

文章编号: 1000-5641(2017)05-0162-12

基于正则表达式的限制性路径规划

王 婧, 刘辉平, 金澈清

(华东师范大学 计算机科学与软件工程学院, 上海 200062)

摘要: 传统的路径规划算法大多以长度、时间或代价等为度量标准搜索起止点间的最优路径, 不适用于解决有位置限制的路径规划需求, 如搜索有序或无序地经过全部或部分用户指定的位置点或位置点类别的最短路径. 本文主要针对这类应用场景, 利用正则表达式表示复杂的限制性路径规划需求, 形式化定义了基于正则表达式的限制性路径规划问题并设计了通用的解决框架, 在此框架基础上提出了基本的限制性路径规划算法 BCRP(Basic Constrained Route Planning) 以及加入剪枝策略的改进的限制性路径规划算法 ICRP(Improved Constrained Route Planning), 有效减少了搜索空间. 最后通过在真实路网数据上的实验结果证明了方法的高效性.

关键词: 限制性路径规划; 正则表达式; 最短路径

中图分类号: TP391 **文献标志码:** A **DOI:** 10.3969/j.issn.1000-5641.2017.05.015

Constrained route planning based on the regular expression

WANG Jing, LIU Hui-ping, JIN Che-qing

(School of Computer Science and Software Engineering,
East China Normal University, Shanghai 200062, China)

Abstract: Traditional route planning algorithms, which mainly focus on metrics such as the distance, time, cost, etc. to find the optimal route from source to destination, are not suitable for solving route planning requirements with location constraints. For example, finding the shortest path passing the whole or a part of user-defined location categories in order or disorder. Mainly focusing on these scenarios, this paper formalizes the constrained route planning problem on the basis of the regular expression generated by user requirements and gives a general framework to solve this problem. Based on this, a basic constrained route planning algorithm (BCRP) and an improved constrained route planning algorithm (ICRP) are proposed while ICRP reduces the search space using pruning rules. Finally, extensive experiments on real road network datasets demonstrate the efficiency of our proposal.

收稿日期: 2017-06-20

基金项目: 国家重点研发计划重点专项(973)(2016YFB1000905); 国家自然科学基金(61370101, 61532021, U1501252, U1401256, 61402180)

第一作者: 王 婧, 女, 硕士研究生, 研究方向为基于位置的服务. E-mail: jingwang@stu.ecnu.edu.cn.

通信作者: 金澈清, 男, 博士生导师, 研究方向为基于位置的服务. E-mail: cqjin@sei.ecnu.edu.cn.

Key words: constrained route planning; the regular expression; the shortest path

0 引言

路径规划问题旨在图中寻找一条或多条从起点到终点并满足一定度量标准如长度、时间、代价或油耗等的最优路径, 已经在 GPS 导航、智慧交通、物流运输、无人机驾驶、旅游路线设计和通信路由等多个领域得到了广泛应用^[1-6]. 传统的路径规划算法主要根据度量标准定义的边的权重来寻找具有最小权重和的路径作为最优路径, 除此之外现实生活中还存在大量对位置有多样性限制的路径规划需求. 如图 1 所示为一个道路网络的一部分, 每个点代表一个兴趣点 (POI, Point of Interest), 表示该点所属的类别; 每条边表示兴趣点间的一条单向道路, 边上的数字表示该边的权重(这里为道路的长度). 假设 Tom 结束了一天在 A 点的工作后, 想先去一家餐厅吃饭, 然后去电影院或酒吧放松一下(由于时间有限这二者只能择其一), 最后回到位于 H 点的家, 他希望规划一条满足上述位置限制要求的最短路径.

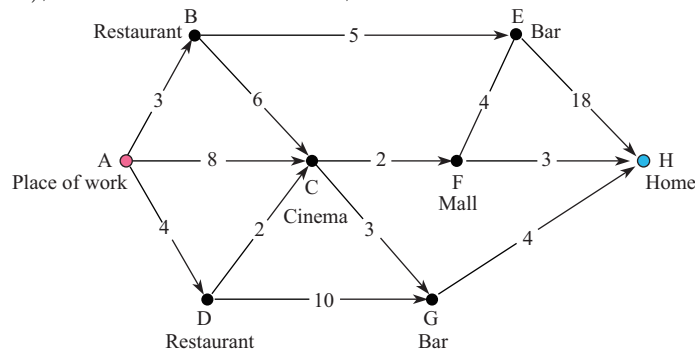


图 1 限制性路径问题的一个示例

Fig. 1 An example of the constrained route planning problem

日常生活中像这样对位置点和位置点类别有顺序、多选等限制的路径规划需求十分常见, 也已有一些研究工作关注: 最优路径查询^[7-13]搜索从查询点出发经过指定类别集合的最短路径, 主要关注位置点类别的无序、有序、部分有序等限制; 文献 [14-16] 查找从起点到终点, 途径指定关键词集合且满足行程预算的路径, 忽略了途径点的顺序限制, 加入关键词匹配度、路径流行度等考量因素更适于具体的应用场景. 现有工作仅是研究限制性路径规划问题的子集或者特殊情况, 不能解决对位置点既有经过的顺序限制, 又有多选需求的路径规划.

鉴于此, 本文研究在图中搜索从给定起点到终点, 并对途径位置点和类别有顺序、多选等多重限制的最优路径问题. 考虑到此类路径规划需求中富含多种语义信息, 引入正则表达式来表达查询. 正则表达式是表示字符串匹配模式的字符序列, 常用来检索或替换特定文本, 利用其已有的运算符定义可以表达路径规划查询中的多种限制, 且其解析后的有限自动机也可以辅助筛选符合条件的路径. 首先, 本文形式化定义了基于正则表达式的限制性路径规划问题 CRPP (Constrained Route Planning Problem); 接着结合自动机理论和 Dijkstra 最短路径算法, 提出了一个解决 CRPP 的通用框架; 在此基础上设计了一个基本的限制性路径规划算法 BCRP (Basic Constrained Route Planning), 并通过加入剪枝策略提出了改进的限制性路径规划算法 ICRP (Improved Constrained Route Planning), 减少了搜索空间并提升了处

理性能.

