主存高效的公交网络路径规划索引

马 慧[1],汤 庸[2],何怀文[1]
(1. 电子科技大学中山学院计算机学院,广东中山 528402; 2. 华南师范大学计算机学院,广东广州 510631)

摘 要: 在公交时间表下给定起始和目标站点,路径规划查询返回一组到达时间早和换乘次数少的帕雷托最优路径.现有的索引方法需要大量运行时内存.本文提出主存空间高效的索引方法(a-)PAINT. (a-)PAINT对每个站点v预计算一组标签,使得对于从站点s到站点d的查询可以通过匹配s和d相关的标签高效地生成查询结果的一条路径. PAINT对任意查询返回最优路径. a-PAINT只需要很小的预处理开销,但可能返回多一趟换乘的次优路径.用真实的公交时间表与模拟查询测试,PAINT具有合理的预处理开销. a-PAINT需要更少量的预处理开销,在大规模公交网络下准确率达90%.

关键词: 路径规划; 索引; 公交网络; 时间表; 换乘次数
中图分类号: TP392 **文献标识码:** A **文章编号:** 0372-2112(2021)11-2273-06
电子学报URL: http://www. ejournal. org. cn **DOI:** 10. 12263/DZXB. 20200436

Memory Efficient Index for Route Planning in Public Transportation Networks

MA Hui[1], TANG Yong[2], HE Huai-wen[1]
(1. *School of Computer Science, University of Electronic Science and Technology of China, Zhongshan Institute, Zhongshan, Guangdong* 528402, *China*;
2. *School of Computer Science, South China Normal University, Guangzhou, Guangdong* 510631, *China*)

Abstract: Given a source stop and a destination stop in a transportation timetable, the route planning query returns a Pareto set of paths that optimized in both earlier arrival time and less transfer times. Current indexing methods require large runtime memory. This paper proposes a memory space efficient index method (a-)PAINT. (a-)PAINT pre-computes, for each stop v, a set of labels, such that, for any query from stop s to stop d, we can efficiently retrieve a path of the query result from the label sets of s and d. PAINT could return optimal paths for every query. a-PAINT incurs far less preprocessing overheads, but might return sub-optimal path that takes one more transfer. We experimentally conducted simulated queries on real timetables. PAINT incurs reasonable preprocessing overheads. a-PAINT requires far less preprocessing overheads, and it has accurate rate up to 90% in large timetables.

Key words: route planning; index; public transportation network; timetable; transfer times

1 引言

在公交网络的公交运行时间表下,输入出发和目标地点,路径规划查询返回多条到达时间早和换乘次数少的帕累托最优路径. 一种求解方法是直接在时间表上搜索[1,2],需要访问大量的站点,不满足地图服务在线搜索的效率要求. 近年,研究者把大规模图查询的高效索引方法应用在路径查询中. 由于两个站点之间存在多条帕累托最优路径,这需要大量的空间记录路径信息. 已有的索引方法在构建索引的时候需要消耗大量内存记录辅助数据,不能在一般配置的机器上运行大数据集.

本文提出一种时空开销合理的构建索引方法(a-)PAINT(Paths optimized in Arrival tIme and Number of Transfers). (a-)PAINT是索引PAINT和a-PAINT的统称. 基于PAINT的查询返回最优路径; 基于a-PAINT的查询可能返回多一趟换乘的路径. 用真实公交时间表实验验证了(a-)PAINT预处理的时空开销和查询效率,并测试了a-PAINT的准确率.

收稿日期:2020-05-07;修回日期:2021-05-19;责任编辑:李勇锋
基金项目:国家自然科学基金(No.U1811263, No.61772211);中山市社会公益科技研究(No.2019B2006);中山市重大科技专项(No.2019A40027)

