TimeReach：关于演化图的历史可达性查询

总结

**一、应用背景**

本文主要解决 在随着时间演变由多个快照组成的图结构中，回答两个顶点在给定时间区间内的两种可达性查询：要么在间隔的整个持续时间内都可达（合取查询），要么在间隔中的至少一个时刻内可达（析取查询）。

**二、解决思路**

1. 前人的方法

①纯遍历无索引的方法（INS）

维护与任何快照相对应的索引。 查询将通过BFS或者DFS遍历时间区间内的每一个快照。

存在的问题：遍历的方法查询会导致的非常高的查询延迟。

②索引所有快照（IAS）

使用基于区间的索引来索引每个快照，并使用区间是否包含的条件判别是否可达。

存在的问题：这种简单方法存在许多缺点。首先，索引每个快照的计算开销将非常高，因为它需要遍历每个快照。其次，由于需要保存每个快照的索引值，因此存储开销也会很高。随着层次结构的增加和更频繁的变化，计算和存储成本都会加剧。第三，在快照的某些部分上可能只有很少的查询，在这种情况下，每个版本的索引在存储和计算方面都是浪费的。但是，还应该注意的是查询分发是不知道的。第四，使用这种方法，使用可达性测试框架的应用程序几乎无法控制索引成本与查询效率之间的权衡。换句话说，应用程序无法调整系统以减少索引开销，即使它们可以容忍查询延迟的小幅增加。

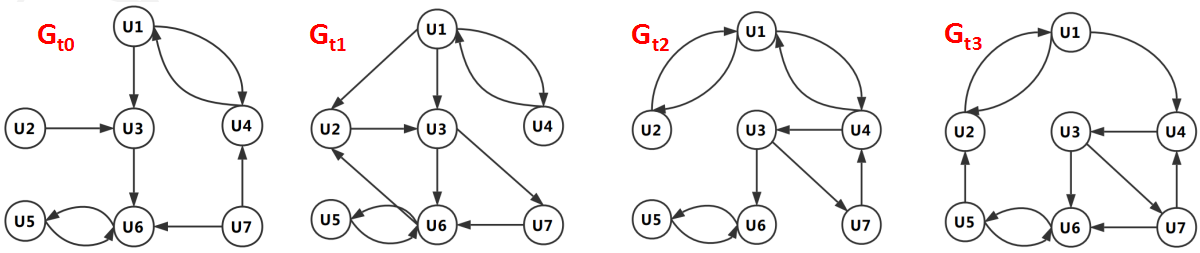
③SCISSOR

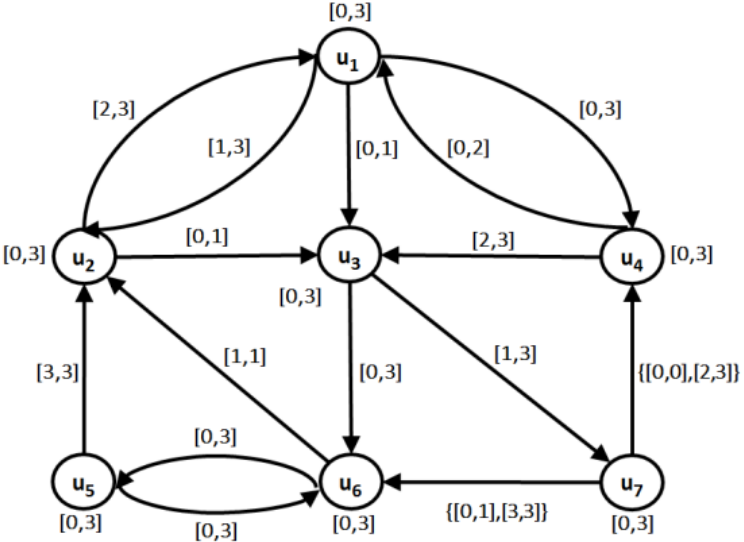
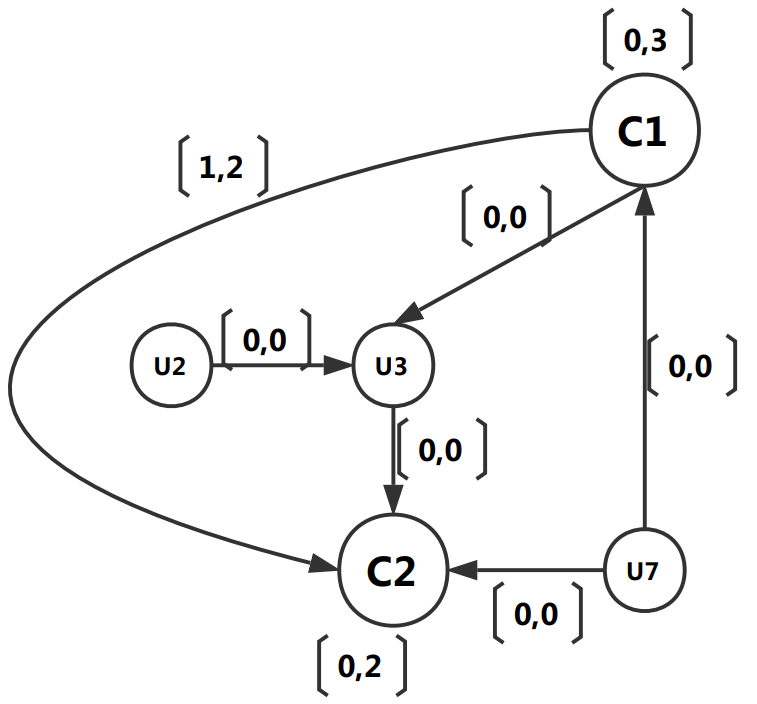
该方法的主要思想： 通过利用临时附近的索引快照的索引并分析两个快照之间发生的结构更改来回答对非索引快照的查询。