本文的主要贡献如下.

(1) 提出并形式化定义了基于正则表达式的限制性路径规划问题, 即找到图中从指定起点到终点, 并满足个性化位置限制条件, 如顺序或无序经过给定位置点或类别集合的全集或子集的最短路径, 利用正则表达式完整准确地表达复杂的限制性路径查询;

(2) 设计了一个解决 CRPP 的通用框架, 并基于此框架提出了基础的和改进的限制性路径规划算法, 改进的限制性路径规划算法通过剪枝策略显著提升了时间和空间效率;

(3) 应用真实数据集的实验结果表明方法的可行性和高效性.

本文内容组织如下: 第 1 节介绍相关工作, 第 2 节给出了问题的形式化定义, 第 3 节叙述了通用的解决框架, 在此框架的基础上第 4 节阐释了两种限制性路径规划算法 BCRP 和 ICRP, 第 5 节介绍了实验及实验结果, 最后第 6 节总结了全文.

1 相关工作

最优路径查询起源于最短路径问题, 由于考虑了多种限制性因素, 因此相比最短路径问题更有实际意义. 李飞飞等^[7]提出了在空间数据库中的路径规划查询 TPQ (Trip Planning Query), 空间数据库中的每个空间对象有位置和类别信息, TPQ 的目的是寻找从起点到终点, 经过给定点类别集合中每个类别至少一个空间对象的最短路径, 是广义旅行商问题的特例. TPQ 中用户不能指定希望经过的类别顺序, 且文献 [7] 采用近邻搜索的思想给出问题的近似解. Sharifzadeh 等人^[8]研究的最优顺序路径查询 OSR (Optimal Sequenced Route Query), 用来寻找从某一点开始, 严格按照指定顺序经过给定点类别集合中每个类别至少一个点的最短路径, 并给出了在欧式空间适用的 LORD 和 R-LORD 算法以及在路网中适用的 PNE 算法. 文献 [9-10] 通过研究 OSR 问题空间的几何学性质, 基于泰森多边形对给定的类别顺序进行预处理, 构建了一系列 AW-Voronoi Diagrams 来有效解决 OSR 查询. 文献 [11] 研究的通用最短路径问题本质上为指定终点且严格有序的 OSR, 使用了动态规划的思想更适用于路网等大规模图中. 不同于 OSR 定义点类别的全序, Chen 等人^[12]提出了一种多规则部分有序路径 MRPSR (Multi-rule Partial Sequenced Route) 查询, 只有部分有序类别集合被定义. 文献 [13] 解决从起点出发经过一个用户定义类别集合的最优路径查询, 支持不同类别间的部分顺序限制. 上述研究主要关注最短路径上位置点的类别特征, 仅支持对位置点类别及其经过顺序的限定, 不能对位置点和类别进行多选等要求.

文献 [14] 提出了一种多近似关键词路径 MAKR (the Multi-Approximate-Keyword Routing) 查询, 查询中不仅定义了起点和终点, 也定义了一组(关键词, 阈值)键值对, 旨在搜索从起点到终点, 至少经过每个关键词的一个空间对象的最短路径, 且该空间对象与此关键词的匹配度需高于指定阈值. 关键词感知的最优路径查询 KOR (Keyword-aware Optimal Route Query)^[15-16]综合考虑了关键词覆盖条件、代价约束和路径流行度三方面因素, 查找从起点到终点, 经过指定关键词集合中的点同时满足行程预算且流行度最大的路径, 实现近似求解. 上述针对关键词的路径查询中, 每个位置点关联一个或多个关键词, 本质上也可以转化为类别信息进行处理. 不同的是, MAKR 和 KOR 只考虑了需经过的点所属的关键词, 并没有考虑经过的顺序限制. 而加入关键词匹配度、路径流行度等度量标准只适合包含这些特定信息的应用场景.

同样使用正则表达式的手段定义限制性路径问题, 文献 [17] 研究了形式化语言限制性最短路径问题. 不同于本文考虑对位置点的限制, 文献 [17] 考虑的是对组成路径的边的限制,

即结果路径上边的标签需满足正则表达式。

通过分析发现现有研究工作主要集中于类别信息,有些没有考虑顺序限制,有些给出的是问题的近似解,有些额外考虑了路径流行度等因素仅适用于具体场景的应用,有些关注的是对路径上边的限制,还没有工作考虑位置点及类别的多选需求。本文面向更加通用的限制性路径规划问题,全面考虑了位置点和类别的经过顺序、多选等限制条件,上述问题均可转化为本文定义的基于正则表达式的限制性路径规划问题加以解决。

2 问题定义

定义 1(图) 给定有向图 $G(V, E)$, V 代表节点集, E 代表边集。每个点 $v \in V$ 有一个类别信息 $v.c$, 每条边 $e = (v_i, v_j) \in E$ 为从 $v_i \in V$ 到 $v_j \in V$ 的有向边。权值 $w(e)$ 取决于使用的度量标准,如在最短路径场景中表示 e 的长度,在最短时间场景中表示经过 e 所需的时间。

G 为广义上的图。当 G 是路网时可简化为仅含有 POI 的子图,此时 V 是 POI 点的集合, E 是这些 POI 间最短路径的集合。

定义 2 (限制性路径查询) 限制性路径查询 q 是一个含有 5 种元素的正则表达式: 起点、终点、 V 中位置点的集合、 V 中位置点类别的集合以及操作符 \cdot 、操作符 $|$ 、操作符 $*$ 、操作符 $+$ 和 $()$ 。①连接操作符 \cdot 可以省略,表示位置点或类别间的顺序关系;②选择操作符 $|$ 将可能的选择划分开来,表示多个位置点或类别任选其一;③ Kleene 闭包操作符 $*$ 表示经过其前面的位置点或类别零次或多次;④正闭包操作符 $+$ 表示至少经过其前面的位置点或类别一次;⑤圆括号 $()$ 定义操作符的范围和优先级。

