 显然与 $\theta(p) \geq \gamma$ 矛盾。证毕。

根据定理 1, 提出第 1 条剪枝规则, 其作用是在路径枚举之前剔除无效顶点。

剪枝规则 1 给定概率图 G 和查询 $q(s, t, k, \gamma)$, 对于任意的顶点 $v \in V$, 如果 $UDist(s, v, \gamma) + UDist(t, v, \gamma) > k$ 或者 $MPP(s, v) \times MPP(t, v) < \gamma$ 成立, 则 v 是一个无效顶点。

基于剪枝规则 1, 提出剔除无效顶点的算法 ExcludeVertices。如算法 2 所示, 对于一个查询, 首先调用算法 GetUDistMPP 分别计算正向图和反向图上从起点和终点到图上任意点的概率约束距离 $UDist$ 和最大概率累积值 MPP (第 1 行)。然后遍历所有的 $v \in V$, 检查 v 是否满足剪枝规则 1。如果满足, 将 v 剔除, 否则将 v 加入集合 S (第 2—5 行)。最后得到 $UDist$ 、 MPP 和有效顶点的集合 S 。

算法 2 ExcludeVertices

输入: $G, G_r, q(s, t, k, \gamma)$

输出: $UDist, MPP, S$

1. $UDist, MPP \leftarrow \text{GetUDistMPP}(G, s, k, \gamma)$;

2. $S \leftarrow \emptyset$;

3. for $v \in V$ do

4. if v cannot meet Pruning rule 1 then

5. $S = S \cup \{v\}$;

算法 3 详细描述了计算起点 s 到 G 中任意一点的 $UDist$ 和 MPP 的过程。第 1—2 行为图中的每一个顶点初始化 $UDist$ 和 MPP 。第 3 行初始化 s 的 $UDist$ 和 MPP , 并将 s 添加到队列 Que 中。第 4—14 行执行 BFS 来计算 s 到所有顶点的 $UDist$ 和 MPP 。对于当前从 Que 出队的顶点 v , 如果 $UDist(s, v, \gamma) < k$, 则继续访问满足 $MPP(s, v) \times \delta(v, v') \geq \gamma$ 的邻居 v' (第 5—7 行)。如果 v' 没有被访问过, 则设置 $UDist(s, v, \gamma)$ 和 $MPP(s, v)$ 的值, 并将 v' 入队 (第 8—11 行)。如果 v' 满足 $MPP(s, v) \times \delta(v, v') > MPP(s, v')$, 则需要更新 $MPP(s, v')$ 以确保其为最大概率累积值, 并将 v' 重新入队 (第 12—14 行)。最终得到 s 到图上所有点的 $UDist$ 和 MPP 。在反向图 G_r 上求终点 t 到图上所有点的 $UDist$ 和 MPP 的过程和正向图一致, 因此省略。

算法 3 GetUDistMPP

输入: G, s, t, k, γ

输出: $UDist, MPP$

1. for $v \in V$ do

2. $UDist(s, v, \gamma) = k, MPP(s, v) = 0$;

3. $UDist(s, s, \gamma) = 0, MPP(s, s) = 1, Que = \{s\}$;

4. while Que is not empty do

5. dequeue v from Que ;

6. if $UDist(s, v, \gamma) < k$ then

7. for $v' \in nbr^+(v)$ s. t. $MPP(s, v) \times \delta(v, v') \geq \gamma$ do

8. if $UDist(s, v', \gamma) = k$ then

9. $UDist(s, v', \gamma) = UDist(s, v, \gamma) + 1$;

10. $MPP(s, v') = MPP(s, v) \times \delta(v, v')$;

11. $Que = Que \cup \{v'\}$;

12. else if $MPP(s, v) \times \delta(v, v') > MPP(s, v')$ then

13. $MPP(s, v') = MPP(s, v) \times \delta(v, v')$;

14. $Que = Que \cup \{v'\}$;

例 2 以图 1 中的 G_0 为例来描述算法 2 剔除无效顶点的过程。算法 2 首先调用算法 3 计算 $UDist$ 和 MPP , 其结果如表 2 所列, 其中 $k = 5, \gamma = 0.5$ 。然后, 算法 2 通过检查 G_0 中的顶点是否满足剪枝规则 1 来剔除无效顶点, 最终得到图 2 中的 G_2 。以 v_8 为例来说明它是如何被剔除的。由表 2 可得, $MPP(s, v_8) \times MPP(t, v_8) = 0.5 \times 0.9 = 0.45 < \gamma$, 满足剪枝规则 1, 因此 v_8 是无效顶点, 可以被剔除。

表 2 G_0 上各个顶点的 $UDist$ 和 MPP

Table 2 $UDist$ and MPP on G_0

顶点 v	$UDist(s, v, \gamma)$	$UDist(t, v, \gamma)$	$MPP(s, v)$	$MPP(t, v)$
v_0	1	5	0.80	0
v_1	1	5	0.90	0
v_2	1	2	0.81	0.63
v_3	1	3	0.90	0.56
v_4	1	3	0.90	0.51
v_5	3	1	0.56	0.90
v_6	2	1	0.63	0.70
v_7	2	1	0.72	0.64
v_8	4	1	0.50	0.90
v_9	3	1	0.58	0.80

