软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(8):2543−2556 [doi: 10.13328/j.cnki.jos.005808]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

大规模路网图下关键词覆盖最优路径查询优化*

郝晋瑶, 牛保宁, 康家兴

(太原理工大学 信息与计算机学院,山西 晋中 030600)

通讯作者: 牛保宁, E-mail: niubaoning@tyut.edu.cn

摘 要: 游客倾向于采用个性化的旅游路线,规划这样的路线需要综合考量路径长度、路径开销和路径覆盖的兴趣点.关键词覆盖最优路径查询(KOR)就是用于规划这样的路线的一类查询,其处理过程通常包括预处理和路径拓展.由于路网图规模的不断扩大,现有算法预处理所需内存开销急剧上升,由于内存不足,导致较大规模的路网不能处理;路径拓展搜索空间快速膨胀,应用场景可扩展性与查询实时性难以保证.针对这些问题,提出一种大规模路网图下关键词覆盖最优路径查询算法 KORL.KORL 在预处理阶段将路网划分为若干子图,仅保存子图内路径和子图之间路径的信息,以减小预处理所需内存.在路径拓展阶段,综合运用最小代价剪枝、近似支配剪枝、全局优先拓展和关键词顶点拓展等策略对现有算法进行优化,以高效地搜索近似最优解.采用美国各地区的路网图,在16G内存环境下进行实验,突破了现有算法只能处理顶点数不超过25K路网图的限制.实验结果表明,KORL算法具有良好的可扩展性.

关键词: 个性化旅游路线;关键词覆盖最优路径;大规模路网图;路网图划分;最小代价剪枝

中图法分类号: TP311

中文引用格式: 郝晋瑶,牛保宁,康家兴.大规模路网图下关键词覆盖最优路径查询优化.软件学报,2020,31(8):2543−2556. http://www.jos.org.cn/1000-9825/5808.htm

英文引用格式: Hao JY, Niu BN, Kang JX. Optimization of keyword-aware optimal route query on large-scale road networks. Ruan Jian Xue Bao/Journal of Software, 2020,31(8):2543−2556 (in Chinese). http://www.jos.org.cn/1000-9825/5808.htm

Optimization of Keyword-aware Optimal Route Query on Large-scale Road Networks

HAO Jin-Yao, NIU Bao-Ning, KANG Jia-Xing

(College of Information and Computer, Taiyuan University of Technology, Jinzhong 030600, China)

Abstract: Visitors tend to choose personalized travel routes. Planning such a route requires a comprehensive consideration of the length and cost of the route, and the points of interest covered by the route. Keyword-aware optimal route query (KOR) is a typical query for this purpose. Processing a KOR consists of preprocessing and route expansion. With the scale of maps of road networks continues to expand, the overhead for preprocessing and the search space for route expansion increase rapidly. The scalability and the real-time responsiveness are hard to guarantee. To alleviate these pain points, an algorithm called keyword-aware optimal route query algorithm on large-scale road networks or KORL is proposed. In the preprocessing stage, KORL reduces memory requirements by partitioning the road network into subgraphs and stores only information about the routes inside and between subgraphs. In the route expansion stage, KORL combines four strategies, namely minimum cost pruning, approximately dominance pruning, global priority expansion, and keyword vertex expansion to efficiently search the approximate optimal solution. The road networks of various regions in the United States are used as experimental datasets and the experiments are run by the computer with 16 G memory. The limitation that existing algorithms can only handle the road network with the number of vertexes less than 25K is broken. Experiments show that KORL has sound scalability.

Key words: personalized travel route; keyword-aware optimal route; large scale road network; road network partitioning; minimum cost pruning

* 基金项目: 国家自然科学基金(61572345)
Foundation item: National Natural Science Foundation of China (61572345)
收稿时间: 2018-06-08; 修改时间: 2018-08-31; 采用时间: 2018-12-18

受益于地图服务和移动终端的普及和发展,游客在规划旅游路线时不仅将路径长度作为考量,同时会综合考虑其他方面的因素[1−5].其中,路径长度、路径开销和路径所覆盖兴趣点是游客普遍关注的 3 个最优路径选择因素[4].关键词覆盖最优路径查询(keyword-aware optimal route query,简称 KOR)[4,5]就是综合考虑这些因素规划路线的一类查询,KOR 可以描述为:游客希望从某个给定出发点开始,寻找一条可以包含所有计划去的兴趣点(如名胜古迹、娱乐美食),最终到达指定目的地的最优路线.同时,这一最优路线应当满足游客的行程预算,并且在总开销上越少越好.

