III. 背景

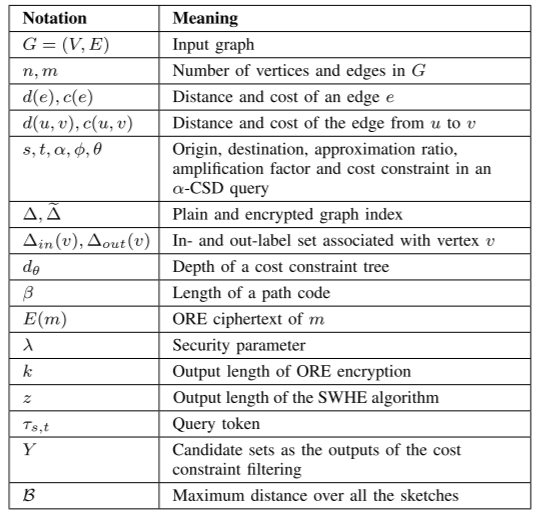
本节给出了CSD查询问题的正式定义，并介绍了用于图形查询的2HCLI结构。

A.近似CSD查询

设G = (V, E)为顶点集V和边集E的有向图（如果没有特别声明，后文提到的图即为有向图）。每条边e ∈ E对应距离d(e) ≥ 0和开销c(e) ≥ 0.

我们把开销c(e)作为约束条件。我们把连接两个顶点的一组边表示为一条路径。对于路径

P= (e1, e2,..., ek )，它的距离d(P) 定义为d(P)= ，即从起点到终点的距离。类似的，我们定义P的开销为c(P) =.本文中的符号在表I中进行了总结。

表Ⅰ 符号列表

给定一个图G，起点s ∈ V，终点t ∈ V ，且开销限制为θ，CSD查询就是找到s和t之间的最短距离d，并且总开销不超过θ. 由于CSD查询问题已被证明是NP-hard，因此我们与已有的解决方案保持一致，本文重点提出一个近似的CSD解决方案。

受未加密图上的近似最短路径查询的通用定义的启发，我们定义近似CSD查询（即α-CSD查询）如下。

定义1（α-CSD查询）：给定起点s和终点t，开销限制θ和近似度α，一个α-CSD查询返回路径P的距离d(P)，使得c(P) ≤ θ 且d(P) ≤ α · d（opt），d（opt）是一个精确CSD查询的最优解，这个精确CSD查询具有同样的起点s，终点t和限制θ.

图1显示了一个简单的图，它有五个顶点，每条边的距离和开销都在它旁边标记。设起点为a，终点为c，开销限制θ=4，精确CSD查询返回的最短距离d（opt） = 6，对应的路径是s (a, b, c). 对于近似度α = 1.5的α-CSD查询（参数不变，即起点为a，终点为c，θ=4），其中一个有效解是8，对应路径为Pα = (a, e, b, c).这是因为d(Pα) = 8 < α · d（opt） = 9 且c(Pα) = 3 < θ.

基于上述定义，对于相同的起点和终点，给定两个路径P1和P2，如果c(P1) ≤ c(P2) 且d(P1) ≤ α · d(P2)，我们称P1 α优于P2.利用这一原理，我们可以显著降低图索引的构造复杂度，因为索引中的大量冗余项可以被过滤掉。我们将在下面的小节中作进一步的说明。

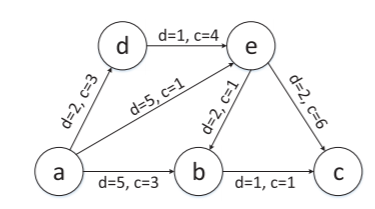


图1. α-CSD查询图示例

B.构建标签索引

本文设计的加密索引主要是基于众所周知的2HCLI构造的，2HCLI是一种特殊的数据结构，能够有效地支持最短距离查询。现在我们简要介绍2HCLI的基本思想，并说明它在构建受限标签索引中的应用。

给定图G = (V, E)，顶点集为V，边集为E，任意顶点v ∈ V都跟一个入标签集△in(v)和出标签集△out(v)相关联。

△in(v)中的每个实体对应顶点u到v的最短路径，u ∈ V.这意味着从u到v有一条或多条通路，但v不一定是u的邻居或两跳邻居。类似的，△out(v)中的每个实体代表从v到V中另一顶点u的最短路径。为了回答从起点s到终点t的最短距离查询，我们首先在标签集△out(s)和△in(t)中找出公共点，然后找出s到t的最短距离。注意△in(v)和△out(v)中的实体必须认真选取，以确保任意两个顶点s和t之间的距离都可以通过△out(s)和△in(t)计算出来。

考虑图1中的图，如果我们忽略边的开销限制，以a为起点，c为终点，基本的无限制最短距离查询就可以通过2HCLI的帮助来解决，详见图2。给定标签集△out(a)和△in(c)，就容易找出公共点的集合，里面包含点b和e.这个基础最短距离查询的最终答案是5，因为d(a, e) + d(e, c) = 5 < d(a, b) + d(b, c) = 6.