正确性:算法2不会将有效顶点错误地剔除。假设存在一个有效顶点 v 满足剪枝规则1,如果 $UDist(s,v,\gamma) + UDist(t,v,\gamma) > k$ 成立,对于任意一条从 s 到 t 并经过 v 的路径 $p, |p| > k$ 恒成立,因此所有经过 v 的路径都是无效路径,这和 v 是一个有效顶点矛盾,如果 $MPP(s,v) \times MPP(t,v) < \gamma$ 成立,对于任意一条从 s 到 t 并经过 v 的路径 $p, \theta(p) < \gamma$ 恒成立,因此所有经过 v 的路径都是无效路径,这和 v 是一个有效顶点矛盾。综上, v 是一个无效顶点,可以被剔除。

时间复杂度:算法2首先执行两次BFS来计算 $UDist$ 和 MPP (第1行),其时间复杂度为 $O(|V| + |E|)$ 。第3行遍历图上每个顶点,其时间复杂度为 $O(|V|)$ 。因此,算法2的总体时间复杂度为 $O(|V| + |E|)$ 。

空间复杂度:算法2维护了 G 中所有顶点的 $UDist$ 和 MPP ,因此其空间复杂度都为 $O(|V|)$ 。集合 S 的空间复杂度为 $O(|V|)$,因为 S 中最多存在 $|V|$ 个顶点。因此,算法2的总体空间复杂度为 $O(|V|)$ 。

4.3 索引构建

在路径枚举阶段,对于任意搜索分支 p' ,UnEnum需要检查 $|p'| + UDist(t, p'[|p'|], \gamma) > k$ 是否成立来判断 p' 是否能被剪枝,导致时间开销过大。本文构建一个轻量级的在线索引UI来避免上述情况发生。

图3以 G_2 为例展示了UI的结构。UI由3部分组成,一个hash表 H 、一个二维数组 O 和一个二维数组 P 。 H 维护了集合 S (算法2求得的有效顶点的集合)中顶点的序号,其键和值分别为顶点及其在 S 中的序号。 P 中的每一行存储了 H 中对应顶点的邻居以及对应的概率值,它们按照到终点 t 的概率约束距离升序排列。以 P 中的第1行为例,顶点 v_2 和 v_3 是顶点 s 的两个邻居,数值0.7和0.9分别表示边 (s, v_2) 和 (s, v_3) 的概率值。为了查询 H 中某一点 v 在 P 中对应的任意邻居的位置,用二维数组 O 记录这些邻居的偏移量。 O 的行数和 H 的行数相同,每一行有 k 个槽位(因为 P 中存储的邻居到终点 t 的概率约束距离最多有 k 个不同的值,图中 $k=5$,因此 O 有5列),第 $i(0 \leq i < k)$ 个槽位中的数字在数值上等于到终点 t 的概率约束距离 $\leq i$ 的邻居的数量。以第1行为例, $O[i][j]$ 表示 O 中第 $i+1$ 行第 $j+1$ 列的数值, $O[0][3]=2$ 表示顶点 s 的邻居中到终点 t 的概率约束距离 ≤ 3 的有2个。由于 P 中每一行的邻居按照到 t 的概率约束距离升序排列,因此上述2个邻居分别是 $P[0][0]$ (含义同上)和 $P[0][1]$,即 v_2 和 v_3 。

	0	1	2	3	4		
s	0	0	0	1	2	$v_2, 0.7$	$v_3, 0.9$
v_2	1	0	1	1	1	$v_5, 0.7$	
v_3	2	0	0	1	1	$v_2, 0.9$	
v_5	3	1	1	1	1	$t, 0.9$	

hash table H offset O neighbor pairs P 图3 UI在 G_2 中的结构Fig. 3 Structure of UI on G_2

利用UI,在路径枚举阶段,不需要检查剪枝规则就能得到当前搜索分支有效的邻居顶点。下面给出一个示例。

例3 以 G_2 为例,如果当前搜索分支为 $p' = \{s, v_3\}$,还

能继续搜索的最大步数 $b = k - |p'| = 5 - 1 = 4$ 。假设 v 是 v_3 的一个邻居,如果 v 是一个有效点,那么 $UDist(t, v) \leq b - 1 = 3$ 一定成立,其中1表示 v_3 到 v 的路径长度。为了得到 v_3 所有的有效邻居,首先得到 v_3 在 H 中的序号 i ,即 $i = H[v_3] = 2$ (见图3)。然后根据 O 得到 v_3 的有效邻居的数量,即 $O[i][b-1] = O[2][3] = 1$ 。最后得到 v_3 的有效邻居的集合为 $\{P[i][j] | i=2, j \in [0, 1)\}$,即 $\{(v_2, 0.9)\}$,其中0.9表示边 (v_3, v_2) 的概率值。

用函数 $UI(p')$ 表示上述查询过程,如算法4所示,其输入为当前搜索分支 p' ,输出为 p' 对应的有效邻居的集合。具体来说,第1行得到当前搜索分支最后一个顶点 v ,第2行计算 v 的剩余步数 b ,第3行得到 v 的序号 i ,第4行计算 v 的有效邻居的数量 $num = O[i][b-1]$,第5行得到有效邻居的集合 $UI(p') = \{P[i][j] | j \in [0, num)\}$ 。 $UI(p')$ 的查询时间复杂度为 $O(1)$,因此其能显著提升路径枚举的效率。

算法4 $UI(p')$

输入: $H, O, P, q(s, t, k, \gamma), p'$

输出: $UI(p')$

1. $v = p'[|p'|]$;
2. $b = k - |p'|$;
3. $i = H[v]$;
4. $num = O[i][b-1]$;
5. $UI(p') = \{P[i][j] | j \in [0, num)\}$;

