文章编号: 1000-5641(2019)05-0100-13

基于用户偏好的最优路径搜索

江 群[1], 戴戈南[1], 张 森[1], 葛又铭[1], 刘玉葆[1,2]

(1. 中山大学 数据科学与计算机学院, 广州 510006;
2. 中山大学 广东省大数据分析与处理重点实验室, 广州 510006)

摘要: 本文研究基于用户偏好的最优路径搜索, 在预算约束下寻找一条满足用户偏好即关键字和权重偏好的最优路径. 此研究问题是NP-hard. 为了高效地解决这类查询问题, 本文提出新的索引建立方法, 在查询阶段利用索引结构过滤出候选节点集. 另外, 提出基于A*的路径搜索算法来做路径查询, 并利用几个有效的剪枝策略加快算法的执行速度. 在两个真实的签到数据集上的实验结果证明了本文提出方法的有效性. 当预算时间设置为4∼7 h时, 与已有最好的PACER算法相比, 本文的路径搜索算法消耗的查询时间更短.

关键词: 路径搜索; 用户偏好; A*算法

中图分类号: TP311.13 **文献标志码**: A **DOI:** 10.3969/j.issn.1000-5641.2019.05.008

Optimal route search based on user preferences

JIANG Qun[1], DAI Ge-nan[1], ZHANG Sen[1],
GE You-ming[1], LIU Yu-bao[1,2]

(1. *School of Data and Computer Science, Sun Yat-Sen University, Guangzhou* 510006, *China*;
2. *Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-Sen University, Guangzhou* 510006, *China*)

Abstract: This paper studies the methodology for optimal route search based on user preferences, such as keyword and weight preferences, under a constraint. The research problem is NP-hard. To solve the query efficiently, we propose two new index building methods and select candidate nodes for retrieving the established indices. This paper subsequently proposes an A* based route search algorithm to identify the optimal route and use several effective pruning strategies to speed up execution. Experimental results on two real-world check-in datasets demonstrates the effectiveness of the proposed method. When the budget ranges from 4 hours to 7 hours, our algorithm performs better than the state-of-the-art PACER algorithm.

Keywords: route search; users' preferences; A* algorithm

收稿日期: 2019-07-29

基金项目: 国家自然科学基金(U1501252, 61572537)

第一作者: 江 群, 女, 硕士研究生, 研究方向为数据库与数据挖掘. E-mail: 2739670944@qq.com.

通信作者: 刘玉葆, 男, 教授, 博士生导师, 研究方向为数据库与数据挖掘.
E-mail: liuyubao@mail.sysu.edu.cn.

0 引　　言

路径搜索与人们的生活息息相关, 被广泛应用于智能导航、路径推荐、AR游戏等领域. 一种人们熟知并常见的路径搜索场景是, 用户希望找到一条从当前所在地出发前往目的地的距离最短或花销最少的路径, 即最短路径. 最近几年流行起来的基于位置的社交软件提供签到功能, 比如Foursquare和Gowalla, 用户可以在地图上标注自己访问过的兴趣点并留下对它们的评价. 这些兴起的基于位置的服务(Location Based Services)[1]积累了很多用户签到数据, 通过对这些蕴含地点信息和路径信息的数据进行挖掘和分析, POI地图初现轮廓. POI地图最主要的特征是每个兴趣点都带有标签, 表明该兴趣点的所属类别. 当兴趣点带有多个功能属性时, 就会标上多个标签. 如一个大型的购物商场, 人们既能在此购物逛街, 又能进行看电影、溜冰的娱乐活动, 那么这个购物商场极有可能被标记为商场、电影院、溜冰场.

近十几年来, 专注于路径搜索问题的研究人员开始把关键字作为查询条件. 一个新的研究问题是, 用户希望被推荐的路径不仅能够满足他的代价预算和空间约束, 还能最优地实现他对兴趣点特征的多样性要求. 其实例化描述是, 一个第一次来到广州的游客, 计划上午9:00从入住的斯特威酒店出发, 前往广州白云国际机场搭乘下午17:00点的飞机. 中间8个小时的时间, 他想去参观广州有名的博物馆、逛逛商业街, 或者游览风景区. 在他心中, 相比于逛商业街, 他更愿意把时间花在博物馆和风景区上. 根据以上描述可知, 用户偏好体现在用户只想访问博物馆、商业街、风景区这类兴趣点, 且对博物馆和风景区的游览兴致大于商业街. 所以, 一条经过一个博物馆、一条商业街和一个风景区的路线对用户的吸引力比经过一个博物馆和两条商业街的路线要大. 对于该场景, 文献[2]实现了一个路径搜索系统, 支持带用户偏好的路径查询.

许多研究工作致力于解决覆盖多关键字的最优路径查询. 2008年, 文献[3]提出了Optimal Sequenced Route (OSR)问题, 在TPQ问题[4]基础上加上关键字访问次序的约束. 研究次序约束问题的还有[5-7]. 论文[8]针对KORS[7]算法基于邻边拓展的路径构建策略在大规模图以及多查询关键字下复杂度过高、可扩展性不足的缺陷, 提出基于关键词序列的路径构建方案. 2018年, 文献[9]研究top-k最优有序路径问题, 即查询从起点到终点按指定次序经过特定类型节点且代价最小的top-k路径. 然而, 这些问题都没有考虑路径的约束条件, 如时间、距离等. 支持多约束条件的路径查询工作有[10-12]. 其中, KOR问题[12]致力于查找约束条件下带查询关键字的近似最优解. 文献[13]提出了查询带差异的top-k最短路径, 使得查出的所有路径之间相似度最低且路径代价之和最小. 为了让求解的路径满足关键字多样性要求, 文献[14]指出需要使用子模函数对这个问题建模. 虽然已有工作[15]研究带路径预算约束的子模函数优化问题, 但是它采用的递归贪婪算法只能找到近似解, 而且不适用于城市范围的路径规划. 为了高效地找到一条最优路径, 文献[16]提出基于A*变体的路径搜索算法, 但是该研究的预算只考虑了路径的出行代价. 根据实际应用场景, 在路径搜索当中同时考虑出行代价和兴趣点的停留代价更加合理. 同样考虑用户偏好的有文献[17], 它从用户的轨迹数据里挖掘出用户的开车偏好, 如时间最短, 汽油最省或介于两者之间, 产生一个偏好向量, 据此向用户推荐个性化路线.