通过上述元素的排列组合,限制性路径查询可以表达丰富的限制性路径规划需求。由于可以将一个位置点视为一个单独的类别,因此位置点的处理可以转化为类别的处理,下面仅以位置点类别为操作对象介绍限制性路径查询及其处理方法。

设正则表达式中第一个点为起点 s , 最后一个点为终点 t , 类别集合 $C = \{c_1, c_2, \dots, c_n\}$, 则 q 可以表示从 s 到 t 的路径且: ①严格有序经过 C 通过 $sc_1c_2 \dots c_nt$; ②以任意顺序经过 C 通过 $s(c_1c_2 \dots c_n)|(c_2c_1 \dots c_n)| \dots |(c_nc_1 \dots c_2)t$; ③经过 C 中任一类别通过 $sc_1|c_2| \dots |c_nt$; ④经过 C 中任一类别零次或多次通过 $s(c_1|c_2| \dots |c_n)^*t$; ⑤经过 C 中任一类别至少一次通过 $s(c_1|c_2| \dots |c_n)^+t$; ⑥以不同限制条件经过 C 通过上述规则的不同组合。如图 1 例子中的限制性路径查询可表示为 $q = work \cdot Restaurant \cdot (Cinema|Bar) \cdot home$, 其中 $work, home$ 是具体的位置点, $Restaurant, Cinema, Bar$ 代表 POI 类别。

值得注意的是,闭包操作符 $*$ 和 $+$ 在实际最短路径规划中几乎不会出现。首先是日常生活中人们更倾向于要么经过某个位置点或类别要么不经过;其次在不含负权的图中,经过某点多次的路径势必不短于与该路径其他部分相同而仅经过该点零次或一次的路径。如果出现上述需求,可以利用下列两条规则将含有闭包操作符 $*$ 和 $+$ 的查询化简:

$$sc_i(c_j)^*t \implies sc_it, \quad (1)$$

$$sc_i(c_j)^+t \implies sc_ic_jt. \quad (2)$$

规则 (1) 的含义是某一个位置点或类别出现零次或多次相当于对路径查询没有影响,可以从查询中去掉;规则 (2) 的含义是某一个位置点或类别出现一次或多次可以根据最短路径性质化简为出现一次。由此下文对限制性路径查询的处理主要考虑其他 3 种符号操作。

在传统正则表达式和有限自动机理论中,当且仅当字符串中的每个字符都在正则表达式中定义,才会被该正则表达式生成的有限自动机接受。如只有“ACFT”或“ACGI”才能通

过 $AC(F|G)I$ 生成的有限自动机, 相差一个字符都是不可接受的. 但 CRPP 中查询仅包含用户关心的位置点或类别, 且在查询中列出所有位置点或类别也是无意义和不现实的. 这是将正则表达式应用到 CRPP 时的关键问题, 将在 4.1 节陈述解决方法.

定义 3(限制性路径规划问题) 给定一个有向图 $G(V, E)$ 和一条限制性路径查询 q , 限制性路径规划问题 $CRPP(G, q)$ 在图 G 中寻找满足查询 q 的最短路径.

Tom 的限制性路径规划问题即: 在图 1 所示的图中找到满足 $q = work \cdot Restaurant \cdot (Cinema|Bar) \cdot home$ 的最短路径. 实际上理想的路径为长度为 11 的 $A \rightarrow D \rightarrow C \rightarrow F \rightarrow H$.

3 查询处理框架

针对上述限制性路径规划问题, 通过正则表达式表达的限制性路径查询, 结合 Dijkstra 最短路径算法, 本文提出了一个通用的查询处理框架. 选择 Dijkstra 算法的原因在于: Dijkstra 算法是最具代表性的求解非负带权图的最短路径算法, 程序设计简单, 通用性强, 是现有众多最短路径算法的基础, 方便扩展到其他最短路径算法上. 本框架首先解析查询 q , 运用 Thompson 算法将正则表达式转化为非确定性有限自动机 NFA, 再用子集构造法将 NFA 确定化为确定性有限自动机 DFA, 然后将 DFA 最小化, 最后将解析结果 DFA 与 G 一起作为限制性路径规划算法的输入, 通过一个类 Dijkstra 算法来计算 G 中满足 q 的最短路径.

NFA 和 DFA 都用状态转移表表示, 给定状态和类别, 转移的结果为状态转移表中(状态, 类别)的值, 即下一个状态; 如果表中不存在对应的项, 该类别就不是针对当前状态的一个可转移类别. 对查询 $q = work \cdot Restaurant \cdot (Cinema|Bar) \cdot home$ 而言, 解析后的 DFA 可以表示为图 2 的状态转移表或图 3 的状态转移图. 图 3 中圆圈表示状态, 初始状态冠以“ \Rightarrow ”, 双圈表示终结状态, 箭弧表示状态转移的方向, 弧上的标记表示该状态转移的输入.