下面介绍如何构建UI,其过程如算法5所示。具体来说,第1-4行初始化 H, O, P 和一个辅助变量 i ,第5-13行遍历 S 中的所有顶点来构建UI。对于当前顶点 v ,首先,设置其在 H 中的序号(第6行)。然后,第7-11行将 v 的邻居根据到 t 的概率约束距离分为 k 组,在这个过程中不断维护 O (第11行)。最后,第12行将排序好的邻居顶点及其概率的集合存入 P ,第13行更新 i 。遍历 S 中所有的顶点后即完成UI的构建。

算法5 BuildIndex

输入: $G, G_r, P, q(s, t, k, \gamma), UDist, S$

输出:UI

1. $H \leftarrow$ a hash table;
2. $O \leftarrow$ a $|S| \times k$ two-dimensional array;
3. $P \leftarrow$ a two-dimensional array with $|S|$ rows;
4. $i = 0$;
5. for $v \in S$ do
6. $H[v] = i$;
7. $row = \emptyset$;
8. for $j \in [0, k)$ do
9. $pairs = \{(v', \delta(v, v')) | v' \in nbr^+(v) \wedge UDist(t, v', \gamma) = j\}$;
10. $row \leftarrow row \cup pairs$;
11. $O[i][j] = |row|$;
12. $P[i] = row$;
13. $i = i + 1$;

正确性:对于任意搜索分支 p' , $UI(p')$ 能够正确返回所有有效邻居。一方面,UI的构建过程保证了所有邻居按照到 t 的概率约束距离升序排列。另一方面,UI的查询过程是正确的。算法4的第2行保证 $b > 0$ 。如果 $b = 0$,则意味着当前

搜索分支 p' 的长度已经达到路径长度约束值, 这时已经没必要查询 $UI(p')$, 因此第 4 行能得到正确的结果。综上所述, $UI(p')$ 能够正确返回所有有效邻居。

下面分析构建 UI 的时间复杂度和空间复杂度。

时间复杂度: 算法 5 第 5—15 行遍历 S 中的所有顶点的邻居, 其时间复杂度为 $O(|E|)$; 第 8—12 行使用计数排序对每个顶点的邻居排序, 其时间复杂度为 $O(|E|)$ 。因此, 算法 5 的时间复杂度为 $O(|E|)$ 。

空间复杂度: H 的空间复杂度为 $O(|V|)$, 因为 H 最多存储所有的顶点。 O 是一个 $|S| \times k$ 的二维数组, 其空间复杂度为 $O(k \times |V|)$, 因为其最多有 $|V|$ 行。 P 的空间复杂度为 $O(|E|)$, 因为 P 最多存储所有的边。 综上, 算法 5 的空间复杂度为 $O(k \times |V| + |E|)$ 。

4.4 路径枚举

在路径枚举阶段, 根据如下定理对无效的搜索分支进行剪枝。

定理 2 给定一个概率图 G 、一个查询 $q(s, t, k, \gamma)$ 和当前搜索分支 p' , 如果 $\theta(p') \times MPP(t, p'[\lfloor |p'| \rfloor]) < \gamma$ 成立, 则不存在一条路径 p 同时满足 $p \in P(q)$ 和 $p' \subseteq p$ 。

证明: 假设存在一条路径 p 同时满足 $p \in P(q)$ 和 $p' \subseteq p$ 。将 p 分为 p' 和 p'' , 即 $p = p' \cup p''$, 那么 $\theta(p') \times \theta(p'') = \theta(p) \geq \gamma$ 。根据 MPP 的定义, 有 $\theta(p'') \leq MPP(t, p'[\lfloor |p'| \rfloor])$, 又已知 $\theta(p') \times MPP(t, p'[\lfloor |p'| \rfloor]) < \gamma$, 于是得到 $\theta(p') \times \theta(p'') < \gamma$, 这与 $\theta(p') \times \theta(p'') \geq \gamma$ 矛盾。证毕。

根据定理 2 介绍剪枝规则 2 其目标是在路径枚举阶段对无效搜索分支进行剪枝。

剪枝规则 2 给定一个概率图 G 、一个查询 $q(s, t, k, \gamma)$ 和当前搜索分支 p' , 如果 $\theta(p') \times MPP(t, p'[\lfloor |p'| \rfloor]) < \gamma$ 成立, 则 p' 为无效搜索分支。

基于 UI 和剪枝规则 2, 提出路径枚举算法, 其描述如算法 6 所示。给定查询 $q(s, t, k, \gamma)$, 算法 6 以递归的形式执行, 用 SEARCH 表示递归函数(第 1—8 行)。具体来说, 第 2 行取当前搜索分支最后一个顶点 v , 第 3 行根据 $UI(p')$ 得到 v 的有效邻居的集合, 并遍历每一个有效邻居 v' 。第 4 行根据剪枝规则 2 判断 v' 是否能被剪枝。对于不能被剪枝的邻居, 如果 v' 等于终点 t , 说明找到了一条有效路径, 将其存入 $P(q)$ (第 5—6 行), 否则进行下一次递归, 直到当前路径长度超过 k (第 7—8 行)。第 9 行调用 SEARCH, 最终输出所有有效路径的集合 $P(q)$ 。