为了研究用户的个人倾向对路径推荐的影响, 本文把用户的类别偏好和权重偏好作为查询条件加入路径搜索问题的定义当中. 另外, 大多数路径搜索问题的预算只考虑在兴趣点

之间的路径代价, 而不考虑访问兴趣点产生的停留代价. 为了让研究问题更加合理和通用, 本文考虑将路径的出行代价和兴趣点的停留代价记入预算当中.

概括起来, 本文的主要贡献有:

- 提出基于用户偏好的最优路径搜索问题, 在预算约束下寻找满足用户偏好的最优路径.
- 提出最小代价矩阵索引和关键字反向索引, 利用索引结构检索出候选节点集.
- 基于 A* 框架设计了一个启发式路径搜索算法, 结合有效的剪枝策略找到问题的最优解.
- 在来源于真实世界的两个签到数据集上进行实验验证, 实验结果证明了本文提出方法的有效性.

本文总共划分为 5 节, 第 1 节给出问题定义; 第 2 节提出两个索引方法; 第 3 节介绍本文提出的基于 A* 的路径搜索算法和剪枝策略; 第 4 节给出实验结果与分析; 第 5 节对全文进行总结.

1 问题定义

1.1 问题描述

给出一个 POI 地图示例, 见图 1. 地图上的每个节点 v_i 代表一个兴趣点. 根据兴趣点拥有的功能属性, 节点 v_i 会标记上一个或多个关键字. 与关键字有关的评分, 在图中标记在关键字之后. 如节点 v_6 标记的关键字有 Museum 和 Restaurant, 对应的评分分别为 0.4 和 0.6. 在节点 v_i 产生的停留代价标记在节点旁边. 另外, 两两节点之间的出行代价标记在连接边上. 如访问节点 v_6 产生的停留代价为 18, 从 v_6 到 v_4 花费的出行代价为 3.

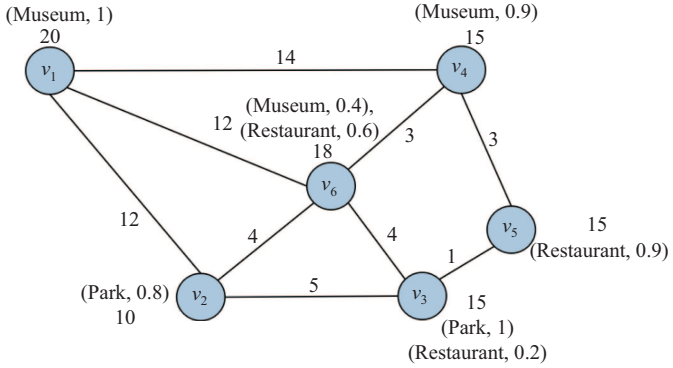


图 1 POI 地图示例

Fig. 1 A POI map

1.2 问题定义

定义 1.1 POI 地图 POI 地图 $G = (V, E)$, 其中点集 V 是由兴趣点 $\{v_1, v_2, \cdots, v_n\}$ 组成, $1 \leqslant n \leqslant |V|$, 每个兴趣点 v_i 表示一个地理位置并被标记一个或一个以上的关键字, 其关键字质量用评分来衡量, 记作 $SC_{k_q}(v_i)$. 访问该兴趣点产生的停留代价记作 $s(v_i)$, 这个代价可以是时间、金钱等. 边集 E 是由兴趣点之间存在的无向边组成, 每一条边 $(v_i, v_j) \in E$ 连接两个兴趣点 v_i 和 v_j, 两点之间的出行代价记作 $T(v_i, v_j)$.

定义 1.2 路径 R 路线 $R = (v_1, \cdots, v_m)$ 是从起点 v_1 到达终点 v_m, 经过一系列非重复兴趣点连成的路径, 路径的起点和终点可以相同. 从节点 v_i 到下个访问点 v_j 之间的最小出行代

价记作$T_{sm}(v_i, v_j)$. 路径R的路径代价由每一段路径的最小代价和兴趣点的停留代价组成, 即

$$cost(R) = \sum_{i=1}^{m-1} s(v_i) + \sum_{i=1}^{m-1} T_{sm}(v_i, v_{i+1}). \tag{1}$$

定义 1.3 (路径收益) 给定查询关键字集合$K = \{k_1, k_2, \cdots, k_m\}$和关键字偏好$\lambda_{k_q}$ (大于0), $k_q \in K$. 路径收益用来衡量路径R对用户偏好的满足程度, 收益函数定义为

$$Gain(R) = \sum_{k_q \in K} \lambda_{k_q} cov_{k_q}(R). \tag{2}$$

其中, $cov_{k_q}(R)$表示路径R上关键字标签为k_q的所有兴趣点对关键字k_q的联合覆盖情况, 定义为

$$cov_{k_q}(R) = 1 - \Pi_{v_i \in R}[1 - SC_{kq}(v_i)]. \tag{3}$$

定义 1.4 (最优路径搜索问题) 给定POI地图$G = (V, E)$, 查询$Q = (v_s, v_t, K, \Upsilon, \Delta)$, 其中$v_s$是起点, v_t是终点, K是一组查询关键字集, Υ是相应的关键字权重集合, Δ用来指定预算约束. 问题的查询目标是找到一条从v_s到v_t的路径R, 使得

$$\begin{aligned} &R = argmax_R\, Gain(R), \\ &\text{满足 } cost(R) \leqslant \Delta. \end{aligned} \tag{4}$$

本文采用的收益函数公式(2)具有子模性质, 满足收益递减特性, 即对$\forall R_1 \subseteq R_2 \subseteq R$, $Gain(R_1 \cup \{v_i\}) - Gain(R_1) \geqslant Gain(R_2 \cup \{v_i\}) - Gain(R_2)$成立. 采用其他的子模函数也可以对路径的多样性特征建模. 很多实际问题都利用子模性质来解决, 如文档推荐[18]、社交网络传播[19]等.