2 相关工作

rRAPTOR采用广度遍历的思想扫描公交时间表的旅程,求解到达时间早与换乘次数少的帕累托最优路径[1]. 为了提高查询效率,通常预先在线下预处理阶段对公交时间表计算一些有用的数据作为索引,然后在线上查询阶段直接在索引上搜索路径. TTL[3]仅支持最快路径查询,返回的路径的换乘次数不一定是最优的. PTL用时间扩展图[4]建模,将到达时间与换乘次数最优的查询转化成图的最短路径查询[5]. 时间扩展图的规模很大,使得PTL在计算索引的时候占用大量的运行时内存,限制了PTL不能在一般配置的机器上运行. STP划分公交网络区域,预先计算所有最优路径在区域内和区域间的换乘模式. 在查询阶段用起点到终点的换乘模式构成一张换乘图,在换乘图上用Dijkstra搜索[6]. TB将时间表中的旅程转化成图中的顶点,若两段旅程之间换乘能产生最优路径,则往图中相应的顶点之间增加一条弧,通过调用多约束标签标记的方法在图上求最优路径[7]. TB引入压缩树[8]可以节省存储空间. Phan等人在公交网络下添加一张站点之间的换乘图,允许乘客步行到换乘站点,并提出一种基于公交时间表和换乘图的索引方法[9]. Delling等人提出一种划分子图的加速查询方法[10]. 除了应用图查询方法,还有蒙特卡洛[11]、遗传算法[12]等方法求解. 文献[13]对路径规划的方法做了综述讨论.

3 问题描述

时间表$\mathcal{T}=(S,R,T)$表示公交网络. S是站点集合;R是线路集合,$r\in R$表示预先设定好的站点序列; T是旅程集合,$t\in T$表示一条线路的具体行驶时间. $\pi_{\text{arr}}(t,s)$和$\pi_{\text{dept}}(t,s)$分别表示t上的每一个站点s的到达时间和出发时间. R是T上的划分,用$\Gamma(r)$表示行驶线路r的旅程的集合. $\Gamma(r)$中的旅程满足<关系.

定义1 <与≤:设$t_1,t_2\in\Gamma(r)$. $t_1\leqslant t_2$,如果r上的所有站点s满足$\pi_{\text{arr}}(t_1,s)\leqslant\pi_{\text{arr}}(t_2,s)$且$\pi_{\text{dept}}(t_1,s)\leqslant\pi_{\text{dept}}(t_2,s)$. 如果至少有一个等号不成立,$t_1<t_2$.

旅程上相邻两个站点的连接称为基本连接. 路径是旅程片段的序列. $P=\langle s_1@t_1,s_2@t_2,\cdots s_n@t_n,s_{n+1}\rangle$表示在站点$s_i$处乘坐旅程$t_i$, 在站点$s_{i+1}$处换乘,满足约束$\pi_{\text{arr}}(t_i,s_{i+1})\leqslant\pi_{\text{dept}}(t_{i+1},s_{i+1})$. P的出发时间$\pi_{\text{dept}}(P)=\pi_{\text{dept}}(t_1,s_1)$;到达时间$\pi_{\text{arr}}(P)=\pi_{\text{arr}}(t_n,s_{n+1})$. P有n段旅程,等价于P经过n-1次换乘. $\text{tn}(P)$表示P的旅程数. 图1是一个贯穿全文的公交网络例子,表1是图1对应的时间表,未定义时间用符号#表示.

定义2 弱支配与全面支配:设P_1和P_2均是从s到d的路径. P_1弱支配P_2, 如果$\pi_{\text{arr}}(P_1)\leqslant\pi_{\text{arr}}(P_2)$且$\text{tn}(P_1)\leqslant\text{tn}(P_2)$. P_1全面支配P_2,如果P_1弱支配P_2且$\pi_{\text{dept}}(P_1)\geqslant\pi_{\text{dept}}(P_2)$.

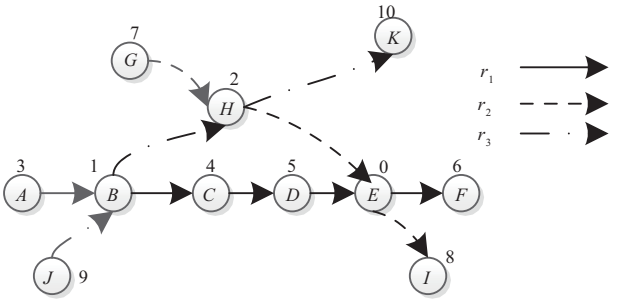

图1 一个公交网络

表1 时间表

线路	旅程	(到达时间,出发时间)					
		A	B	C	D	E	F
r_1	t_1	(#,3)	(6,7)	(12,12)	(18,18)	(21,21)	(24,#)
	t_2	(#,4)	(7,8)	(14,14)	(20,20)	(23,24)	(27,#)
	t_3	(#,6)	(9,10)	(16,16)	(21,21)	(24,24)	(27,#)
r_2		G		H		E	I
	t_4	(#,8)		(16,16)		(20,21)	(23,#)
r_3		J		B		H	K
	t_5	(#,4)		(8,8)		(15,16)	(18,#)

定义3 紧凑路径:P是紧凑的,如果不存在另一条路径P'全面支配P.