(1, work) \rightarrow 2
(2, Restaurant) \rightarrow 3
(3, Cinema) \rightarrow 4
(3, Bar) \rightarrow 5
(4, home) \rightarrow 6
(5, home) \rightarrow 6

图 2 示例查询解析后的状态转移表

Fig. 2 The state transition table

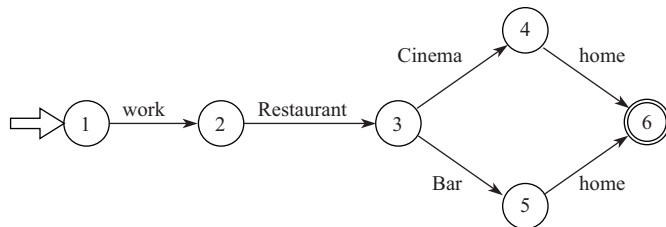


图 3 示例查询解析后的状态转移图

Fig. 3 The state transition diagram

因此可以通过状态转移表来检查路径是否符合限制性条件: 当且仅当以该路径上点的顺序可以驱动有限自动机从初始状态到接受状态, 即终结状态, 那么这条路径是符合限制性路径查询的路径。

4 限制性路径规划算法

在上节提出的查询处理框架的基础上, 本节介绍限制性路径规划算法, 包括基本算法 BCRP 和它的改进算法 ICRP。

4.1 基本的限制性路径规划算法BCRP

BCRP 算法以类 Dijkstra 算法为主线来寻找最短路径, 同时用限制性路径查询解析后的状态转移表过滤不符合限制条件的路径。不直接使用 Dijkstra 算法的原因在于: Dijkstra 算法中被标记为访问过的点不会在后续过程中被扩展, 因为被标记时对应的路径一定是从 s 到该点的最短路径。而在 CRPP 中, 两点间的最短路径不一定是符合限制条件的路径, 即 Dijkstra 算法不足以解决 CRPP。因此在更新 s 到某点的距离时, 也应将其他可达路径保存, 这样所有可能成为符合限制条件最短路径的路径都会被考虑在内。对此本文提出的类 Dijkstra 算法为: 首先设置 s 为当前点并标记, 接着更新所有标记点的直接可达点到 s 的距离并保存其路径, 持续选取目前拥有最短距离的点为当前点并标记, 直到到达状态转移表的终结状态, 此时对应的路径即是符合限制性条件的最短路径。

在定义 2 中已经提到, 传统的正则表达式是字符串的完全匹配, 而 CRPP 中查询只包含用户关心的位置点或类别, 因而是一种部分匹配。对此, 定义一个额外的任意状态 arb , 表示不在状态转移表中的一个不关心的状态; 且为每条路径的状态转移增加 $nowstate$ 和 $paststate$ 两个属性, $nowstate$ 表示扩展过程中的当前状态, $paststate$ 表示上一个非任意状态。对于恰好满足当前状态转移条件的点, 直接按照状态转移表进行转移; 对于其类别不在当前状态转移条件中的点, 由于其后节点有可能满足状态转移条件, 因此先将其 $nowstate$ 标记为 arb 。当扩展 $nowstate$ 为 arb 的路径时, 根据其 $paststate$ 属性进行转移。

算法 1 BCRP(G, T)

输入: 图 G , 状态转移表 T

输出: 图 G 中满足正则表达式的最短路径及其长度

```

1: 获取  $T$  的开始状态  $startState$  和终结状态  $endState$ ;
2: 创建  $item_s(s, path.add(s), 0, T(startState, s.c), startState)$  并插入优先级队列  $h$ ;
3: while  $h$  不为空 do
4:   取出  $h$  的队首元素  $item_i$ ;
5:   if  $item_i.nowstate = endState$  then
6:     return  $item_i$ ;
7:   else
8:     BCRPGetLinks( $G, T, h, item_i$ );
9:   end if
10: end while

```

算法 2 BCRPGetLinks($G, T, h, item_i$)**输入:** 图 G , 状态转移表 T , 优先级队列 h , 待扩展的队首元素 $item_i$

```

1: for  $item_i.vnode$  的所有直接可达连接点  $u$  do
2:   创建  $item_k(u, item_i.path.add(u), item_i.mindist + w(item_i.vnode, u), nowstate, paststate)$ ;
3:   SetItemState( $T, u, item_i, item_k$ );
4:   将  $item_k$  插入  $h$ ;
5: end for

```

算法 3 SetItemState($T, linknode, item_i, item_k$)**输入:** 状态转移表 T , 当前点 $linknode$, 待扩展的队首元素 $item_i$, 新的扩展元素 $item_k$

```

1: if  $item_i.nowstate = arb$  then
2:    $item_k.paststate \leftarrow item_i.paststate$ ;
3:   if  $T$  中存在转移( $item_i.paststate, linknode.c$ ) then
4:      $item_k.nowstate \leftarrow T(item_i.paststate, linknode.c)$ ;
5:   else
6:      $item_k.nowstate \leftarrow arb$ ;
7:   end if
8: else
9:   与上述过程相同, 区别在于根据  $item_i.nowstate$  的转移情况判断;
10: end if

```

为了更好地实现上述方法, 使用最短路径长度增序的优先级队列 h 来存储 s 到各点的路径. h 中每个元素是一个五元组 $item(vnode, path, mindist, nowstate, paststate)$, $vnode$ 代表当前节点, $path$ 和 $mindist$ 分别是 s 到 $vnode$ 的路径及长度. 算法 1 展示了 BCRP(G, T) 算法: 先从起点 s 开始扩展, 只要 h 非空, 就不断取出队首元素, 直到其 $nowstate$ 到达终结状态, 这时队首元素代表的路径即为符合正则表达式的最短路径. 其中, 通过调用算法 2 BCRPGetlinks($G, T, h, item_i$) 扩展当前点的所有连接点并将它们加入到 h 中, 算法 3 SetItemState($T, linknode, item_i, item_k$) 根据上述规则设置新扩展元素的转移状态.

BCRP 算法的执行过程可以通过优先级队列的变化过程表示. 图 4 为 BCRP 算法针对图 1 所示的限制性路径规划问题的前五步执行过程(深色元素表示此时的最短路径, 需在下一步被扩展). 实际上 BCRP 相当于持续找下一条 s 到 t 的最短路径然后检查其是否符合限制要求, 直到当前最短路径是符合条件的, 故结果的正确性是显而易见的, 但也因此十分低效. 下节介绍的 ICRP 算法主要针对此问题, 通过加入剪枝策略进行改进.