算法 6 PathEnum

输入: $G, G_r, q(s, t, k, \gamma), UI, UDist, MPP$

输出: $P(q)$

```

1. function SEARCH( $p'$ )
2.    $v = p'[\lfloor |p'| \rfloor]$ ;
3.   for ( $v', pb$ )  $\in UI(p')$  do
4.     if  $\theta(p') \times pb \times MPP(t, v') \geq \gamma \wedge v' \notin p'$  then
5.       if  $v' = t$  then
6.         add  $p' \cup \{v'\}$  to  $P(q)$ ;

```

```

7.   else if  $|p'| + 1 < k$  then
8.     SEARCH( $p' \cup \{v'\}$ );
9.    $P(q) \leftarrow SEARCH(\{s\})$ ;

```

例 4 以图 2 为例来说明算法 6 枚举路径的过程。算法 6 从 s 开始枚举路径, 通过查询 UI 得到两个有效邻居 v_2 和 v_3 。对于搜索分支 $p' = \{s, v_2\}$, $\theta(p') = 0.7$, $MPP(t, v_2) = 0.63$ (见表 2), $\theta(p') \times MPP(t, v_2) = 0.7 \times 0.63 < 0.5$, 满足剪枝规则 2, 因此 $p' = \{s, v_2\}$ 被剪枝。然后, 算法依次遍历 v_3, v_2, v_5, t , 完成所有遍历, 最终得到有效路径 $p = \{s, v_3, v_2, v_5, t\}$ 。

正确性: 算法 6 能够输出所有满足查询 $q(s, t, k, \gamma)$ 的路径。鉴于算法 2 和 UI 的正确性, 算法 6 的正确性能够得到保证。需要说明的是, 算法 6 本身能够保证枚举的路径没有重复并且都是简单路径。

时间复杂度: 首先考虑输出一条有效路径的时间复杂度。由于搜索的深度被 k 限制, 并且任意搜索分支获取有效邻居的时间开销都为 $O(1)$ (算法 6 第 3 行), 因此输出一条有效路径的时间复杂度为 $O(k)$ 。用 τ 表示有效路径的数量, 则算法 6 的时间复杂度为 $O(k \times \tau)$ 。

空间复杂度: 算法 6 的空间复杂度为 $O(k \times \tau)$, 其中 τ 表示有效路径的数量, 因为每条路径的最大长度为 k 。

4.5 小结

本章介绍了本文算法的总体思路和具体实现细节, 下面阐述 6 个算法之间的联系。算法 1 描述了本文算法的总体思路, 其由算法 2、算法 5、算法 6 组成, 其中算法 2 描述了路径枚举前剔除无效顶点的过程, 算法 5 描述了构建索引的过程, 算法 6 描述了最终枚举路径的过程。算法 3 是算法 2 的一个环节, 其描述了如何求解概率约束距离 UDist 和最大概率累积值 MPP 的过程。算法 4 描述了如何利用索引 UI 查询当前搜索分支的有效邻居的过程, 其在算法 6 中被调用。

5 实验

5.1 实验设置

所有实验都运行在一台配置了 20 个型号为 Intel Xeon Gold 5218R 的 CPU 的 Linux 服务器上。该服务器内存大小为 1 TB, 操作系统为 64 位 Ubuntu 20.04.6。所有算法都使用 C/C++ 实现, 并使用 g++ 9.4.0 进行编译。所有的数据集都预先加载到内存中, 忽略磁盘 I/O 的时间。

数据集: 为了检验本文算法在实际应用中的性能, 选择了 10 个不同种类的真实数据集开展实验, 包括社交网络和生物网络。这些数据集全部来自 SNAP¹⁾ 和 NetworkRepository²⁾。其详细信息如表 3 所列。使用一个超过 10 亿条边的数据集 TM 来评估算法的可扩展性。为了得到概率图, 使用均匀分布为图上的每一条边分配一个范围为 0.5~1 的概率值。

查询: 为每个数据集随机生成 1000 条查询, 每条查询保证至少存在一条有效路径。

指标: 记录每个算法的查询时间(Query Time), 即一个查询开始到结束的总时间, 单位为 ms。若无特殊说明, 则最终

¹⁾ <http://snap.stanford.edu/data/>

²⁾ <http://networkrepository.com/networks.php>

使用算法平均值来表示一组查询的查询时间。如果查询时间超过 1000 s, 则用 INF 表示。

对比算法: 实验的几种对比算法如下。

1) PIEnum: 本文提出的算法。

2) PIEnum⁺: 基于 Join 模式^[21]的 PIEnum 算法。PIEnum⁺ 首先使用 PIEnum 枚举长度不超过 $k/2$ 的左边部分路径, 并将这些路径保存起来。然后使用 PIEnum 从每条左边部分路径的右端点枚举长度不超过 $k/2$ 的右边部分路径。最后将左右两部分路径根据中间点连接, 并过滤包含重复点的路径, 最终得到所有有效路径。

3) UnEnum: 文献^[27]提出的算法。

4) UnEnum⁺: 基于 Join 模式的 UnEnum 算法。其思想和 PIEnum⁺ 一致。

表 3 数据集详细信息

Table 3 Detailed information of datasets

数据集	缩写	顶点数	边数	平均度数	类型
Skitter	SK	1.69×10^6	1.109×10^7	13.09	Mixed
Web-google	GG	9.16×10^5	5.100×10^6	11.10	Web
Baidu-baike	BK	4.15×10^5	3.300×10^6	15.80	Web
Soc-Epinions1	EP	7.60×10^4	5.090×10^5	13.40	Social
Slashdot0922	SL	8.20×10^4	8.700×10^5	21.20	Social
Bio-grid-yeast	YE	6.00×10^3	3.140×10^5	104.50	Biologic
LiveJournal	LJ	4.80×10^6	6.850×10^7	28.30	Social
WikiTalk	WT	2.40×10^6	5.000×10^6	4.20	Mixed
Web-uk-2005	UK	3.95×10^7	9.213×10^8	46.70	Web
Twitter-mpi	TM	5.20×10^7	1.960×10^{21}	74.70	Mixed