定义4 s-d-π最早到达查询:给定时间表$\mathcal{T}$、站点s和d、出发时间π(例如"现在"),查询返回从s到d的路径集合$\mathcal{P}=\{P\mid\pi_{\text{dept}}(P)\geqslant\pi$, 且不存在$\pi_{\text{dept}}(P')\geqslant\pi$的路径$P'$, P'弱支配$P\}$.

定义5 s-d-π_a-π_b范围查询:给定时间表$\mathcal{T}$、站点s和d、出发时间区间$[\pi_a,\pi_b]$,查询返回从s到d的路径集合$\mathcal{P}=\{P\mid\pi_{\text{dept}}(P)\in[\pi_a,\pi_b]$, 且$P$是紧凑的$\}$.

例1 $P_1=\langle A@t_1,F\rangle$, $P_2=\langle A@t_2,F\rangle$, $P_3=\langle A@t_3,F\rangle$, $P_4=\langle A@t_2,B@t_5,H@t_4,E@t_1,F\rangle$是从$A$到$F$的其中四条路径. A-F-4最早到达查询返回$\mathcal{P}=\{P_2,P_4\}$. $\mathcal{P}=\{P_3,P_4\}$是另一组解. 出发时间区间是[3,10]的范围查询返回$\mathcal{P}=\{P_1,P_3,P_4\}$.

4 索引框架

4.1 索引设计思想

(a-)PAINT借鉴图查询下高效的最短路径查询索引2-Hub-Labling的思想[14],对每个站点v预先计算一些标签,使得对从s到d的查询,可以在从s出发和到达d的标签中,将合适的标签连接起来生成最优路径. 假设索引中已存储标签l_1表示$P_1=\langle A@t_2,B@t_5,H@t_4,E\rangle$和$l_2$表示$P_2=\langle E@t_1,F\rangle$,则连接$l_1$和$l_2$就可以生成$\langle A@t_2,B@t_5,H@t_4,E@t_1,F\rangle$.

(a-)PAINT的构建基于一个站点的等级顺序o. o

衡量站点在公交网络中的重要程度,例如公交枢纽是重要的站点. $o(u)<o(v)$表示站点u比v重要,称u的等级比v高. o的选取不影响在PAINT上查询的正确性,但会影响索引预处理耗费和查询效率. 此外,o影响a-PAINT的准确率. 下文的讨论假设已知o,将在第6.3节中讨论o的计算.

定义6　支点、前缀路径、后缀路径:设P是从s到d的路径,支点是P中等级最高的站点,记为$s_\top$. P上从s到$s_\top$的路径称为前缀路径,记为P_+;从$s_\top$到d的路径称为后缀路径,记为P_-.

定义7　等级限制路径:起(终)点是支点的路径.

支点将P划分成P_+和P_-两段等级限制路径,索引只需要包含符合以下条件的等级限制路径即可.

每个站点$v\in S$具有标签集合$L_+(v)$和$L_-(v)$:

(1) $L_+(v)$中的每一条标签$\langle u,x,\pi_d,\pi_a,(s_+,r_+,\pi_+)\rangle$代表一条从$v$到$u$的等级限制路径$P_+$, $o(u)<o(v)$.

(2) $L_-(v)$中的每一条标签$\langle u,y,\pi_d,\pi_a,(s_-,r_-,\pi_-)\rangle$代表一条从$u$到$v$的等级限制路径$P_-$, $o(u)<o(v)$.

(3) 对于任意s-d查询q,存在$l_+\in L_+(s)$, $l_-\in L_-(d)$,使得连接l_+和l_-对应的路径构成q的其中一个解.