第1步				
vnode	path	mindist	nowstate	paststate
A	A	0	2	1

第2步				
vnode	path	mindist	nowstate	paststate
B	A→B	3	3	2
D	A→D	4	3	2
C	A→C	8	arb	2

第3步				
vnode	path	mindist	nowstate	paststate
D	A→D	4	3	2
C	A→C	8	arb	2
C	A→B→E	8	5	3
C	A→B→C	9	4	3

第4步				
vnode	path	mindist	nowstate	paststate
C	A→D→C	6	4	3
C	A→C	8	arb	2
E	A→B→E	8	5	3
C	A→B→C	9	4	3
G	A→D→G	14	5	3

第5步				
vnode	path	mindist	nowstate	paststate
C	A→C	8	arb	2
E	A→B→E	8	5	3
F	A→D→C→F	8	arb	4
B	A→B→C	9	4	3
G	A→D→C→G	9	arb	4
G	A→D→G	14	5	3

图 4 图 1 中例子的 BCRP 算法执行过程

Fig. 4 BCRP algorithm execution process of the example in Fig. 1

4.2 改进的限制性路径规划算法ICRP

通过分析发现在BCRP运行过程中有大量的冗余项, 即队列中已存在比该项更优的项被加入优先级队列. 这不仅是对内存和计算资源的浪费, 更重要的是在后续的步骤中冗余项可能被扩展, 将导致持续的不必要的计算. 如在图4的第4步, 当考虑加入 $item_j(C, A \rightarrow D \rightarrow C, 6, 4, 3)$ 时, 队列中已经存在了一个关于点 C 的项 $item_i(C, A \rightarrow B \rightarrow C, 9, 4, 3)$, 两项的 *nowstate* 和 *paststate* 属性均相同意味着满足相同的限制条件, 而 $item_i$ 的路径长度更长, 故 $item_i$ 是一个冗余项. 基于上述观察, 设计了以下两种剪枝策略.

剪枝策略 当考虑同一点 v 的两条路径 (即优先级队列中的两项) $item_i$ 和 $item_j$ 时:

(1) 如果 $item_i.nowstate = item_j.nowstate \neq arb$, 更长的候选路径应该被剪枝;

(2) 如果 $item_i.nowstate = item_j.nowstate = arb$ 且 $item_i.paststate = item_j.paststate$, 更长的候选路径应该被剪枝.

证明: 由于两条路径的当前扩展结点均为 v , 假设 $item_j.mindist < item_i.mindist$, 从 s 到 t 符合条件的最短路径为 $item_i.path + Path(v, t)$, $Path(v, t)$ 代表从 v 到 t 的路径 (这里+表示路径的连接), 即可以通过 $Path(v, t)$ 从 $item_i.nowstate$ (当 $item_i.nowstate \neq arb$ 时) 或 $item_i.paststate$ (当 $item_i.nowstate = arb$ 时) 到达 $endState$. 又由于两条路径处于相同的状态 (当 $nowstate \neq arb$ 时为 $nowstate$, 否则为 $paststate$), 故 $item_j$ 也可以通过 $Path(v, t)$ 到达 $endState$, 即 $item_j.path + Path(v, t)$ 也符合限制性条件. 由于 $item_j.mindist < item_i.mindist$, 那么 $item_j.path + Path(v, t)$ 比 $item_i.path + Path(v, t)$ 短, 与条件冲突. 即如果 $item_j.path + Path(v, t)$ 不是符合限制性条件的最短路径, $item_i.path + Path(v, t)$ 也不可能是. 因此 $item_j$ 总是比 $item_i$ 更优, 将 $item_i$ 剪枝不会影响算法的正确性.

为实现上述剪枝策略, 除为所有点维护一个全局优先队列 h 外, 还为每个点 v 设计一个以最短路径长度增序的局部优先级队列 *ownheap* 来存储所有 s 到 v 的路径. 全局优先队列中仅需存储处理中的节点, 每次取当前最短路径即为从全局队列中取出队首节点的队首元素. 算法4展示了ICRP算法的处理流程, 与BCRP类似, ICRP也从 s 点开始扩展, 持续取全局优先级队列队首元素直到其到达终结状态, 此时对应的路径即为符合条件的最优路径. 如果 h 为空则不存在符合条件的路径, 否则将通过算法5中的ICRPGetlinks($G, T, h, node_i, item_i$)扩展队首元素, 扩展过程中会根据剪枝规则判断新的扩展路径是否是冗余项, 只有非冗余项才会保留在该节点自己的局部优先级队列中.

算法4 ICRP(G, T)

输入: 图 G , 状态转移表 T

输出: 图 G 中满足正则表达式的最短路径及其长度

```

1: 获取  $T$  的开始状态  $startState$  和终结状态  $endState$ ;
2: 将  $item_s(item_s.path.add(s), 0, T(startState, s.c), startState)$  插入  $s.ownheap$ , 将  $s$  插入  $h$ ;
3: while  $h$  不为空 do
4:   取出  $h$  的队首节点  $node_i$  的队首元素  $item_i$ ;
5:   if  $item_i.nowstate=endState$  then
6:     return  $item_i$ ;
7:   else
8:     if  $node_i.ownheap$ 不为空 then
9:       将  $node_i$ 插入 $h$ ;
10:    end if
11:    ICRPGetLinks( $G, T, h, node_i, item_i$ );
12:  end if
13: end while

```

算法 5 ICRPGetLinks($G, T, h, node_i, item_i$)**输入:** 图 G , 状态转移表 T , 全局优先级队列 h , 当前点 $node_i$, 待扩展的队首元素 $item_i$

```

1: for  $node_i$  的所有直接可达连接点  $u$  do
2:   创建  $item_k$  ( $item_i.path.add(u)$ ,  $item_i.mindist + w(item_i.vnode_i, u)$ ,  $nowstate$ ,  $paststate$ );
3:   SetItemState( $T, u, item_i, item_k$ );
4:   if  $u$  不在  $h$  中 then
5:     将  $item_k$  插入  $u$  的  $ownheap$ , 将  $u$  插入  $h$ ;
6:   else if 在  $u$  的  $ownheap$  中存在一项  $item_j$  使得  $item_j.nowstate = item_k.nowstate$  then
7:     if  $item_j$  和  $item_k$  的  $nowstate$  均为  $arb$  then
8:       if  $item_j.paststate = item_k.paststate$  then
9:         将拥有更短  $mindist$  的项保留在  $u$  的  $ownheap$  中;
10:      else
11:        将  $item_k$  插入  $u$  的  $ownheap$ ;
12:      end if
13:    else
14:      将拥有更短  $mindist$  的项保留在  $u$  的  $ownheap$  中;
15:    end if
16:  else
17:    将  $item_k$  插入  $u$  的  $ownheap$ ;
18:  end if
19: end for