下面用表1总结与本文相关的符号说明.

表 1　符号及其含义

Tab. 1　Nomenclature

符号	含义
$s(v_i)$	停留兴趣点v_i产生的代价
$SC_{k_q}(v_i)$	关键字k_q在兴趣点v_i处的评分
$T(v_i, v_j)$	(v_i, v_j)边代价
$T_{sm}(v_i, v_j)$	(v_i, v_j)边最小代价
$cost(R)$	路径R的代价
$Gain(R)$	路径R的收益
$Q = (v_s, v_t, K, \Upsilon, \Delta)$	v_s为起点, v_t为终点, K为查询关键字集, Υ为关键字权重集, Δ为预算约束
V_Q	查询Q的候选节点集
R_i^k	从v_s到v_i的第k条路径
L_i^k	从v_s到v_i的第k条路径的标签

2 索引方法

2.1 索引结构

2.1.1 最小代价矩阵索引

在路径搜索过程中, 经常需要计算两个节点之间的最小代价. 然而, 从POI地图里各兴趣点之间的邻接边构造出来的带权邻接矩阵, 只能反映相邻节点之间的路径代价, 不存在邻接边的两个节点在矩阵上的表示为无穷大, 即不可达. 对POI地图G抽象出的邻接矩阵执行Floyd算法, 对每一对顶点$v_i \neq v_j$, 求出v_i和v_j之间的最小代价. 算法执行结束后即可得到最小代价矩阵索引, 如图2所示.

2.1.2 关键字反向索引

根据查询Q指定的关键字集K, 需要检索出携带这些关键字的兴趣点. 然后, 在这些候选节点集里进行路径搜索. 在此, 提出关键字反向索引.

关键字反向索引把每一个关键字k_p映射到一个列表, 这个列表记录着所有标记了k_p的兴趣点和兴趣点关于此关键字的评分, 所以列表的标签项为$(v_i, SC_{k_p}(v_i))$, 记录项之间按兴趣点评分由大到小排列. 以上面图1所示的POI地图为例, 为图中的三个关键字K ={Park, Museum, Restaurant}建立关键字反向索引, 如图3所示. 携带Museum标签的关键字有v_1, v_4, v_6, 它们之间的排列顺序是评分高的在前, 评分低的在后.

$$\begin{pmatrix} 0 & 12 & 16 & 14 & 17 & 12 \\ 12 & 0 & 5 & 7 & 6 & 4 \\ 16 & 5 & 0 & 4 & 1 & 4 \\ 14 & 7 & 4 & 0 & 3 & 3 \\ 17 & 6 & 1 & 3 & 0 & 5 \\ 12 & 4 & 4 & 3 & 5 & 0 \end{pmatrix}$$

图2 POI地图的最小代价矩阵索引

Fig. 2 Minimum cost matrix index of the POI map

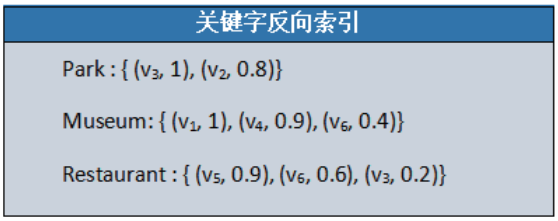


图3 POI地图的关键字反向索引

Fig. 3 Keyword inverted index of the POI map

2.2 检索过程

利用上一小节建立的两个索引, 根据查询$Q = (v_s, v_t, K, \Upsilon, \Delta)$里的预算代价$\Delta$和查询关键字集$K$, 可以检索出POI候选节点集$V_Q$. 下面介绍整个检索过程.

第一步, 检索候选节点. 使用关键字反向索引, 依次检索集合K中的关键字对应的列表, 将列表中的节点加入候选节点集V_Q.

第二步, 检查候选节点. 利用最小代价矩阵索引, 以x为起点, y为终点, 检查在单点访问的情况下, 候选节点集V_Q中的兴趣点v_i是否满足预算约束. 如果$T(x, v_i) + s(v_i) + T(v_i, y) \leqslant \Delta$, 那么兴趣点$v_i$可以作为候选节点等待扩展. 否则, 该兴趣点距离起点和终点太远, 无法被访问,

从候选节点集V_Q中删除.

经过这两步操作, 得到最终的POI候选点集V_Q. 很明显, $|V_Q| << |V|$. 下面通过一个具体的例子来说明如何进行这个检索过程.

例 2.1 对图1所示的POI地图给出查询$Q = (v_2, v_6,\{\text{Park,Museum}\},\{0.5,0.5\},30)$. 为图1建立的最小代价矩阵索引、关键字反向索引如图2、图3所示. 首先检索"Park"的反向索引, 从列表中得到$\{v_2, v_3\}$; 检索"Museum"的反向索引, 得到$\{v_1, v_4, v_6\}$. 去掉查询指定的起点v_2和终点v_6之后, 第一步得到的候选节点集$V_Q = \{v_1, v_3, v_4\}$. 接着进行第二步, 检查集合里的每个候选节点的单点访问代价是否超出预算. 对于节点v_1, $T(v_2, v_1) + s(v_1) + T(v_1, v_6) = 12 + 20 + 12 = 44 > 30$, 从起点经过它直接回到终点的路径代价超出预算约束, 需要从候选点集里删除. 最后得到$V_Q = \{v_3, v_4\}$.