标签的各个分量的含义如图2所示. 表2列出了$L_+(A)$和$L_-(F)$所包含的标签

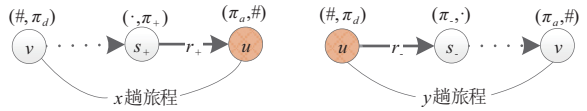

图2　标签所表示的路径.顶点表示换乘站点,元组(π_a,π_d)表示位于该站点的到达和出发时间.阴影站点表示支点,非换乘站点未显示

表2　集合$L_+(A)$和$L_-(F)$的标签

	$l_+=\langle u,x,\pi_d,\pi_a,(s_+,r_+,\pi_+)\rangle$	
$L_+(A)$	$l_1=\langle E,1,3,21,(A,r_1,3)\rangle$	$l_5=\langle B,1,3,6,(A,r_1,3)\rangle$
	$l_2=\langle E,1,4,23,(A,r_1,4)\rangle$	$l_6=\langle B,1,4,7,(A,r_1,4)\rangle$
	$l_3=\langle E,1,6,24,(A,r_1,6)\rangle$	$l_7=\langle B,1,6,9,(A,r_1,6)\rangle$
	$l_4=\langle E,3,4,20,(H,r_2,16)\rangle$	
	$l_-=\langle u,y,\pi_d,\pi_a,(s_-,r_-,\pi_-)\rangle$	
$L_-(F)$	$l_8=\langle E,1,21,24,(F,r_1,24)\rangle$	
	$l_9=\langle E,1,24,27,(F,r_1,27)\rangle$	

4.2　基本路径与a-PAINT

性质1　设P是查询q返回的其中一条路径,P_{sub}是P上两个换乘站点之间的子路径. 若P'_{sub}全面支配P_{sub},则在P上用P'_{sub}代替P_{sub}所得的路径仍然是q的一个解.

定义8　基本路径:如果P是等级限制且紧凑的.

若索引仅包含表示基本路径的标签,可否对任意查询返回正确解?讨论路径P是否在$s_\top$处换乘:

情况1　$s_\top$是换乘站点. P可以由$L_+(s)$和$L_-(d)$中的标签连接生成.

情况2　$s_\top$不是换乘站点,P_+和P_-不一定是紧凑的. 此时$L_+(s)$中必定存标签对应P'_+, $P'_+=P_+$或P'_+全面支配P_+. 同理$L_-(d)$中也存在标签对应路径P'_-, $P'_-=P_-$或P'_-全面支配P_-. P'_+连接P'_-可能生成P,也可能生成与P到达时间相同但多一趟换乘的路径.

标签集合只包含基本路径的索引记为a-PAINT. a-PAINT能返回一条次优路径,次优路径在时间方面不差于最优路径,仅比最优路径多一趟换乘.

4.3　线路-等级限制路径与PAINT

若要求返回所有最优解,PAINT需要存储比a-PAINT更多的路径. 继续第4.2节的讨论. 不妨设$L_-(d)$中丢失对应P_-的标签. 如果P'_-和P_-的始发路线相同,则可以用P'_-代替P_-生成与P具有相同时间和换乘数的路径. 不妨设P'_-乘坐$t_-(t_-\in\Gamma(r))$离开$s_\top$,在u处换乘并继续余下的旅程到达d. u是r上的站点,不一定在P上. 假设P_+乘坐旅程$t_+(t_+\in\Gamma(r))$途经$s_\top$,直到s_-处换乘. 由于P'_-全面支配P_-,有$\pi_{dept}(P_-)=\pi_{dept}(t_+,s_\top)\leqslant\pi_{dept}(P'_-)=\pi_{dept}(t_-,s_\top)$,于是$t_+\leqslant t_-$,$\pi_{arr}(t_+,u)\leqslant\pi_{arr}(t_-,u)$. 由此可以构造$P'$:沿着$P_+$行驶,在$s_+$处换乘$t_+$,途经$s_\top$直到在$u$处换乘,继续行驶$P'_-$余下的路径到达$d$. P和P'有相同的出发到达时间和换乘次数.

定义9　线路-等级限制路径:P满足(1)P是等级限制路径;(2)不存在另外一条从支点出发(或到达支点)的线路与P的相同的路径P',P'全面支配P.

PAINT中仅保留所有线路-等级限制路径,就可以正确回答最早到达查询和范围查询.

4.4　压缩标签

通过分析标签的相似性压缩标签存储,从而:(1)减少存储空间;(2)减少标签匹配量提高查询效率.

(a-)PAINT对站点v的标签集合$L_+(v)$压缩如下:(1) 具有相同支点的标签合并成一组,将标签组按照支点的等级从高到低排序. (2) 在同一组标签中,标签按旅程数升序排序. 用指针trips[x]指向首条旅程数为x的标签的位置. (3) 具有相同旅程数的标签按出发时间升序排列. 在a-PAINT中,这些标签自然地也是按到达时间升序排列. 在PAINT中,这些标签的到达时间不一定是按升序排列.

$L_+(A)$的标签压缩存储如图3所示. $L_-(v)$中相同支点和旅程数的标签按到达时间升序排序.