```

同样针对图 1 中的例子, 图 5 展示了 ICRP 算法运行过程的前五步. 通过比较 ICRP 和 BCRP 的部分运行过程, 可以发现冗余项被排除. 由于 BCRP 的正确性是显而易见的, ICRP 通过使用剪枝策略减少了搜索空间, 只会提升算法效率而不会破坏算法的正确性, 因此 ICRP 的正确性也被证明.

第1步				
	path	mindist	nowstate	paststate
A	A	0	2	1
第2步				
	path	mindist	nowstate	paststate
B	A→B	3	3	2
D	A→D	4	3	2
C	A→C	8	arb	2
第3步				
	path	mindist	nowstate	paststate
D	A→D	4	3	2
C	A→C	8	arb	2
	A→B→C	9	4	3
E	A→B→E	8	5	3
第4步				
	path	mindist	nowstate	paststate
E	A→D→C	6	4	3
	A→C	8	arb	2
E	A→B→E	8	5	3
G	A→D→G	14	5	3
第5步				
	path	mindist	nowstate	paststate
C	A→C	8	arb	2
E	A→B→E	8	5	3
F	A→D→C→F	8	arb	4
G	A→D→C→G	9	arb	4
	A→D→G	14	5	3

图 5 图 1 中例子的 ICRP 算法执行过程

Fig. 5 ICRP algorithm execution process of the example in Fig. 1

5 实验与结果

本节使用真实的路网数据进行实验. 所有算法均由 Java 语言编程实现且运行在处理器为 Intel Core i5-4460 3.20 GHz, 内存 16 G, Windows 操作系统的 PC 机上.

5.1 实验设置

数据集 使用 California 真实路网数据^[7](21 048 个点, 21 693 条边), 为每个点随机分配一个正整数 c ($0 \leq c \leq 19$) 代表其所属类别, 并随机抽取部分节点及它们相关的边生成原图不同的子图(节点数目 num)来考察图规模对算法的影响以及算法在不同结构图上的适用性.

查询 为不失查询的一般性, 设 q 的形式化格式为 $s(c_1|c_2|\cdots|c_k)_1()_2\cdots()_{j-1}t$, 其中, s, t 分别代表起点和终点; 连接运算符分隔开多个括号, 表示需依顺序经过的多个类别; 每个括号中多个以选择运算符分隔开的类别表示选其一. 假设每条查询有 j 个需要顺序连接的类别, 每个类别有 k 种选择 (j, k 均为正整数). 实验首先使用默认参数, 通过随机选择 s, t 和类别来生成 20 条基本查询, 接着通过调整参数 j, k 生成新的随机查询, 最大限度地保证变量唯一性.

表 1 详细展示了实验的具体设置, 非特殊说明都采用默认值.

表 1 实验设置

Tab. 1 Settings of experiments		
参数	取值范围	默认值
num	5 k, 10 k, 15 k, 20 k	10 k
j	2, 3, 4	3
k	1, 2, 3	2

5.2 实验结果

由于第 4 节中已经证明了算法的正确性, 因此在实验阶段主要关注算法的有效性, 体现在查询响应时间和扩展结点数目两个方面. 查询响应时间代表算法的处理效率, 扩展结点数目表示有多少可能的路径被验证, 代表查询空间的规模. 由于据我们所知尚没有相关工作解决与我们定义相同的限制性路径规划问题, 因此本文主要对 BCRP 和 ICRP 算法进行对比实验来考察剪枝策略对算法有效性和性能的影响.

图 G 对算法的影响 通过抽取子图中点的数目 num 来考察 G 的结构和规模对算法的影响. 图 6(a) 和 7(a) 展示了随着图中点数的增多, 算法的搜索空间和处理时间也随之增加, 这是由于大图中两点之间最短路径中点的个数普遍多于小图. 同时当图规模增大时, BCRP 和 ICRP 间的性能差异也越来越大, 这是由于扩展阶段连接点的数目持续增加, ICRP 对不符合条件的路径进行剪枝, 而 BCRP 中不符合条件的路径会被持续扩展, 带来了大量无用计算.

限制性路径查询 q 对算法的影响 q 对算法的影响主要体现在查询的复杂性上, 即限制条件的多少. 实验主要考虑 q 中顺序经过类别数目 j 和每个类别可能的选择数目 k 这两个方面. 图 6(b)、7(b) 和 6(c)、7(c) 分别展示了 j 和 k 对算法的影响. 可以直观地看到, 随着 j 和 k 的增加, 算法的性能有一定下降. 对于 j 而言, 数目越多表示路径需要经过的类别越多, 那么算法需要检查限制条件的满足情况而进行的计算也随之增多; 对于 k 而言, 每个类别的选择越多意味着需要考虑更多的可能路径.

BCRP 和 ICRP 算法的有效性分析 假设运行时间超过 3 min 就是不可接受的, 在实验过程中发现, 只有在较小的图上进行较简单的查询时 BCRP 算法才可以运行出可以接受的结果, 如当 num 为 5 k 时, BCRP 算法尚能获得较好的性能, 但是增加到 10 k 时, 查询的不可接受率几乎达到 50%, 15 k 时超过 70%, 当 q 中限制性条件增加时也出现了相同的情况. 对比 ICRP 算

法, 查询的不可接受率为 0. 由上述实验结果可以看出, 在所有情况中 ICRP 算法的性能都优于 BCRP, 且随着图规模的增大和限制条件的增多, ICRP 的性能下降相比 BCRP 十分缓慢. 以上结果均表明了剪枝策略的有效性.

