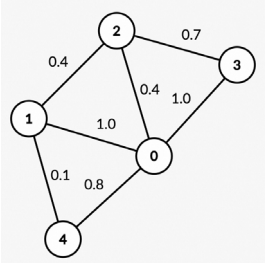
2023 大作业——阅读材料

1. 前言

1.1 问题阐释

给定一个带权图（edge-weighted graph）$G(V,E,w)$。V是顶点的集合，E是边的集合，w是一个函数，它映射每条边$(u,v) \in E$在一对u和v顶点之间到正值$w(u,v)$，我们称之为边的权重。设$n = |V|$为顶点数，$m = |E|$为边数，而$N(v)$为v的相邻顶点的集合。我们把一条路径的**跳数（hop）**看作是这条路径上的边数。给定一个自然数$k \in \mathbb{N}$，则***k*-hop-constrained 路径**是指跳数不大于k的路径。我们使用$d_k(v_i, v_j)$表示一对顶点v_i和v_j之间的最短***k*-hop-constrained 距离**，它表示v_i和v_j之间任意k-hop-constrained 路径中边的总权值的最小值。此外,我们使用$P_k(v_i, v_j)$来表示一个v_i和v_j之间的最短k-hop-constrained 路径，这是一个对应$d_k(v_i, v_j)$的路径。例如，在下面的图中，顶点 0 到顶点 1 间的最短 1-hop-constrained 距离为 1.0，对应的 1-hop-constrained 的最短路径为$\{(0,1)\}$。如果我们将跳跃约束设置为 2，则最短距离变为 0.8，对应的最短路径为$\{(0,2),(2,1)\}$。我们关注以下 hop-constrained 的最短距离（或路径）问题。



问题 1 给定一个带权图$G(V,E,w)$和一个跳数上界$k \in \mathbb{N}$，hop-constrained 的最短距离（或路径）问题是查询图中任意一对顶点距离（或路径）问题的k-hop-constrained 最短距离（或路径）。

查询 hop-constrained 的最短距离或路径对于检索关系数据库中的信息、检测电子商务交易网络中的欺诈行为以及通信网络中的路由数据非常有用。在本文中，我们利用索引方法来有效地执行此任务。

1.2 Hop-constrained 标签

经典的 2-hop 标签使我们能够有效地查询顶点之间 hop-constrained 的最短距离和路径，在过去的十年中得到了广泛的研究。最近，Zhang 等人将 hop-constrains 合并到经典的 2-hop 标签中，用于查询跳约束的最短距离和路径。我们将得到的 hop-constrained 标签介绍如下。对于每个顶点$v \in V$，有一个标签集$L(v)$。每个$L(v)$中的标签是一个三元素的元组$\left(u, h, d_h(v,u)\right)$，其中$u$是顶点，称为$v$的中心（hub）；$h$是跳数上界；$d_h(v,u)$是$u$和$v$之间的$h$-hop-constrained 的最短距离。我们使用L来表示跳跃约束标签的总集合，即$L = \{L(v)|\forall v \in \mathrm{V}\}$。我们让$L$满足 hop cover constraint。

定义 1　L对于跳数上限$k \in \mathbb{N}$满足 **hop cover constraint**，当且仅当对任意两个顶点v_i和v_j，如果有v_i和v_j之间的k-hop-constrained 路径，就有两个标签$\left(u, h_1, d_{h_1}(v_i,u)\right) \in L(v_i)$和$\left(u, h_2, d_{h_2}(v_j,u)\right) \in L(v_j)$ 使得：(i)$h_1 + h_2 \le k$；(ii)$d_{h_1}(v_i,u) + d_{h_2}(v_j,u) = d_k(v_i,v_j)$，这表明公共中心$u$在$v_i$和$v_j$之间的一个$k$-hop-constrained 最短路径上。

有了满足 hop cover constraint 的一组标签L，我们可以使用以下公式查询$d_k(v_i,v_j)$：

$$d_k(v_i,v_j) = \min_{\substack{(u,h_1,d_{h_1}(v_i,u))\in L(v_i)\\(u,h_2,d_{h_2}(v_j,u))\in L(v_j)\\h_1+h_2\le k}} d_{h_1}(v_i,u) + d_{h_2}(v_j,u). \qquad (1)$$