KOR 本身是一个 NP-hard 问题[4],查询时间随关键词个数呈指数型增长.现有查询算法[4−7]的思路都是寻求近似最优解.其中,OSScalling 算法[4]提供了一套高效的算法框架.该框架把地图顶点与路径抽象成路网图,在预处理阶段,通过 Floyd 算法[8]得到每对顶点间的代价值和目标值数组,用以查询过程中判别剪枝和拓展路径;在路径拓展阶段,利用路径标签在拓展路径的同时剪除不满足阈值的中间结果以加快拓展,从可行解中选择目标值最小的路径作为最优解.

随着路网图规模的不断扩大,越来越多的兴趣点也被添加到路网图中[9,10].截至 2006 年,西欧地区路网图已拥有 18M 个顶点[11].现有算法的预处理数组占用内存随顶点数量增加呈多项式级别增长.在顶点数量为 1K 时,内存占用约为 20M;但当顶点数量达到 10M 时,内存占用为 2PB.这显然是大多数设备无法承受的.在路网图局部更新后,现有算法预处理必须全局更新数组,而大多数顶点间最小目标(代价)值的计算并不会受到新加入的顶点的影响,因此,更新过程会产生大量无用的计算.此外,在每轮路径拓展中,会有更多的待拓展顶点加入拓展中,严重影响了查询速度.

针对以上问题,本文提出一种大规模路网图下关键词覆盖最优路径查询算法 KORL(keyword-aware optimal route on large-scale road networks),分别从预处理存储方法和路径拓展策略上对现有算法进行改进.对于预处理中现有算法保存大规模路网图信息时占用海量内存的问题,本文采用分治思想,将路网图进行分割,并构建相应的树形索引结构,仅保存子图内路径和子图之间路径的信息,将预处理空间复杂度由 $O(n^2)$降低至 $O(n)$,极大节省了内存.对于路径拓展中存在大量无效拓展(拓展后路径代价超出阈值,从而路径被剪枝)的问题,本文通过对路网图结构的分析,发现从某一顶点到另一不同子图顶点必然经过两子图间的关联路径.基于此,提出最小代价剪枝策略.该策略利用最小中间路径代价值来快速剪枝所有完整路径,快速筛除大量无效待拓展顶点.

本文的创新点总结如下:

(1) 本文提出了分治存储的预处理方法.采用 PUNCH 算法[12]划分路网图,根据划分构建树形索引结构,在树形索引的叶节点上保存子图内部顶点间的信息,在根节点上保存各子图边界点间的信息,极大地减少预处理的内存占用.

(2) 本文提出了一种快速筛除无效待拓展顶点的策略——最小代价剪枝策略.在路径拓展前,首先计算有效待拓展子图(与拓展顶点所在子图间最小代价值不超过剩余代价阈值的子图),若待拓展顶点不属于有效待拓展子图,则不予拓展,避免大量无效拓展.

(3) 本文综合运用最小代价剪枝、近似支配剪枝、全局优先拓展和关键词顶点拓展等策略,加快了查询速度.

本文第 1 节介绍 KOR 相关研究.第 2 节介绍 KOR 的形式化定义与查询算法框架.第 3 节给出大规模路网图下关键词覆盖最优路径查询算法 KORL.第 4 节通过实验验证 KORL 算法良好的可扩展性.第 5 节总结全文.

1 相关工作

目前,多约束最优路径查询问题已得到广泛关注[4−7,13−16],KOR[4−7]是典型的一类多约束最优路径查询问题.在求解多约束最优路径查询问题时,关键是如何处理多约束之间的关系.Li 等人[13]通过平衡路径代价与长度所得的得分函数来寻找最优路径,但 KOR 旨在寻求一定代价阈值下目标最小的路径,不能简单使用得分函数来实现.张金增等人[14]利用区域子网划分技术和贪婪算法实现了路网环境下访问序列受限的多标签最优路径查询,但贪婪策略返回结果无法保证精度,无法适用于 KOR.鲍金玲等人[15]提出一种支持约束关系的高效的行程规划

算法,其问题定义中路网图中的顶点具有游览代价和游览分数,边具有游览代价.由于问题定义与 KOR 不同,因此不适用于 KOR.在此基础上,他们解决多约束的方法是认为顶点的目标值与代价值存在正比关系[16],通过降低约束维度来加快查询速度,而在现实中无法准确找到这种比例关系,因此不适用于 KOR.