3　路径搜索算法

3.1　算法设计

解决最优路径搜索问题的一个可行方法是做暴力搜索. 暴力搜索是一种穷举方法, 枚举出所有从源点出发到达终点的小于预算的可行路径, 然后在可行路径中找出一条收益值最高的路径作为问题的最优解. 虽然暴力搜索方法能保证找到一条最优路径, 但是它的计算量很大, 时间开销和空间开销让计算机难以负荷. 为了避免枚举所有的候选路径, 本文提出一种A* 算法的变体, 利用启发式信息和有效的剪枝策略减少路径的搜索空间.

从源点到终点的宽度优先搜索过程可以表示成一棵搜索树. 当没有任何启发式信息的时候, 搜索只能按照层次遍历或深度遍历的方式进行. 已知A* 算法采用的知识启发式代价函数可以用来决定搜索进行的方向, 当路径扩展到节点n的时候, 定义它的启发式收益估值函数为

$$f^n(R_{s\to t}) = g^n(R_{s\to n}) + h^n(R_{n\to t}|R_{s\to n}). \tag{5}$$

这里的$g^n(R_{s\to n})$表示路径$R_{s\to n}$的实际收益, 可用收益函数$Gain(R_{s\to n})$求解, $h^n(R_{n\to t}|R_{s\to n})$用来估计从节点$n$出发的后继路径$R_{n\to t}$关于前序路径$R_{s\to n}$的边际收益. 与传统的A* 算法不同的是, 启发函数$h^n(\cdot)$依赖于$g^n(\cdot)$, 这就给估值函数$f^n(R_{s\to t})$的设计带来了很大的困难.

路径标签 从起点v_s到节点v_i存在多条路径. 为了区分经过v_i节点的多条路径, 定义路径标签$L_i^k = \langle R_i^k, cost(R_i^k), Gain(R_i^k), f^i(R_{s\to t})\rangle$来记录路径信息. 其中$R_i^k$表示从节点$v_s$到节点$v_i$的第k条路径, $cost(R_i^k)$、$Gain(R_i^k)$分别表示这条路径的代价和收益, $f^i(R_{s\to t})$则表示经由节点v_i到达终点的整条路径的收益估值. 使用最大优先队列Q来存储这些路径标签, 队列元素按$f^i(R_{s\to t})$从大到小排列.

本文提出一种基于A* 的路径搜索算法, 其伪代码如算法1所示. 算法的处理逻辑是, 首先, 初始化一个最大优先队列Q用来存储当前等待扩展的路径标签, 最优路径R置空, $Gain_{\max}$记录当前最优路径的收益, $BS_{\min}$记录它的路径代价. 接着, 为源点v_s建立第一个路径标签, 并将它插入队列Q (第1—4行). 当队列Q不为空的时候, 每次从队列中移除一个路径标签, 检查该标签的路径收益估值是否小于当前的最大收益. 如果是, 则退出循环, 返回R. 否则从路径标签中取出路径, 并使用新的候选节点集CandiSet存储V_Q中没有被R_i^k访问过的节点 (第7—11行). 然后, 依次把CandiSet中的每一个元素加入到路径R_i^k, 创建一条新的开放路径R_j^l. 如果这条开放路径加上终点形成的闭合路径代价大于Δ, 就回到第12行 (第12—18行). 否则计算出路径收益$Gain(R_j^l)$, 做如下处理: ①若路径收益大于当前的$Gain_{\max}$, 则更新R、$Gain_{\max}$和$BS_{\min}$. ②若路径收益等于当前的$Gain_{\max}$且路径代价小于$BS_{\min}$, 那么取代价更小的那条作为最优路

径 (第 20—30 行). 接着估算新开放路径的收益上界, 然后为收益估值大于 $Gain_{\max}$ 的路径创建路径标签 L_j^l 并加入队列 Q (第 32—36 行). 最后, 返回最优路径 R.

算法 1 基于 A* 的路径搜索算法

输入: 查询 $Q=(v_s,v_t,K,\Upsilon,\Delta)$, 图 $G=(V_Q,E)$, 最小代价矩阵索引 T_{sm}, 停留时间列表 s

输出: 最优路径 R

```
1: 初始化一个最大优先队列 Q;
2: R = ∅, Gain_max = −∞, BS_min = +∞;
3: 创建路径标签 L_s^0 = ⟨(v_s), 0, 0, 0⟩;
4: Q.enqueue (L_s^0);
5: While Q 不为空 do
6: {
7:     L_i^k =Q.dequeue;
8:     If L_i^k.f^i(R_{s→t}) ⩽ Gain_max then
9:         Break;
10:    从路径标签 L_i^k 中取出路径 R_i^k;
11:    新的候选节点集 CandiSet =V_Q \ V_{R_i^k};
12:    For v_j in CandiSet do
13:    {
14:        创建一条路径 R_j^l = R_i^k ∪ {v_j};
15:        cost(R_j^l) = L_i^k.cost(R_i^k) + T_sm(v_i, v_j) + s(v_j);
16:        finalCost = cost(R_j^l) + T_sm(v_j, v_t);
17:        If finalCost > Δ then
18:            Continue;
19:        计算路径收益 Gain(R_j^l);
20:        If Gain(R_j^l) > Gain_max then
21:        {
22:            R = R_j^l ∪ {v_t};
23:            Gain_max = Gain(R_j^l);
24:            BS_min = finalCost;
25:        }
26:        If Gain(R_j^l) == Gain_max and finalCost < BS_min then
27:        {
28:            R = R_j^l ∪ {v_t};
29:            BS_min = finalCost;
30:        }
31:        估计收益上界 f^j(R_{s→t});
32:        If f^j(R_{s→t}) ⩾ Gain_max then
33:        {
34:            创建路径标签 L_l^j = ⟨R_l^j, cost(R_l^j), Gain(R_l^j), f^j(R_{s→t}))⟩;
35:            Q.enqueue(L_l^j);
36:        }
37:    }
38: }
39: return R;
```