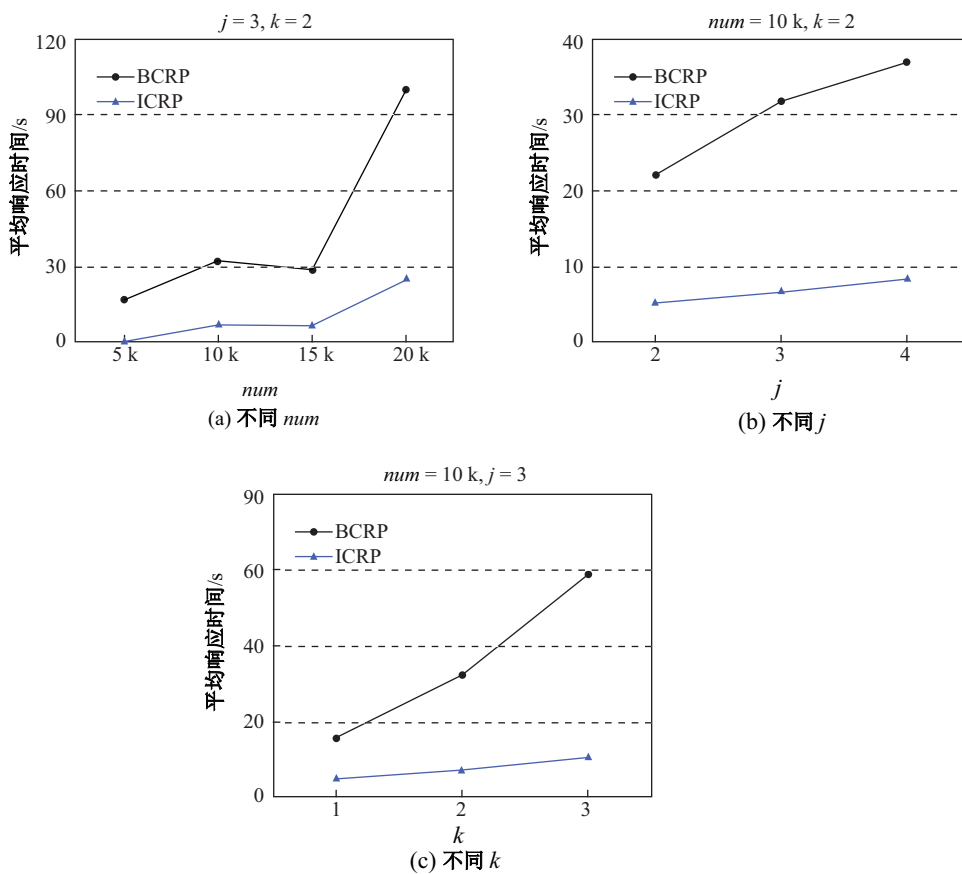
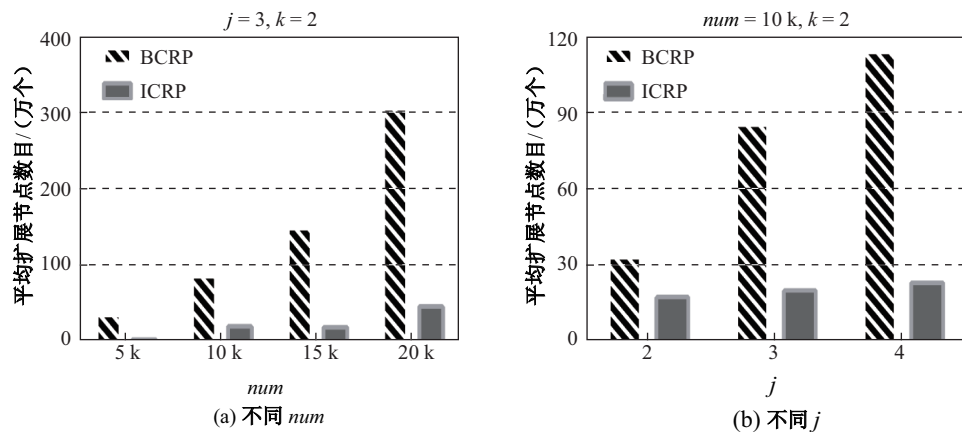
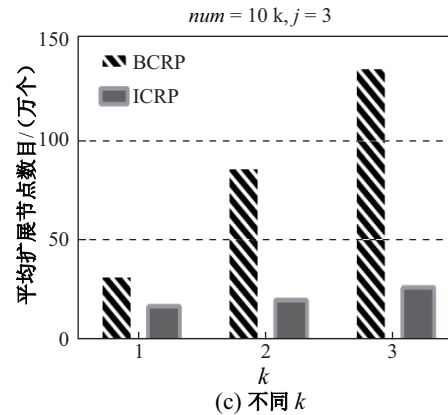


图 6 采用不同 num, j, k 时的平均响应时间

Fig. 6 Average response time when varying num, j and k



图 7 采用不同 num, j, k 时的平均扩展节点数目Fig. 7 Average number of expanded nodes when varying num, j and k

6 总结与展望

本文研究了限制性路径规划问题 CRPP, 首先通过利用正则表达式表示限制性路径查询的方式形式化定义了该问题, 接着设计了一种解决的通用型框架, 然后提出了基本的限制性路径规划算法 BCRP, 并在此基础上引入剪枝策略减少冗余路径和计算代价, 最后通过大量的实验分析表明本文的方法是可行且高效的. 未来的工作主要包括提升现有算法在大规模图中的适用性.