5　查询算法

查询过程实际上是匹配$L_+(s)$和$L_-(d)$内的标签. 以一个例子解释查询过程. 假设查询从A到F的最早到达路径,π=4. $L_+(A)$和$L_-(F)$的共同支点集合$U=\{E\}$. 按

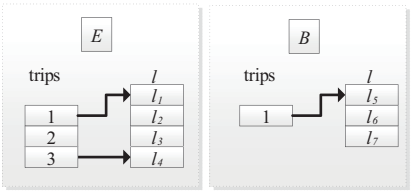

图3 表2中集合$L_+(A)$的标签压缩存储

x+y升序枚举(x,y)组合，x(或y)表示$L_+(A)$(或$L_-(F)$)中的标签的旅程数. 首先考虑(1,1). $L_+(A)$中支点为E、一趟旅程的标签是l_1,l_2和l_3,当中l_2是首条出发时间不早于π的标签. 在$L_-(F)$中,支点为E、一趟旅程且出发时间不早于23(l_2的到达时间)的标签是l_9. 考虑l_2的最后一个元组$(A,r_1,4)$和l_9的最后一个元组$(F,r_1,27)$. 存在$t_2\in\Gamma(r_1)$, t_2在时间4离开A, 27到达F,因此l_2连接l_9生成一条路径,旅程数为2-1=1. 另一条路径是l_4连接l_8,用了4趟旅程.

s-d-π_a-π_b范围查询与最早到达查询的过程类似,用元组(π_d^i,π_a^i)表示i趟旅程的最优路径的出发、到达时间. 当在$L_+(s)$中找到标签l_+与$L_-(d)$中的标签l_-相匹配时,l_+的出发时间不一定是最优的. 此时扫描l_+之后的标签l'_+, l'_+的出发时间应在$[\pi_a,\pi_b]$内. 将l'_+与l_-连接生成候选路径,更新(π_d^i,π_a^i).

6 预处理算法

6.1 计算最早到达路径

算法1 OneToAllEAP(s,π)求解在时刻π从站点s出发的到达时间-换乘次数帕累托最优路径集合. $\pi_i(v)$表示经过i趟旅程到达v的最早到达时间. 算法按照i的升序计算$\pi_i(v)$:在已有的i趟旅程的路径之后追加一段旅程片段,生成i+1趟旅程的路径.

S_i表示满足$\pi_i(v)<\pi_{i-1}(v)$的站点v的集合. 对于$u\in S_i$,算法扫描途经u的线路r. 采用二分查找在$\Gamma(r)$中找到首条满足换乘条件$\pi_{\text{dept}}(t,u)\geqslant\pi_i(u)$的旅程$t$,然后扫描$t$上位于$u$之后的站点$v$,用$\pi_{\text{arr}}(t,v)$更新$\pi_{i+1}(v)$. 第10行的$\tau_{i+1}(v)$记录标签的最后一个元组.

为了避免重复扫描线路,算法记录$\lambda_{i+1}(r,v)=t$,$t\in\Gamma(r)$,表示站点v在第i+1趟旅程乘坐t到达. 在第8行,若发现$\lambda_{i+1}(r,v)\leqslant t$,则可终止扫描$t$,因为$t$不可能对$v$后的站点产生更早的到达时间. 第2行的$t_\top$表示虚拟的最大旅程,即对于$\forall t\in T$,$t<t_\top$.

算法1 OneToAllEAP(s,π)

输入: 出发站点s,出发时间π

输出: 所有在π时间从s出发的到达时间-换乘次数帕累托最优路径

1. $\pi_0(s)=\pi$, $\pi_0(v)=\infty$, $v\in S-\{s\}$; $S_0=\{s\}$; $i=0$;
2. $\forall r\in R$, 对r上的各站点v, $\lambda_0(r,v)=t_\top$;
3. **WHILE** $S_i\neq\varnothing$ **DO**
4. **FOREACH** $u\in S_i$,路过u的线路r **DO**
5. $t\leftarrow\Gamma(r)$下满足: $\pi_{\text{dept}}(t,u)\geqslant\pi_i(u)$,且$t<\lambda_{i+1}(r,u)$的最小的旅程.
6. **IF** t不存在 **THEN** CONTINUE;
7. **FOREACH** t上u之后的站点v **DO**
8. **IF** $\lambda_{i+1}(r,v)\leqslant t$ **THEN** BREAK;
9. **IF** $\pi_{\text{arr}}(t,v)<\pi_{i+1}(v)$ **THEN** $\pi_{i+1}(v)=\pi_{\text{arr}}(t,v)$;$\lambda_{i+1}(r,v)=t$;$S_{i+1}$添加$v$;
10. **IF** $i=0$ **THEN** $\tau_1(v)=(v,r,\pi_1(v))$; **ELSE** $\tau_{i+1}(v)=\tau_i(u)$;
11. $i=i+1$;
12. 返回$\forall v\in S$,$\pi_i(v)$及对应的路径.