3.2　剪枝策略

3.2.1　基于代价的剪枝

设查询Q的起点为v_s, 终点为v_t, 把终点$end(R) \neq v_t$的路径称为开放路径; 把$end(R) = v_t$的路径称为闭合路径. 根据问题定义, 初始化的时候开放路径只包含节点v_s. 随着路径的扩展, 每向下搜索一层, 开放路径就新增一个节点. 开放路径可以扩展的深度由它的路径代价决定, 路径代价必须小于查询预算.

定理 3.1　定义1.4的一条可行路径包含路径R_j^l, 当且仅当它的闭合路径$R_j^l \cup \{v_t\}$满足$cost(R_j^l) + T_{sm}(v_j, v_t) \leqslant \Delta$.

证　明　(1) 首先证明定理的充分性. 由已知条件, 问题定义的一条可行路径包含路径R_j^l, 设可行路径为$R_{s\to j\to t}^l$. 根据路径代价的定义, 有$cost(R_{s\to j\to t}^l) = cost(R_j^l) + cost(R_{j\to t}) \leqslant \Delta$. 因为$T_{sm}(v_j, v_t) \leqslant cost(R_{j\to t})$, 等号当且仅当$R_{j\to t}$没有经过任何中间节点而是直接到达$v_t$时成立, 所以有$cost(R_j^l) + T_{sm}(v_j, v_t) \leqslant \Delta$. 定理的充分性成立.

(2) 其次证明定理的必要性. 根据结论, 闭合路径$R_j^l \cup \{v_t\}$满足$cost(R_j^l) + T_{sm}(v_j, v_t) \leqslant \Delta$, 所以路径$R_j^l \cup \{v_t\}$是问题定义的一个可行解, 或者说是一条可行路径. 由此可知, 定义1.4的一条可行路径包含路径R_j^l成立.

综上, 定理3.1得证.

算法1的伪代码描述用到了基于代价的剪枝: 在为每一条开放路径扩展新的节点时, 如果要判断形成的新路径在未来能否扩展成一条可行路径, 可以通过检查它的闭合路径是否满足$cost(R_j^l) + T_{sm}(v_j, v_t) \leqslant \Delta$. 如果满足条件, 则为这条新路径创建路径标签, 把它加入优先队列, 等待后面被扩展. 见代码第15至第18行.

3.2.2　基于上界的剪枝

令路径$R \to \widehat{R}$为闭合路径$R \to R'$中收益最大的那条. 如果路径$R \to \widehat{R}$的收益上界小于当前的最大收益, 那么$R \to \widehat{R}$不可能是最优路径, 所有从$end(R)$出发扩展的路径都可以被裁剪掉. 这就是基于收益的剪枝的基本思想.

当路径R从起点v_s到达了中间节点v_n, 后续路径的预算剩下$\Delta' = \Delta - cost(R_{s\to n})$. 用集合$L_n$记录以$v_s$为起点, v_t为终点, 代价预算为Δ'时, $R_{n\to t}$可以经过的节点:

$$L_n = \{v_i | T_{sm}(v_n, v_i) + s(v_i) + T_{sm}(v_i, v_t) + s(v_t) \leqslant \Delta'\}. \tag{6}$$

这里的节点v_i是之前没有访问过的节点, 并且不包括v_t.

由于$\Delta Gain(R'|R) = Gain(R \to R') - Gain(R)$只与路径$R'$经过的节点相关, 而不依赖于节点的访问次序, 把$R'$的路径代价近似为路径节点的集合代价. 节点代价定义为

$$c(v_j) = s(v_j) + \frac{\min\{T_{sm}(v_i, v_j) + T_{sm}(v_j, v_k)\}}{2}, \quad v_j \in L_n. \tag{7}$$

其中, $< v_i, v_j >$是v_j的一条入边, $< v_j, v_k >$是v_j的一条出边. 为了更加准确地估计集合代价, 终点v_t的代价定义为$c(v_t) = s(v_t) + \frac{\min\{T_{sm}(v_i, v_t)\}}{2}$, $s(v_t)$可以计入也可以忽略不计; R的终端节点v_n的代价定义为$c(v_n) = \frac{\min\{T_{sm}(v_n, v_k)\}}{2}$.

这样一来, 路径R'产生的边际收益估计就转化为求解下面的最优化问题:

$$\max \Delta Gain(R'|R) \quad \text{满足} \sum_{v_i \in V_{R'}} c(v_i) \leqslant \Delta' - c(v_n). \tag{8}$$

由于所有节点的代价$c(v_i)$都不超过它的真实值, 上面公式求出的最大值$UP \geqslant \Delta Gain(R'|R)$对于所有的路径$R'$成立. 公式(8)表示的是一个服从背包约束的子模最大化问题, 它也是NP-hard, 采用下面的定理求解.

定理 3.2 对每一个兴趣点$i \in L_n \cup \{v_t\}$, 令$B = \Delta' - c(v_n)$, $\sigma_i = \Delta Gain(\{i\}|R)$, $r_i = \sigma_i/c(i)$, $i_1, \cdots, i_m$是这些兴趣点的序, 按r_i降序排列. 设l满足$C = \sum_{j=1}^{l-1} c(i_j) \leqslant B$且$\sum_{j=1}^{l} c(i_j) > B$. 令$\lambda = (B-C)/c(i_l)$, 那么

$$UP = \sum_{j=1}^{l-1} \sigma_{i_j} + \lambda \sigma_{i_l} \geqslant \Delta Gain(R'|R). \tag{9}$$

证 明 文献[20]指出, 求受约束的子模最大化问题的解$\widehat{A}$的上界可以转化为求$\widehat{A}$与最优解的偏离. 把文献[20]提到的方法应用于公式(8), 令$\widehat{A} = 0$, 就推导出公式(9).