5.2 实验结果分析

实验 1 整体比较。图 4 展示了所有算法在不同数据集上的总体性能比较结果, 其中 $k=8, \gamma=0.8$ 。从图中可以看出, 不同数据集上的总体查询时间变化范围较大, 最短不到 1 ms, 最长为 1000 s。总体上看, PIEnum 在所有数据集上的查询时间都比 UnEnum 快 10 倍以上, 表明 PIEnum 的剪枝和索引技术发挥了较好的性能。同时, PIEnum 在所有数据集上的性能都优于 UnEnum⁺。在数据集 SL 和 YE 上, PIEnum⁺ 的查询时间比 PIEnum 快 5~10 倍, 这是因为 Join 模式在稠密图上能够更好地发挥性能。对于最大的数据集 TM, UnEnum 和 UnEnum⁺ 都有超时问题, 而 PIEnum 和 PIEnum⁺ 的查询时间分别约为 500 s 和 100 s。这表明, PIEnum 和 PIEnum⁺ 有较好的可拓展性。

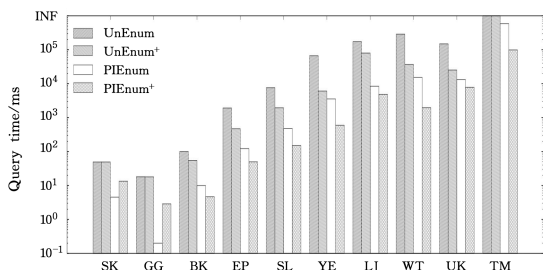


图 4 各算法在所有数据集上查询时间的总体比较

Fig. 4 Overall comparison of all algorithms on different graphs

实验 2 调节 k 。为了研究当 k 变化时各算法的性能, 选择 4 个有代表性的数据集 (EP, LJ, UK, TM) 开展实验, 结果

如图 5 所示, 其中 k 的变化范围为 3~8, $\gamma=0.8$ 不变。可以看出, 当 k 增大时, 所有数据集上的查询时间都增加。这是因为 k 越大, 路径枚举的搜索空间越大, 导致了更大的时间开销。总体上, PIEnum 的查询时间比 UnEnum 快 10 倍左右。这表明, 和 UnEnum 相比, PIEnum 的搜索空间更小且能够剪枝更多的无效路径。此外, 随着 k 的增大, PIEnum 的查询时间曲线比 UnEnum 更平缓。这表明, PIEnum 具有较好的可扩展性。当 k 大于 6 时, PIEnum⁺ 比 PIEnum 的表现更好。这是因为 k 越大, 枚举的路径数量越多, 越有利于发挥 Join 模式的优势。

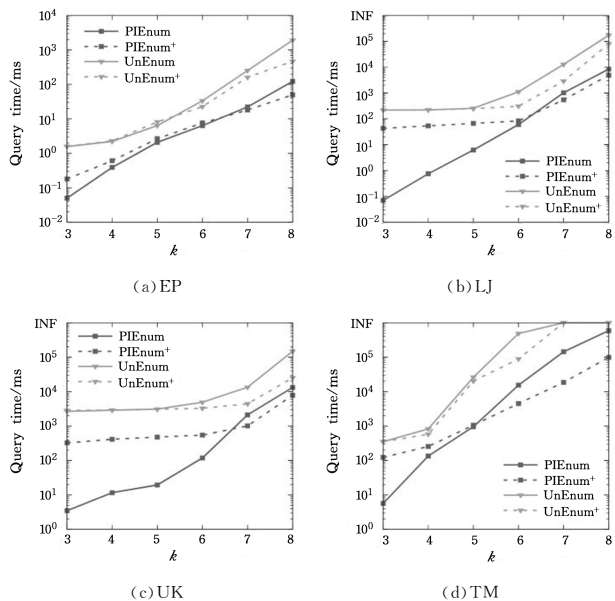


图 5 调节 k 时各模型在 4 个数据集上的查询时间对比

Fig. 5 Comparison of query time of each model on four datasets when tuning k

实验 3 调节 γ 。与实验 2 类似, 研究当 γ 变化时各算法的性能, 结果如图 6 所示, 其中 γ 的变化范围为 0.7~0.9, $k=8$ 不变。总体上, 随着 γ 的增大, 各算法的查询时间都缩短。这是因为 γ 越大, 概率阈值的约束越严格, 枚举的路径越少, 所需的时间越短。PIEnum 和 PIEnum⁺ 的查询时间同样比其他两个算法快 10 倍左右。随着 γ 的增大, 各算法的变化规律和实验 2 相反, 这里不重复解释。

实验 4 查询时间分解。图 7 展示了各算法的查询时间分解后的情况, 其中 $k=8, \gamma=0.8$ 。Pre-enum 表示剔除无效顶点和构建索引的时间之和, enum 表示路径枚举的时间。可以看出, 各算法的 Pre-enum 时间非常接近。这表明 PIEnum 构建索引的时间非常短。需要说明的是, BFS 占据 Pre-enum 阶段的绝大部分时间。此外, PIEnum 的 Pre-enum 时间只占总查询时间的很小一部分, 却能有效提升查询的性能。这表明本文在路径枚举前提出的方法能够有效地加速查询。