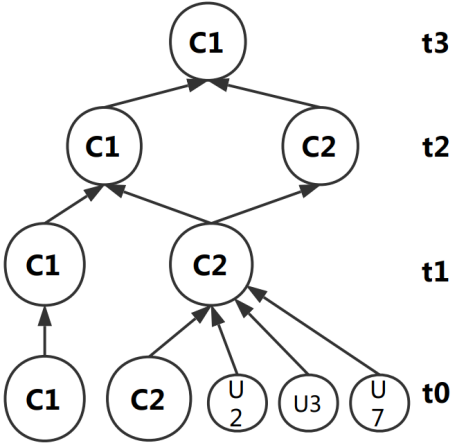
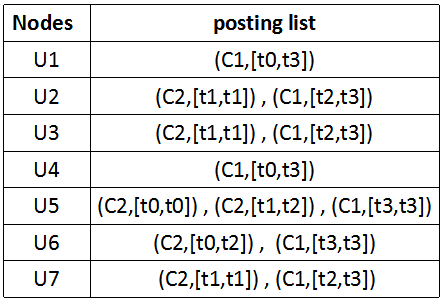
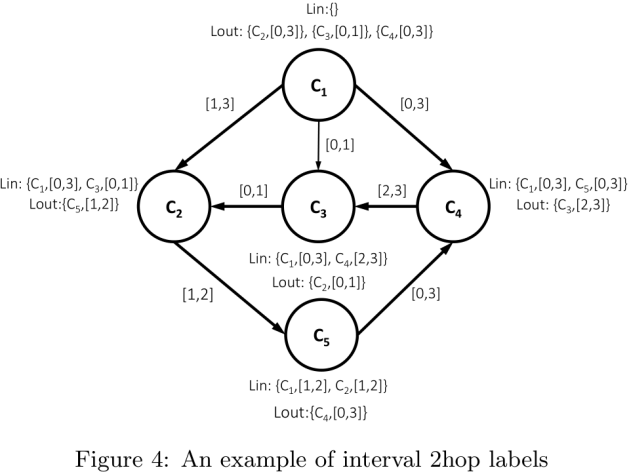
存在的问题：该方法只适用于层次结构（树状结构）

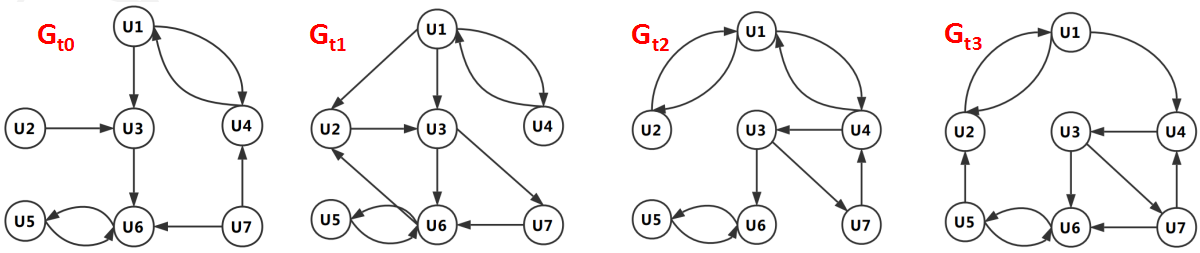
1. 作者的方法

作者提出了两种方法：1、将所有快照中的顶点和边压缩成一个大的稠密图（我们称之为Version Graph）该图中顶点和边的权重分别记录该顶点和边存在于哪些快照（生存周期）。再根据强连通分量，将一些区域内的点压缩成超点，把维护点与点之间的关系变成维护强连通分量之间的关系。2、利用强连通分量的关系，将不同时段快照之间顶点所属强连通分量的关系，将图的信息建立成一个二维表的维护。根据每个顶点在不同时刻所属强连通分量的关系来进行对两种不同查询的回答。最后提出了两种优化，1、就是将前两种方案结合使用；2、在用强连通分量压缩过后的Version Graph上加入2-hop点的维护，加速查询。









**c6**

**c4**

**c1**

**c2**

**c3**

**c5**

**c7**