3.3 复杂度分析

以扩展的路径数作为算法的时间复杂度. 设候选节点集V_Q的节点个数为n, 最优路径的长度为p, 搜索树的每一层最多有占比γ的节点被扩展. 那么, 每一层l的路径有

$$f(l) = \begin{cases} n, & l = 1, \\ \gamma \cdot f(l-1) \cdot (n-l+1), & l > 1. \end{cases}$$

第p层的路径数目为$f(p) = \gamma^{p-1} \cdot n!/(n-p)!$. 当找到最优路径时, 已经扩展的路径最多有$\sum_{i=1}^{p} f(i) = \sum_{i=1}^{p} \gamma^{i-1} \cdot n!/(n-i)!$. 因此, 算法的时间复杂度为$O(\gamma^{p-1} \cdot n!/(n-p)!)$.

4 实验结果与分析

4.1 实验数据与环境

本文的实验数据是两个真实的签到数据集, 其中Singapore (SG)来源于Foursquare软件在新加坡收集的数据, 数据产生的时间是从2010年8月到2011年7月[21]; Austin (AS)来源于Gowalla软件在奥斯汀收集的数据, 数据产生的时间是从2009年11月到2010年10月[22].

跟相关研究工作[12,14,16]的处理相同, 我们为一天之内被某用户连续访问的两个兴趣点建立一条无向边. 那些孤立的没有边直接相连的兴趣点则从兴趣点集中去掉. 这些信息可以通过对签到数据进行分析得来. 对初始数据集进行[14]所述处理之后, 得到表2所示的描述性统计信息.

表 2 数据集的描述性统计信息

Tab. 2 Descriptive statistics of datasets

数据集	兴趣点个数	边数	平均路程时间	关键字个数
SG	1 625	24 969	16.24 min	202
AS	2 609	34 340	11.12 min	252

实验环境为Ubuntu 18.04 LTS操作系统, 电脑配置为Intel E5-2609 v4 1.70 GHz, 15.40 GB内存, 编程语言为C++.

4.2 实验设置

对比算法 本文以暴力算法BF作为基准方法, 与本文提出的基于A*的路径搜索算法进行实验比较. 为了比较的公平, 将本文在第2节提出的索引方法应用到暴力算法中. 由于Hongwei Liang和Ke Wang在SIGIR'18会议上提出的索引结构和PACER算法[14]也适用于求解本研究问题, 在实验部分加上与当前先进的PACER算法的比较. 把它的目标函数改为采用本文的覆盖函数, 并且设置k=1, 求top-1的最优路径. 因为[14]与本文算法求解的都是最优路径且实验数据集相同, 本实验结果可以说明算法的有效性.

评价指标 以运行时间、内存大小作为实验度量标准. 每一组查询结束之后, 记录本组 20 个查询的平均值. 当某个查询运行 1 h 后还没结束, 或者耗尽了计算机内存资源, 会被终止运行, 然后设置为查询失败, 不参与本次查询组的性能评估.

4.3　结果与分析

4.3.1　预算代价对算法性能的影响

运行时间 SG 数据集的实验结果如图 4(a) 所示. 随着预算时间的增加, 三个算法的性能开始下降, 但是本文的 A* 算法始终比 BF 快 2 到 3 倍. 算法性能下降一方面是因为预算值的增大使得索引检索阶段留下的候选节点增多, 另一方面是路径扩展的深度会增长. 在 4~7 h 内, A* 算法的查询速度要快于 PACER. 当 $\Delta = 8$ h 的时候, A* 和 PACER 都能成功执行所有查询且两个算法的查询效率相近, 而 BF 有 1 个查询失败. 相对于 SG 而言, AS 是一个更大的数据集, 其实验结果如图 4(b) 所示. 在 4~7 h 内, A* 算法的性能总是稍稍优于 PACER 算法. 由于在 $\Delta = 9$ h 的时候, BF、A* 和 PACER 只能成功执行 12、15、17 个查询, 成功率不高, 因此认为在 AS 数据集上不能做 $\Delta = 9$ h 的查询.

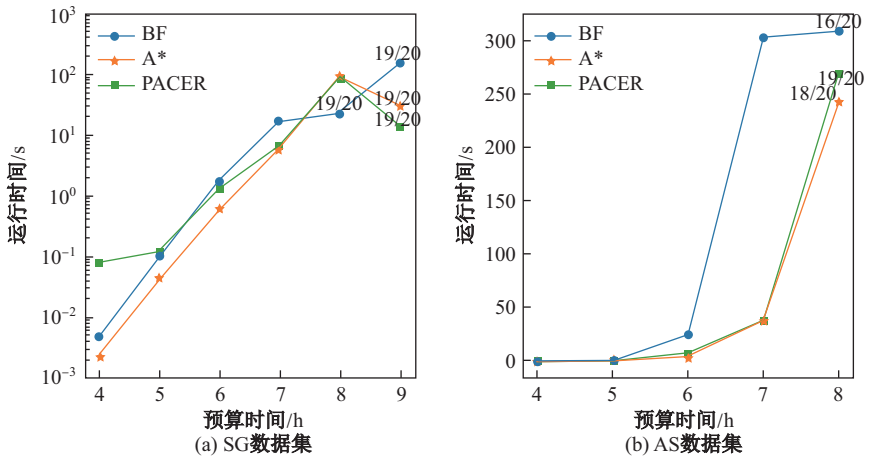


图 4　预算代价对运行时间的影响

Fig. 4　Impact of budget on runtime

总的来说, A* 算法的优势在于当预算时间取值为 [4, 5, 6, 7] 时, 它能成功执行所有查询并且它的查询性能是三个算法里最高的. 随着预算代价的增加, A* 算法和 BF 算法的性能差距越来越大.