实验 5 99% 延迟查询时间比较。为了更深入地验证 PIEnum 的效率, 在 k 变化的情况下, 比较 PIEnum 和 UnEnum 的 99% 延迟查询时间 (99% 查询完成所需的时间), 结果如图 8 所示。可以看出, 在各个数据集上, PIEnum 的时间开销明显小于 UnEnum。对于 EP 这样的常规图, 99% 的查询能够

在 0.1 s 内完成。这表明,PIEnum 能够很好地应对常规查询。

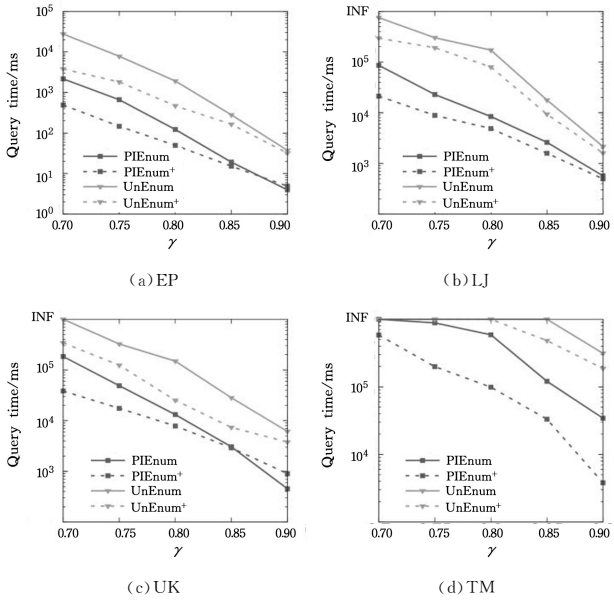


图 6 调节 γ 时各模型在 4 个数据集上的查询时间对比

Fig. 6 Comparison of query time of each model on four datasets when tuning γ

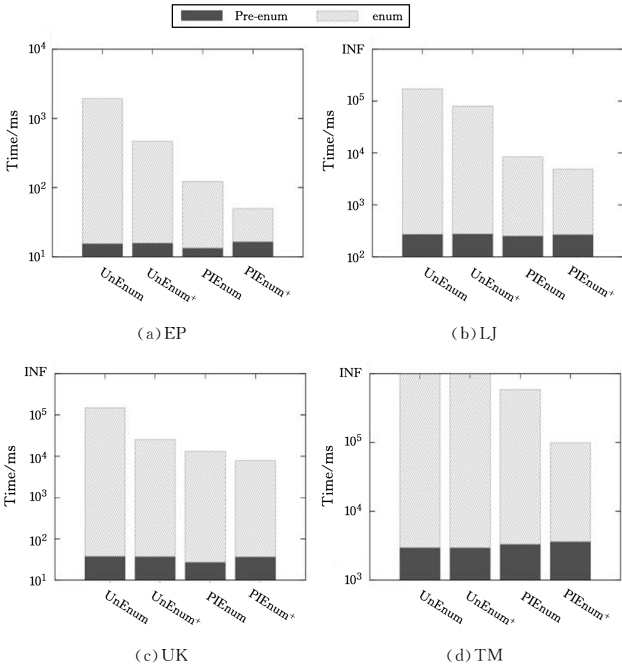


图 7 查询时间分解

Fig. 7 Query time decomposition

实验 6 具体指标对比。为了对比 PIEnum 和 UnEnum 的剪枝能力,检验了有效顶点的数量 (Actives)、结果路径的数量 (Results),以及枚举过程中访问的边数 (Edges)。实验结果如图 9 所示。可以看出,在各个数据集上,PIEnum 有效顶点的数量大约是 UnEnum 的一半,表明 PIEnum 能够更好地剔除无效顶点。同时,PIEnum 访问的边数也明显少于 UnEnum,表明 PIEnum 在枚举过程中能够更充分地对无效路径进行剪枝。在一些数据集上,当 k 较大时,UnEnum 的结果路径数量比 PIEnum 少,原因是 UnEnum