[参 考 文 献]

- [1] LIU H, JIN C, ZHOU A. Popular route planning with travel cost estimation [C]// Proceedings, Part II, of the 21st International Conference on Database Systems for Advanced Applications. New York: Springer-Verlag, 2016, 9643: 403-418.
- [2] YUAN J, ZHENG Y, ZHANG C, et al. T-drive: Driving directions based on taxi trajectories [C]// Sigspatial International Conference on Advances in Geographic Information Systems. New York: ACM, 2010: 99-108.
- [3] ZHENG Y, CAPRA L, WOLFSON O, et al. Urban computing: Concepts, methodologies, and applications [J]. ACM Transactions on Intelligent Systems and Technology, 2014, 5(3): 38.
- [4] ZHANG S, QIN L, ZHENG Y, et al. Effective and efficient: Large-scale dynamic city express [J]. IEEE Transactions on Knowledge & Data Engineering, 2016, 28(12): 3203-3217.
- [5] LU H C, LIN C Y, TSENG V S. Trip-Mine: An efficient trip planning approach with travel time constraints [C]//IEEE International Conference on Mobile Data Management. [S.l.]: IEEE, 2011: 152-161.
- [6] MALVIYA N, MADDEN S, BHATTACHARYA A. A continuous query system for dynamic route planning [C]// IEEE, International Conference on Data Engineering. [S.l.]: IEEE Computer Society, 2011: 792-803.
- [7] LI F, CHENG D, HADJIELEFTHERIOU M, et al. On trip planning queries in spatial databases [J]. Journal of Combinatorial Optimization, 2005, 31(1): 1-16.
- [8] SHARIFZADEH M, KOLAHDOUZAN M, SHAHABI C. The optimal sequenced route query [J]. The VLDB Journal, 2008, 17(4): 765-787.
- [9] SHARIFZADEH M, SHAHABI C. Additively weighted voronoi diagrams for optimal sequenced route queries [J]. Workshop on Spatio-Temporal Database Management, 2006.
- [10] SHARIFZADEH M, SHAHABI C. Processing optimal sequenced route queries using voronoi diagrams [J]. Geoinformatica, 2008, 12(4): 411-433.
- [11] RICE M N, TSOTRAS V J. Engineering generalized shortest path queries [C]// IEEE International Conference on Data Engineering. [S.l.]: IEEE Computer Society, 2013: 949-960.
- [12] CHEN H, KU W S, SUN M T, et al. The multi-rule partial sequenced route query [C]// ACM Sigspatial International Symposium on Advances in Geographic Information Systems. USA: DBLP, 2008: 1-10.

(下转第 235 页)

况下, 运维子系统可以有效地实现对业务子系统的性能监控、基于用户行为分析的系统优化和服务异常调试的三个运维目标.

5 结 论

基于日志收集与分析的运维模式为现代大型软件系统提供了丰富的运维信息, 从而有效地确保了大型业务系统运行期间的可靠性与稳定性. 基于现代运维服务的理念, 本文采用开源的 ELK 框架, 设计并实现了华东师范大学研究生院信息系统中的运维子系统, 解决了研究生院业务系统在实际使用中遇到的性能与负载监控、用户行为分析以及服务异常调试等问题. 实践证明, 本文实现的运维子系统不仅很好地满足了华东师范大学新一代研究生院信息系统的各种实际运维需求, 而且还确保了业务系统与运维系统之间的相互独立性, 这使得运维系统可以灵活地适应未来业务系统的升级改造.

[参 考 文 献]

- [1] 于长虹. 智慧校园智慧服务和运维平台构建研究[J]. 中国电化教育, 2015(8): 16-20+28.
- [2] 岑荣伟, 刘奕群, 张敏, 等. 基于日志挖掘的搜索引擎用户行为分析[J]. 中文信息学报, 2010(3): 49-54.
- [3] 郭岩, 白硕, 杨志峰, 等. 网络日志规模分析和用户兴趣挖掘[J]. 计算机学报, 2005(9): 1483-1496.
- [4] 邢东山, 沈钧毅, 宋擒豹. 从 Web 日志中挖掘用户浏览偏爱路径[J]. 计算机学报, 2003(11): 1518-1523.
- [5] 白俊, 郭贺彬. 基于 Elasticsearch 的大日志实时搜索的软件集成方案研究[J]. 吉林师范大学学报(自然科学版), 2014(1): 85-87.
- [6] 刘庆磊, 信师国, 李晓林. 虚拟技术在 IT 运维管理中的应用研究[J]. 信息技术与信息化, 2010(1): 43-45.
- [7] 王新, 马万青, 潘文林. 基于 Web 日志的用户访问模式挖掘[J]. 计算机工程与应用, 2006(21): 156-158.
- [8] 鲍钰, 黄国兴, 张召. 基于 Web 日志挖掘的网站结构优化方法[J]. 计算机工程, 2003(12): 82-84.
- [9] 周映, 韩晓霞. ELK 日志分析平台在电子商务系统监控服务中的应用[J]. 信息技术与标准化, 2016(7): 67-70.

(责任编辑: 李万会)

(上接第 173 页)

- [13] LI J, YANG Y, MAMOULIS N. Optimal route queries with arbitrary order constraints [J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(5): 1097-1110.
- [14] YAO B, TANG M, LI F. Multi-approximate-keyword routing in GIS data [C]// ACM Sigspatial International Conference on Advances in Geographic Information Systems. New York: ACM, 2011: 201-210.
- [15] CAO X, CHEN L, CONG G, et al. Keyword-aware optimal route search [J]. Proceedings of the VLDB Endowment, 2012, 5(11): 1136-1147.
- [16] CAO X, CHEN L, CONG G, et al. KORS: Keyword-aware optimal route search system [C]// IEEE, International Conference on Data Engineering. [S.l.]: IEEE, 2013: 1340-1343.
- [17] BARRETT C, JACOB R, MARATHE M. Formal language constrained path problems [J]. Computing, 1998, 30(3): 234-245.

(责任编辑: 张 晶)