6.2 构建索引

a-PAINT构建索引算法BuildIndex如算法2所示. 算法按照站点s的等级顺序降序,依次调用OneToAllEAP及其逆向算法找到连接s的基本路径生成标签集合.

算法2 BuildIndex$(\mathcal{T},o)$

输入: $\mathcal{T}$和o. 假设站点已经按等级顺序排序,即$1\leqslant i<j\leqslant|S|$有$o(s_i)<o(s_j)$.

输出: 标签集合$L_+(v)$与$L_-(v)$, $v\in S$.

1. **FOR** $i=1,2,\cdots,|S|$ **DO**
2. $\Pi=s_i$的出发时间的集合,时间按降序排序.
3. **FOREACH** $\pi\in\Pi$ **DO**
4. 调用OneToAllEAP(s_i,π)得到路径集合$\mathcal{P}$. 改动: (1) 第1-2行不重置$\pi_-(\cdot)$和$\lambda_-(\cdot,\cdot)$的值;(2)第8行若v比s_i等级高,停止扫描线路.
5. **FOREACH** $P\in\mathcal{P}$ **DO**
6. 设P在时间π从s_i出发,在时间$\pi_k(u)$到达u,需要k趟旅程;
7. **IF** $L_+(s_i)$和$L_-(u)$生成路径P'全面支配P **THEN** CONTINUE;
8. 生成标签$\langle s_i,k,\pi,\pi_k(u),\tau_k(u)\rangle$加进$L_-(u)$;
9. 类似第2~8行, 调用反向OneToAllEAP,按到达时间升序的顺序扩展到达s_i的路径,生成标签添加进集合$L_+(u)$,u是路径的始发站.
10. 返回$\forall v\in S$, $L_+(v)$和$L_-(v)$.

PAINT在计算从支点s_i出发(到达)的路径时需要记录路径的首(末)线路. 算法在BuildIndex的第3行改用双重循环:首先枚举路过s_i的线路r,按降序访问$\Gamma(r)$中的旅程t,用t更新s_i的后续站点的到达时间,得到从s_i出发用一趟旅程可达的站点集合S_1. 接下来的计算过程与a-PAINT的处理一致.

定理1 设M是$\mathcal{T}$中的基本连接数,$\text{tn}_{\max}$是$\mathcal{T}$上紧凑路径的旅程数的最大值,$r_{\max}$是站点所在的线路数的最大值. a-PAINT预处理的最坏时间复杂度是$O(|S|\cdot\text{tn}_{\max}\cdot M)$, PAINT的是$O(|S|\cdot r_{\max}\cdot\text{tn}_{\max}\cdot M)$. 平均每个站点的标签集合大小是$l$,所有线路的站点数之和是$rs$,预处理空间复杂度是$O(l\cdot|S|+\text{tn}_{\max}\cdot(|S|+rs))$.

6.3 计算站点等级顺序o

本文采用启发式思想[3]计算o. 如果v是路径P上的换乘站点,称v覆盖P. v覆盖的路径越多,v越可能成

为路径的支点,则v的等级应越高. 对于每个$s\in S$,随机生成时间π,计算在π从s出发的最早到达路径,不考虑旅程数,得到一棵以s为根的树T_s. 如果从s到达u的路径在v处换乘,则在T_s上v是u的祖先. v在T_s中覆盖数等于以v为根的子树的结点数,总覆盖数为v在$|S|$棵树的覆盖数之和. 反复执行以下操作选择站点计算o:对于i=1,2,3,…,$|S|$,选择总覆盖数最大的s_i作为等级为i的点,从剩下的树中移除以s_i为根的子树,更新移除站点的总覆盖数.

7　实验

实验采用C++语言编写,在Windows 10上运行. 机器配置Intel(R) Core(TM) i7-8700K CPU, 64GB内存. 数据集使用地区一天的公交时间表①. 实验对比(a-)PAINT与PTL[5]的性能,并测试a-PAINT的准确率. PTL将时间表转化成MC图,将换乘限制查询问题转化成图的最短路计算问题,详见文献[5,15]. 各数据集的特征如表3所示.