部分查询超时。

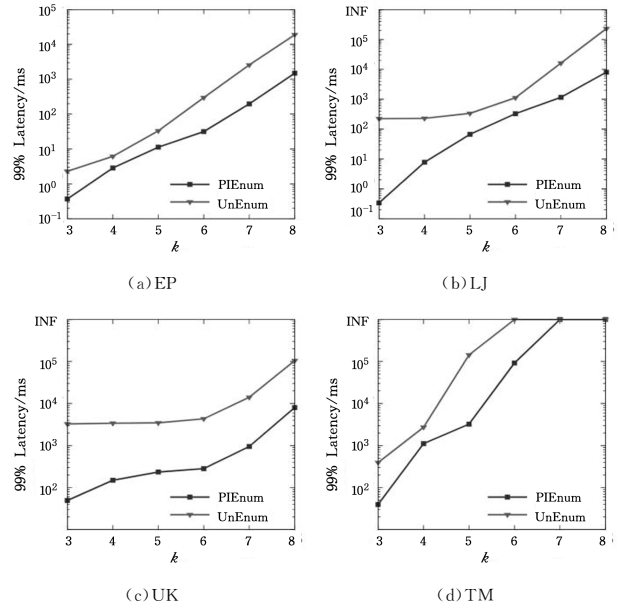


图 8 k 变化时 99% 延迟时间比较

Fig. 8 Comparison of 99% latency with k varied

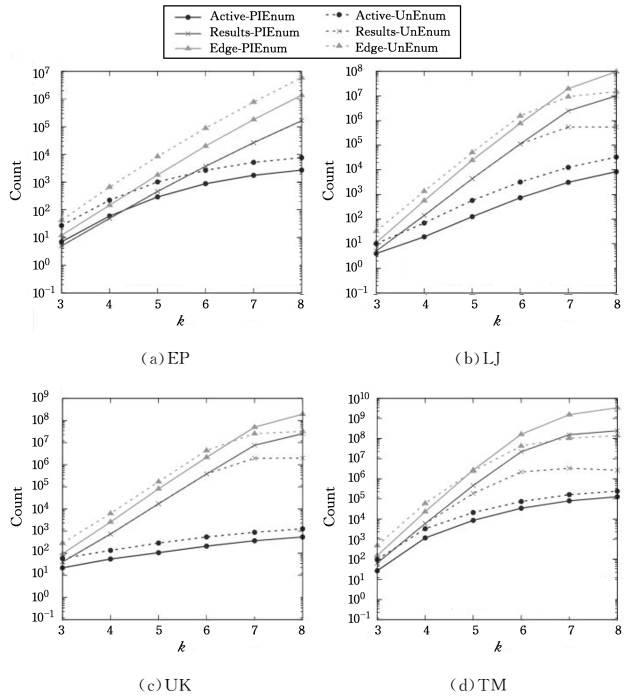


图 9 k 变化时具体指标对比

Fig. 9 Comparison of detailed metrics with k varied

实验 7 内存消耗比较。为了研究 k 变化时,PIEnum 内存消耗的情况,对比了加载图的内存消耗 (Graph)、存储结果路径的内存消耗 (Result Paths),以及构建索引的内存消耗 (Index)。

图 10 展示了在数据集 UK 和 TM 上的实验结果。可以看到,索引的内存消耗比图小得多,这是因为 S 中顶点的数量要远远小于图上所有顶点的数量。同时,当 k 较大时,索引占用的内存也比结果路径占用的内存少。这表明,PIEnum 设计的索引是轻量级的。

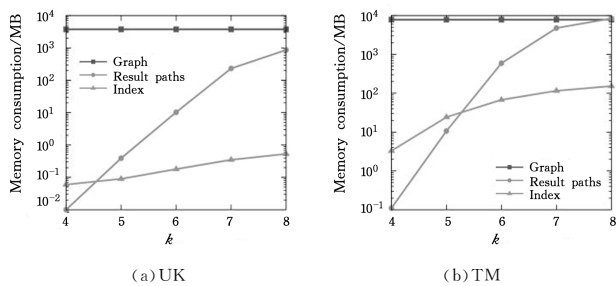


图 10 内存消耗比较

Fig. 10 Comparison of memory consumption

实验 8 使用不同分布生成概率图。为了研究不同算法在使用不同分布生成的概率图上的性能差异,本文使用均匀分布、正态分布、指数分布 3 种代表性的分布来生成概率图。通过调节参数,为数据集 EP 生成上述 3 种不同分布的概率图,其边上的概率值的分布情况如表 4 所列,其中指数分布 1 和指数分布 2 是相互对称的。随后,为上述每个概率图随机生成 1000 条查询。几种算法在不同图上的查询时间如表 5 所列。可以看到,PIEnum 的查询时间在不同分布的概率图上都比 UnEnum 快 10~20 倍。同时,PIEnum⁺的表现也比 UnEnum⁺更好。在正态分布和指数分布 1 的图上,几种算法的查询时间都较短,这是因为大多数边上的概率值比给定的概率阈值 $\gamma=0.8$ 小,所以结果路径较少。相反,在指数分布 2 的图上,几种算法的查询时间都较长,这是因为大多数边上的概率值比 $\gamma=0.8$ 大。

表 4 数据集 EP 上概率值分布情况

Table 4 Distribution of probabilities on EP (%)

分布	[0.5,0.6)	[0.6,0.7)	[0.7,0.8)	[0.8,0.9)	[0.9,1]
均匀分布	20	20	20	20	20
正态分布	10	20	40	20	10
指数分布 1	52	17	12	10	9
指数分布 2	9	10	12	17	52

表 5 各算法在不同分布产生的数据集上的查询时间对比

Table 5 Comparison of query time with different distributions on EP (ms)

分布	PIEnum	PIEnum ⁺	UnEnum	UnEnum ⁺
均匀分布	87.10	39.73	1441.60	325.13
正态分布	2.51	2.80	32.71	27.29
指数分布 1	2.03	2.59	19.44	18.19
指数分布 2	32076	6897	219757	34025

结束语 本文研究了概率图上路径长度和概率约束下的路径枚举问题,并提出算法 PIEnum 来解决已有算法时间开销较高的问题。本文设计了高效的剪枝技术进一步剔除无效顶点以及无效的搜索分支,同时设计了高效的在线索引来避免路径枚举过程中的重复计算。使用真实世界的数据集开展了全面的实验,结果显示,PIEnum 的整体性能比已有算法提升了 10 倍以上。

尽管 PIEnum 能够显著加速 UHcPE 查询,但其仍然存在提升空间。首先,大图上的查询时间还需要进一步缩短(在

TM 上,当 $k>7$ 时查询时间超过 100 s)。可能有效的解决方案是构建一个全局离线索引来缩减路径枚举前的时间开销。但此种方案具有较大的难度,因为需要占用较大的内存,且离线索引需要设计更新策略来适应动态图。其次,PIEnum⁺并非在任何时候都优于 PIEnum。设计一种评估方法在查询之前评估当前查询应该使用哪种算法枚举路径是一个有意义的研究方向。评估方法的执行时间应该明显短于查询的总时间。