搜索空间 SG 数据集和 AS 数据集的搜索空间实验结果如图 5(a)、5(b) 所示. 各个算法的路径搜索空间随着时间的增加而增长, 其中 BF 算法的增长速度最快. 在 SG 数据集上观察 4~8 h 的平均搜索空间可以发现, A* 算法比 BF 算法差不多少两个数量级, 并且比 PACER 算法的搜索空间要少. 在 AS 数据集上也有同样的表现.

4.3.2　关键字数量对算法性能的影响

SG 数据集和 AS 数据集的运行时间实验结果如图 6(a)、6(b) 所示, 实验的预算时间取定为 6 h. 从图中可以看到, 随着关键字个数的增加, 三个算法的运行时间呈增长趋势. A* 算法在 SG 数据集上大概 1 s 就能完成一个查询. 在 AS 数据集上, A* 的查询性能也很高, 运行时间平

均不到 2 s.

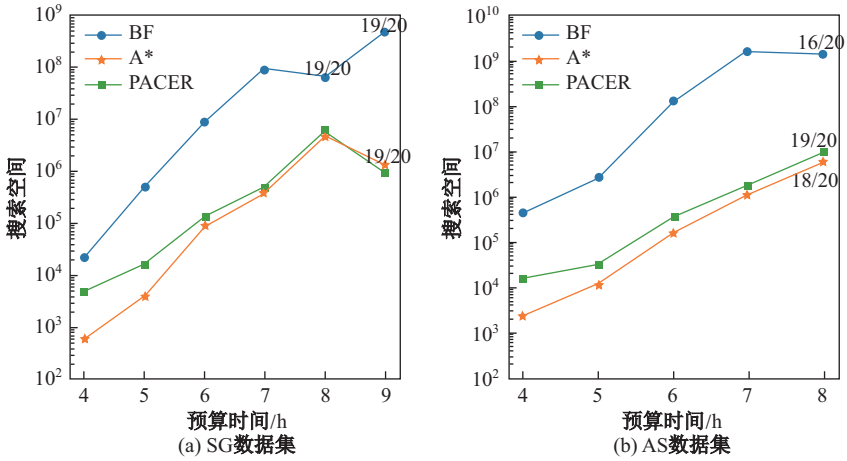


图 5 预算代价对搜索空间的影响

Fig. 5 Impact of budget on search space

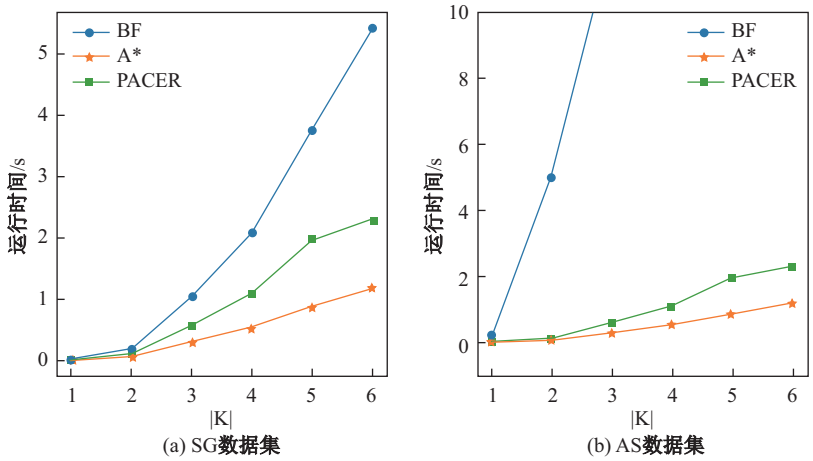


图 6 关键字数量对运行时间的影响

Fig. 6 Impact of |K| on runtime

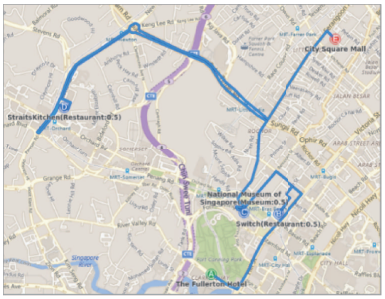
实质上, 基于 A* 的最优路径搜索算法能支持的查询关键字的数量, 与最后查询算法处理的候选节点数量相关. 若 Δ 设置为 4~7 h, 本文提出的 A* 算法能支持带更多的关键字查询.

4.3.3 关键字偏好对路径结果的影响

在这里, 通过改变查询关键字 K 的偏好值 Υ, 来说明关键字偏好对路径结果的影响. 对于查询 Q_1 =(The Fullerton Hotel, City Square Mall, {Park, Restaurant, Museum}, {0.1, 0.6, 0.3}, 360), 关键字 Park、Restaurant、Museum 的偏好值分别为 0.1、0.6、0.3, 在给定预算 360 min 的条件下求出的最优路径如图 7 所示, 右边的提示框用来描述路径信息. 由图可知, 推荐路径 R_1 会经过两个餐馆和一个博物馆, 路径收益值为 0.6, 路径代价为 287 min.