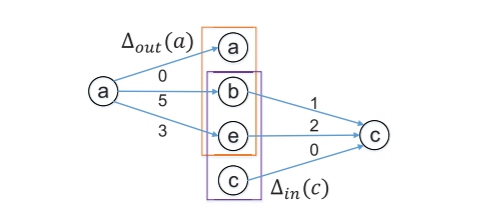


图2. 一个基于2HCLI的基础最短距离查询示例。箭头旁边每个实体d表示从起点到终点的最短距离，如a到e的最短距离为3。

虽然为只有距离条件的图构造2HCLI是简单而直接的，但是为CSD查询构造基于2HCLI的标签索引要复杂得多。这是因为在CSD查询中，边的限制条件有两种，因此在△in(v)和△out(v)标签集中，任意两点之间都可能有多种距离和开销的组合。为了便于说明，我们还以图1中的图和CSD查询为例。对应的2HCLI如图3所示，其中每个箭头旁边的二元组表示从起点到终点的距离和开销。注意，在图2的最短距离查询中，从a到c的最短距离是唯一的，就是经过e的那一条。然而，在图3所示的CSD查询设置中，按照不同的开销，通过e从a到c有四种可能的距离。由于开销条件的存在，在大尺度图中，每对顶点之间距离的数目可能显著增加，这使得构建2HCLI和解决CSD查询问题变得更加复杂。

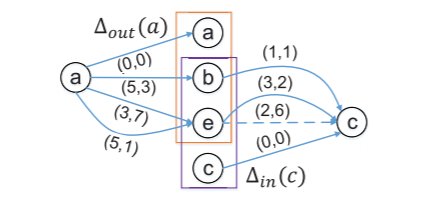


图3. 精确CSD查询的2HCLI示例。箭头旁边的每个实体(距离，开销)分别表示距离和开销。开销限制θ = 4 时，从a到e的最短距离是5.

为了提高查询效率，我们采用了离线过滤和在线过滤相结合的方法。

离线过滤的目的是降低2HCLI的结构复杂度，并尽可能地减少入标签集和出标签集中的实体数量。我们采用了[2]中提出的方法。2HCLI中的实体是精心挑选的，使得对于任何从u到v且满足限制条件θ的CSD查询，都可以只用2HCLI就可以正确解答。由于在特定的CSD查询中，2HCLI的结构应该独立于开销限制，我们可以使用“α优于”的定义去过滤入和出标签集中的冗余实体。

以图3为例，α = 1.5，从e到c的实体有两个，路径P1ec = (e, b, c)，距离-开销二元组为（3，2），路径P2ec = (e, c)，距离-开销二元组为（2，6），P1ec α优于P2ec. 因此，P2ec对应的实体可以被过滤掉（如虚线箭头所示），这有助于减少△in(c)中的实体数量。得到的2HCLI如图4所示。我们请读者参考[2]以获得更多的构造细节。

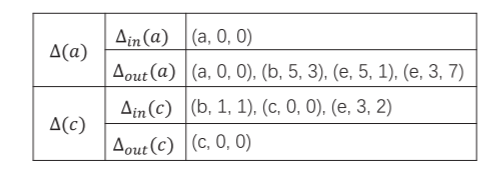


图4. 对图3中原始2HCLI进行离线过滤后得到的2HCLI.实体(u, d, c)中各项分别表示顶点标识，距离和开销。这个近似CSD查询（起点为a，终点为c，α = 1.5, θ = 4）的解是6，恰好是精确CSD查询的解。

在线过滤的目的是为给定的CSD查询选择可能有效的答案，并且仅基于2HCLI。例如，给定从a到c的α-CSD查询，限制条件θ = 4，我们可以先找到△out(a)和△in(c)的公共点集V’。对于每个v ∈V’，返回c(a,v)+c(v, c) ≤ θ条件下d(a,v)+d(v, c)的最小值。由于上述比较需要与相应的密文一起执行，因此在第六节将设计一种有效的在线过滤方法。