表3　数据集特征(K=1024)

数据集	时间表		MC图	
	$\|S\|$	$\|T\|$	$\|V\|$	$\|E\|$
Austin	2.6K	11.7K	733.1K	1476.0K
Denver	9.5K	31.4K	2204.2K	4820.2K
Budapest	7.2K	114.7K	3859.8K	8144.7K
Sweden	49K	195.6K	8186.5K	10848.3K
Switzerland	30.4K	1085K	14666K	35398K

7.1　预处理时间

表4列出了预处理时间.(a-)PAINT的预处理时间为计算站点等级顺序和生成标签的时间之和. PTL计算顶点的等级顺序与生成标签的过程交替进行. 最后两个数据集转换成MC图的规模太大,运行PTL时内存不足.(a-)PAINT对$|S|$个站点计算等级顺序,$|S|$比$|V|$小约2个数量级. 在生成标签阶段,(a-)PAINT充分利用了路径支配的性质剪枝掉不可能生成解的路径,因此预处理速度较快.

表4　预处理时间(时:分:秒)

数据集	站点等级顺序	a-PAINT	PAINT	PTL
Austin	0:00:4	0:00:21	0:02:12	0:26:15
Denver	0:01:15	0:03:13	0:20:03	3:10:41
Budapest	0:01:12	0:06:04	0:40:01	6:44:59
Sweden	1:08:11	0:52:19	16:37:39	-
Switzerland	0:17:25	1:11:16	27:38:46	-

7.2　索引大小与运行时空间

表5列出索引大小和平均标签数.(a-)PAINT用32位表示时间,16位表示线路ID和站点ID.(a-)PAINT除了存储索引,只需要少量的额外空间存储辅助数据. PTL采用增量压缩方法[15],平均一条标签占约24位. 尽管PTL生成紧凑的索引,但运行时需要大量的内存存储辅助数据计算顶点的等级.

表5　索引大小(单位:MB). lbl./|S|指平均每个站点的$L_+(\cdot)$与$L_-(\cdot)$的标签数

数据集	a-PAINT		PAINT		PTL	
	lbl./|S|	索引	lbl./|S|	索引	lbl./|S|	索引
Austin	1299	110	5467	676	99133	680
Denver	1035	315	7161	3354	82911	2142
Budapest	2470	572	9709	3539	251285	4891
Sweden	761	962	12381	14333	-	-
Switzerland	1098	1153	11147	10743	-	-

7.3　查询时间

每个数据集包含十万个查询. 出发和目标站点均匀地随机选择,π在午夜到正午时间段内随机产生. 范围查询的出发时间段设成2小时,即$\pi_b=\pi_a+7200$. 由于范围查询速度较慢,只测试前一万个查询,查询时间如表6所示. 在最早到达查询中,(a-) PAINT需要扫描$L_+(s)$中所有出发时间不晚于π的标签;而范围查询只需要扫描出发时间在$[\pi_a,\pi_b]$范围内的标签,因此(a-)PAINT的范围查询速度更快.

表6　平均查询时间(单位:μs. 标记*号的是rRAPTOR[1]的查询时间)

数据集	最早到达查询			范围查询		
	a-	PAINT	PTL	a-	PAINT	PTL
Austin	14	97	11	17	100	80
Denver	18	198	16	17	129	62
Budapest	33	128	48	27	95	554
Sweden	7	251	28.4ms*	10	86	48.4ms*
Switzerland	32	335	40ms*	26	122	52.2ms*

7.4　a-PAINT准确率

本节测试a-PAINT的准确率. 对于一个最早到达查询q,设OP_q是最优路径的集合,AP_q是a-PAINT返回的集合. $|AP_q\cap OP_q|$表示AP_q中有多少条路径是最优的. 对于查询集合Q,准确率计算如下:

$$A(Q)=\frac{\sum_{q\in Q}|AP_q\cap OP_q|}{|OP_q|}$$

为了测试a-PAINT返回的路径与相同旅程数的最优路径在到达时间上的误差,令$P_k\in AP_q$是a-PAINT返回的一条有k趟旅程的路径,P_k^*表示k趟旅程的最优路径. P_k^*不一定存在,有可能从s到d的k趟旅程的路径的最早到达时间不优于k-1趟旅程的. 表7的第三列给出了在所有的次优路径中,存在比对路径的比例,用

①http://www.transitfeeds.com

$(\pi_{arr}(P_k)-\pi_{arr}(P_k^*))/(\pi_{arr}(P_k^*)-\pi)$计算到达时间延迟率,其中$\pi$是查询的出发时间.