此外，通过将前导（predecessor）信息合并到标签中，我们还可以递归地查询v_i和v_j之间最短的k-hop-constrained 路径中的每条边。具体地说，我们将$L(v)$中的每个标签扩展为四元素的元组$(u, h, d_h(v,u), p_{vu})$，其中p_{vu}的前身（predecessor）是v和u之间的h-hop-constrained 最短路径。使用这种标签递归查询k-hop-constrained 最短路径的示例如下。在图 1 中，我们假设$(0,2,0.8,2) \in L(1)$，$(0,0,0,0) \in L(0)$。我们可以用这两个标签得到 0 到 1 之间的最短距离。由 1 的 predecessor 可知，顶点 1 对应的最短路径上的下一个顶点是 2，所以边$(0,2)$一定在最短路径上。然后我们只需要找到 1 和 2 之间的路径，当然是$(1,2)$因此，最短路径是$\{(0,2),(2,1)\}$。

2 THE HSDL ALGORITHM

据我们所知，HBLL算法是唯一现有的生成跳约束标签的方法。我们注意到，HBLL在实践中可能没有足够高的效率。其主要原因是，它可能会在多线程环境中生成大量的冗余标签。为了解决这个问题，我们提出了跳约束最短距离标记（HSDL）算法，该算法可以有效地生成并行的最小标签集。

HSDL算法的细节

该算法接受一个带有边权的图 $G(V, E, w)$ 和一个参数 K，表示可能查询的跳数上限的最大值。首先，它初始化一个空的标签集合 L_{temp}（第1行）。假设顶点按度数从大到小排序，即 $V = v_1, \ldots, v_{|V|}$。用 $r(v_i)$ 表示 v_i 的排名，其中 $deg(v_i)$ 是 v_i 的度数。算法按顶点排名的降序将每个 $v_i \in V$ 推入一个线程池，以并行和大致顺序地处理每个 $v_i \in V$（第2行）。

Algorithm 1: The HSDL Algorithm

Input: a graph $G(V, E, w)$ and the maximum hop upper bound K

Output: a set L_{HSDL} of hop-constrained labels

1. $L_{temp} = \emptyset$
2. for each sorted vertex $v_i \in V$ in parallel do
    3. Initialize Q that contains $(v_i, 0)$ with the priority of 0
    4. while $Q \neq \emptyset$ do
        5. Pop (u, h_u) out of Q with the priority of d_u
        6. if $r(v_i) \geq r(u)$ then
            7. if $Query(u, v_i, h_u) > d_u$ then
                8. Insert (v_i, h_u, d_u) into $L_{temp}(u)$
            9. if $h' = h_u + 1 \leq K$ then
                10. for each vertex $v \in N(u)$ do
                    11. $d_v = d_u + w(u, v)$
                    12. if $d_v < Q((v, h')).$priority then
                        13. Push (or update) $(v, h')|d_v$ into Q
                    14. end if
                15. end for
            16. end if
        17. end if
    18. end while
    19. end if
20.end for

Output: $L_{HSDL} \leftarrow \text{sort } L_{temp}$

HSDL通过类似于Dijkstra的搜索来生成以 v_i 为枢纽顶点的标签。首先，它初始化一个最小优先队列 Q，其中包含一个二元组 $(v_i, 0)$，其优先级为0（第3行）。当 $Q \neq \emptyset$ 时（第4行），它弹出队列顶部的二元组 (u, h_u)，其优先级为 d_u（第5行）。它尝试将一个标签 (v_i, h_u, d_u) 插入到 $L_{temp}(u)$ 中，仅当 $r(v_i) \geq r(u)$ 时才执行这个插入（第6行）。此外，它使用 L_{temp} 查询 u 和 v_i 之间的 h_u 跳数约束的最短距离。如果查询的距离大于 d_u（第7行），则将标签 (v_i, h_u, d_u) 插入到 $L_{temp}(u)$ 中（第8行）。之后，如果 $h' = h_u + 1 \leq K$（第9行），它尝试将新的元素推入 Q，如下所示。对于每个顶点 $v \in N(u)$（第10行），它计算 $d_v = d_u + w(u, v)$（第11行）。如果 d_v 小于 Q 中二元组 (v, h') 的优先级（第12行，如果 (v, h') 不在 Q 中，则优先级为 ∞），则将（或更新）(v, h') 推入 Q，其优先级为 d_v（第13行）。

HSDL算法的示例：

以图1为例，经过HSDL算法后，生成的标签如下，格式为 $(vertex, distance, hop-constraint, predecessor)$。图2显示了生成的标签。