如果把关键字 Park、Restaurant、Museum 的偏好值调整为 0.6、0.2、0.2, 那么新的查询 Q_2 =(The Fullerton Hotel, City Square Mall, {Park, Restaurant, Museum}, {0.6,0.2,0.2}, 360) 的路径推荐如图 8 所示. 由图可知, 给 Q_2 推荐的路径 R_2 不同于 R_1. 由于查询设置

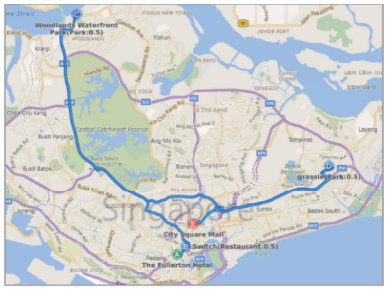
的Park偏好值大于Restaurant, 所以R_2经过的两个Park和1个Restaurant, 在预算不足的情况下, 没有覆盖关键字Museum.



A: The Fullerton Hotel→
B: Switch(Restaurant: 0.5)→
C: National Museum of Singapore (Museum: 0.5)→
D: StraitsKitchen(Restaurant: 0.5)→
E: City Square Mall;
路径收益=0.6 **路径代价**=287 min

图 7　查询Q_1的路径结果R_1

Fig. 7　Query result R_1 of Q_1



A: The Fullerton Hotel→
B: Switch(Restaurant: 0.5)→
C: Woodlands Waterfront Park (Park: 0.5)→
D: grassie(Park: 0.5)→
E: City Square Mall;
路径收益=0.55 **路径代价**=332 min

图 8　查询Q_2的路径结果R_2

Fig. 8　Query result R_2 of Q_2

5　总　　结

本文在构建的POI地图上进行基于用户偏好的最优路径搜索研究, 考虑在预算约束条件下, 关键字偏好和权重偏好对路径推荐的影响. 为了找到一条满足多样性特征的路径, 本文将目标函数定义为具有子模性质的覆盖函数. 在预处理阶段提出索引建立方法, 并在查询阶段利用索引结构过滤出候选节点集, 接着设计基于A* 的路径搜索算法来做最优路径查询, 并利用几个有效的剪枝策略加快算法的执行速度. 本文也通过设置多个实验测试A* 算法的查询性能, 实验结果表明本文算法在一定的查询条件下具有优越性.

[参　考　文　献]

[1] JUNGLAS I A, WATSON R T. Location-based services[J]. Communications of The ACM, 2008, 51(3): 65-69.

[2] JIANG Q, TENG W, LIU Y. ORSUP: Optimal route search with users' preferences [C]// MDM, 2019: 357-358.

[3] SHARIFZADEH M, KOLAHDOUZAN M R, SHAHABI C, et al. The optimal sequenced route query [J]. Very Large Data Bases, 2008, 17(4): 765-787.

[4] LI F, CHENG D, HADJIELEFTHERIOU M, et al. On trip planning queries in spatial databases [C]// Proceedings of the 9th International Symposium on Spatial and Temporal Databases (SSTD '05), 2005: 273-290.

[5] CHEN H, KU W, SUN M, et al. The multi-rule partial sequenced route query [C]// Advances in Geographic Information Systems, 2008: 1–10.

[6] KANZA Y, LEVIN R, SAFRA E, et al. Interactive route search in the presence of order constraints [J]. Very Large Data Bases, 2010, 3(1): 117-128.

[7] LI J, YANG Y D, MAMOULIS N, et al. Optimal route queries with arbitrary order constraints [J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(5): 1097-1110.

[8] 金鹏飞, 牛保宁, 张兴忠. 高效的多关键词匹配最优路径查询算法 KSRG [J]. 计算机应用, 2017, 37(2): 352-359.
[9] LIU H, JIN C, YANG B, et al. Finding Top-k Optimal Sequenced Routes [C]// International Conference on Data Engineering, 2018: 569-580.
[10] 鲍金玲, 杨晓春. 一种支持约束关系的高效的行程规划算法 [J]. 小型微型计算机系统, 2013, 34(12): 2702-2707.
[11] 吴澎, 朱家明. 基于多目标规划和智能优化算法的旅游线路设计研究 [J]. 数学的实践与认识, 2016, 46(15): 105-114.
[12] CAO X, CHEN L, CONG G, et al. Keyword-aware optimal route search [J]. Very Large Data Bases, 2012, 5(11): 1136-1147.
[13] LIU H, JIN C, YANG B, et al. Finding Top-k Shortest Paths with Diversity [J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(3): 488-502.
[14] LIANG H, WANG K. Top-k Route Search through Submodularity Modeling of Recurrent POI Features [C]// International Acm Sigir Conference on Research and Development in Information Retrieval, 2018: 545-554.
[15] CHEKURI C S, PAL M. A recursive greedy algorithm for walks in directed graphs [C]// Foundations of Computer Science, 2005: 245-253.
[16] ZENG Y, CHEN X, CAO X, et al. Optimal route search with the coverage of users' preferences [C]// International Conference on Artificial Intelligence, 2015: 2118-2124.
[17] DAI J, YANG B, GUO C, et al. Personalized route recommendation using big trajectory data [C]// International Conference on Data Engineering, 2015: 543-554.
[18] ELARINI K, VEDA G, SHAHAF D, et al. Turning down the noise in the blogosphere [C]// Knowledge Discovery and Data Mining, 2009: 289-298.
[19] KEMPE D, KLEINBERG J M, TARDOS E, et al. Maximizing the spread of influence through a social network [C]// Knowledge Discovery and Data Mining, 2003: 137-146.
[20] LESKOVEC J, KRAUSE A, GUESTRIN C, et al. Cost-effective outbreak detection in networks [C]// Knowledge Discovery and Data Mining, 2007: 420-429.
[21] YUAN Q, CONG G, MA Z, et al. Time-aware point-of-interest recommendation [C]// International Acm Sigir Conference on Research and Development in Information Retrieval, 2013: 363-372.
[22] CHO E, MYERS S A, LESKOVEC J, et al. Friendship and mobility: user movement in location-based social networks [C]// Proceedings of the 17th ACM SIGKDD International Conference, 2011: 1082-1090.

(责任编辑: 李万会)