表7 各数据集的查询结果的准确率

数据集	$A(Q)$	有比对的比例	有比对部分的延迟率	
			0~10%	10%~20%
Austin	0.78	0.42	0.25	0.2
Denver	0.77	0.42	0.26	0.261
Budapest	0.76	0.62	0.55	0.13
Sweden	0.86	0.49	0.34	0.29
Switzerland	0.95	0.43	0.5	0.18

8 总结

本文给出了一种运行时内存空间高效的公交网络路径规划索引方法,支持换乘次数约束的最早到达和出发时间范围限制的最优路径快速查询.本文对索引的存储结构、预处理方法和查询方法做了多方面的讨论和优化,用真实数据集测试算法的性能以及近似算法的准确率.(a-)PAINT的预处理耗费与时间表中的基本路径、线路-等级限制路径数目正相关.当时间表扩大到多天的时候,(a-)PAINT的预处理耗费也会成倍增长.考虑到公交时间表一般具有周期性,下一步工作计划将(a-)PAINT扩展到周期性时间表上.

参考文献

[1] Delling D, Pajor T, Werneck R F. Round-based public transit routing[J]. Transportation Science, 2015, 49(3): 591 – 604.

[2] Sun X, Wandelt S, Hansen M, et al. Multiple airport regions based on inter-airport temporal distances[J]. Transportation Research Part E: Logistics and Transportation Review, 2017, 101: 84 – 98.

[3] Wang S, Lin W, Yang Y, et al. Efficient route planning on public transportation networks: A labelling approach[A]. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data[C]. ACM, 2015. 967 – 982.

[4] Pyrga E, Schulz F, Wagner D, et al. Efficient models for timetable information in public transportation systems[J]. Journal of Experimental Algorithmics (JEA), 2008, 12: 1 – 39.

[5] Delling D, Dibbelt J, Pajor T, et al. Public transit labeling [A]. International Symposium on Experimental Algorithms [C]. Springer, Cham, 2015. 273 – 285.

[6] Bast H, Hertel M, Storandt S. Scalable transfer patterns[A]. Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX) [C]. USA: SIAM, 2016. 15 – 29.

[7] Witt S. Trip-based public transit routing. Algorithms-ESA 2015[M]. Berlin, Heidelberg: Springer, 2015. 1025 – 1036.

[8] Witt S. Trip-based public transit routing using condensed search trees[J]. arXiv preprint arXiv:1607.01299, 2016.

[9] Phan D M, Viennot L. Fast public transit routing with unrestricted walking through hub labeling[A].International Symposium on Experimental Algorithms[C]. Springer, Cham, 2019. 237 – 247.

[10] Delling D, Dibbelt J, Pajor T, et al. Faster transit routing by hyper partitioning[A]. 17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017) [C].Vienna, 2017.8.

[11] Weng D, Chen R, Zhang J, et al. Pareto-optimal transit route planning with multi-objective Monte-Carlo tree search[J]. IEEE Transactions on Intelligent Transportation Systems, 2020.1185 – 1195.

[12] 沈国江, 杜建波. 基于非等间隔的两线路公交换乘模型研究[J]. 浙江工业大学学报, 2017, (5):587 – 590.

[13] Bast H, Delling D, Goldberg A, et al. Route planning in transportation networks. Algorithm Engineering[M]. Berlin: Springer, 2016. 19 – 80.

[14] Abraham I, Delling D, Goldberg A V, et al. Hierarchical hub labelings for shortest paths[A].European Symposium on Algorithms[C]. Berlin, Heidelberg: Springer, 2012. 24 – 35.

[15] Delling D, Goldberg A V, Pajor T, et al. Robust distance queries on massive networks[A]. European Symposium on Algorithms[C]. Berlin, Heidelberg: Springer, 2014. 321 – 333.

作者简介

马 慧 女,1981年生,博士,副教授.研究方向为大规模图数据处理.

汤 庸(通信作者) 男,1964年生,博士,教授,学者网创始人(个人主页Scholat.com/ytang).主要研究领域为学术社交网络、教育大数据服务.E-mail:ytang@scnu.edu.cn