参考文献

- [1] LIU H, JIN C, YANG B, et al. Finding top-k shortest paths with diversity[C]//ICDE. IEEE Computer Society, 2018;1761-1762.
- [2] OUYANG D, YUAN L, QIN L, et al. Efficient shortest path index maintenance on dynamic road networks with theoretical guarantees[J]. VLDB Endow, 2020, 13(5):602-615.
- [3] PENG Y, YU J X, WANG S. PSPC: efficient parallel shortest path counting on large-scale graphs[C]//ICDE. IEEE, 2023; 896-908.
- [4] LAI L, QING Z, YANG Z, et al. Distributed subgraph matching on timely dataflow[J]. VLDB Endow, 2019, 12(10):1099-1112.
- [5] WANG K, LIN X, QIN L, et al. Vertex priority based butterfly counting for large-scale bipartite networks[J]. VLDB Endow, 2019, 12(10):1139-1152.
- [6] KAPUSTA J, MUNK M, SVEC P. Selection of suitable page-rank calculation for analysis of differences between expected and observed probability of accesses to web page[C]//MIWAI. Springer, 2018;139-150.
- [7] LIBEN D, KLEINBERG J M. The link-prediction problem for social networks[J]. JASIST, 2007, 58(7):1019-1031.
- [8] YAN J, WANG W, WU Z, et al. An uncertain graph method based on shuffle model to preserve link privacy of mobile social networks[J]. JISE, 2024, 40(1):125-150.
- [9] SEVON P, ERONEN L, HINTSANEN P, et al. Link discovery in graphs derived from biological databases[C]//DILS. Springer, 2006;35-49.
- [10] KIMURA M, SAITO K. Tractable models for information diffusion in social networks[C]//PKDD. Berlin; Springer, 2006;259-271.
- [11] SAHA A, BROKKELKAMP R, VELAJ Y, et al. Shortest paths and centrality in uncertain networks[J]. VLDB Endow, 2021, 14(7):1188-1201.
- [12] PENG Y, ZHANG Y, ZHANG W, et al. Efficient probabilistic k-core computation on uncertain graphs[C]//ICDE. IEEE Computer Society, 2018;1192-1203.
- [13] ZHOU A, WANG Y, CHEN L. Butterfly counting and bitruss decomposition on uncertain bipartite graphs[J]. VLDB J, 2023, 32(5):1013-1036.
- [14] KE X, KHAN A, QUAN L H. An in-depth comparison of s-t reliability algorithms over uncertain graphs[J]. VLDB Endow, 2019, 12(8):864-876.
- [15] DAI Q, LI R, LIAO M, CHEN H, et al. Fast maximal clique

- enumeration on uncertain graphs: A pivot-based approach[C]// SIGMOD. ACM, 2022: 2034-2047.
- [16] YASUDA N, SUGAYA T, MINATO S. Fast compilation of s-t paths on a graph for counting and enumeration[C]// AMBN. PMLR, 2017: 129-140.
- [17] BIRMELE E, FERREIRA R A, GROSSI R, et al. Optimal listing of cycles and st-paths in undirected graphs[C]// SODA. SIAM, 2013: 1884-1896.
- [18] NISHINO M, YASUDA N, MINATO S, et al. Compiling graph substructures into sentential decision diagrams[C]// AAAI. Springer, 2017: 1213-1221.
- [19] GROSSI R, MARINO A, VERSARI L. Efficient algorithms for listing k disjoint st-paths in graphs[C]// LATIN. Springer, 2018: 544-557.
- [20] RIZZI R, SACOMOTO G, SAGOT M. Efficiently listing bounded length st-paths[C]// IWOCA. Springer, 2014: 318-329.
- [21] PENG Y, ZHANG Y, LIN X, et al. Hop-constrained s-t simple path enumeration: Towards bridging theory and practice[J]. VLDB Endow, 2019, 13(4): 463-476.
- [22] QIU X, CEN W, QIAN Z, et al. Realtime constrained cycle detection in large dynamic graphs[J]. VLDB Endow, 2018, 11(12): 1876-1888.
- [23] LAI Z, PENG Y, YANG S, et al. PEFP: efficient k hop constrained s-t simple path enumeration on FPGA[C]// ICDE. IEEE, 2021: 1320-1331.
- [24] HAO K, YUAN L, ZHANG W. Distributed hop-constrained s-t simple path enumeration at billion scale[J]. VLDB Endow, 2021, 15(2): 169-182.
- [25] SUN S, CHEN Y, HE B, et al. Pathenum: Towards real-time hop constrained s-t path enumeration[C]// SIGMOD. ACM, 2021: 1758-1770.
- [26] ZHANG J, YANG S, OUYANG D, et al. Hop constrained s-t simple path enumeration on large dynamic graphs[C]// ICDE. IEEE, 2023: 762-775.
- [27] LI X, HAO K, YANG Z, et al. Hop-constrained s-t simple path enumeration in large uncertain graphs[C]// ADC. Springer, 2022: 115-127.



XIE Wenlin, born in 1997, postgraduate. His main research interest is query processing on large scale graph data and optimization.



DU Ming, born in 1975, Ph.D, professor. His main research interests include natural language processing and information analysis.

(责任编辑:何杨)