Cao 等人[4]定义了 KOR 并给出了 3 个求解算法:OSScalling、BucketBound 和 Greedy.金鹏飞等人[6]提出 KSRG 算法,在 OSScalling 算法基础上加入关键词顶点拓展策略,并在后续工作[7]中引入 Skyline 路径集合,将路径拓展剪枝过程放在预处理中进行,进一步加快查询速度.这些算法求解过程中都需要产生大量拓展操作,而拓展操作的本质就是更新拓展路径的目标值、代价值(下文统称权值).为了避免在路径拓展阶段产生大量这样的计算而影响查询速度,需要预先将所有路径的权值计算并保存以便调用.

在预处理阶段,OSScalling 是通过 Floyd 算法计算路网图每对顶点间的最小权值并保存在数组中,但该方法空间复杂度为 $O(n^2)$.在大规模路网下,大多数设备都无法承受这样急剧膨胀的内存开销.此外,在路网图局部更新时,该方法必须对数组进行全局更新,造成大量时空开销.由于现有其他研究都是对路径拓展阶段进行改进,并未解决甚至加重(需要在预处理中生成所有顶点间 skyline 路径集合)预处理时空开销,因此本文采用一种分治存储的方法,将路网划分后仅保存子图内路径和子图之间路径的权值,极大减少了内存开销.

为了实现分治存储,本文采用划分路网图的方法以分块保存,因此需要选择合适的路网图划分算法.常见的划分算法有 METIS[17]、DiBaP[18]和 PUNCH[12]等.METIS 和 DiBaP 虽然能将路网图划分成大小平均的区域,但 METIS 丢失了很多密集区域顶点间的联系,DiBaP 划分路网图需要切割大量的边.PUNCH 算法仅切割了自然连接处少量的边,较好地保存了路网图的整体结构,更适合用于路网图划分.

在路径拓展方面,OSScalling 为后续研究提供了一套算法框架.BucketBound 在此基础上优化了拓展顺序.Greedy 采用贪婪策略,精度无法保证.KSRG 引入近似支配剪枝、关键词顶点拓展等策略,但这些算法均未解决拓展是否有效必须经过完整拓展后才能判别的问题.因此,本文在综合运用除 Greedy 外所有改进策略的基础上,提出最小代价剪枝,仅通过局部拓展结果判别拓展是否有效,极大加快了拓展速度.

2 KOR

KOR 的本质是三重约束下的路径拓展问题.约束条件分别为关键词全覆盖最优路径必须覆盖所有关键词、满足代价阈值(最优路径总代价不超过代价阈值)和目标最优化(最优路径为所有满足前两个约束下的目标值最优的路径).

本节给出 KOR 的形式化定义(第 2.1 节)并介绍 KOR 查询算法框架(第 2.2 节),作为后续讨论的基础.

2.1 问题定义

KOR 查询使用的路网图 G 由实际地图的顶点和边抽象所得,如图 1 所示.为方便筛出包含特定关键词的顶点,根据路网图中的关键词与顶点的对应关系构建出关键词倒排列表,如图 2 所示.

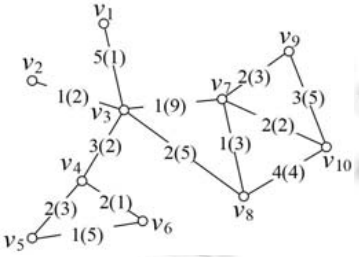


Fig.1 Road network structure

图 1 路网图结构

关键词	包含关键词顶点集
kw_1	v_3,v_7
kw_2	v_8,v_9
kw_3	v_1,v_5,v_{10}
kw_4	v_2,v_4,v_6

Fig.2 Keyword inverted list

图 2 关键词倒排列表

定义 1(路网图). $G(V,E)$由一个无向路网图的顶点集 V 和边集 E 构成,其中,任意顶点 $v\in V$ 表示路网图中的一个兴趣点.兴趣点的本质是一个空间文本对象,既包含空间经纬度,表示为 $v.loc$;又包含对该兴趣点的文本描

述关键词集合$\langle v.kw_1,v.kw_2,\ldots\rangle$,表示为 $v.\psi$.任意边 $e\in E$ 表示两个邻接顶点 v_i,v_j 之间的路径,记为(v_i,v_j).每条边包含两个属性值:目标值 OS 和代价值 BS.对于一条边 e,目标值表示其路径费用 $OS(e)$,在图中表示为括号外的数值;代价值表示其路径长度,记为 $BS(e)$,在图中表示为括号内的数值.

定义 2(路径). 路径 $p=(v_0,v_1,v_2,\ldots,v_n)$表示由 v_0 出发,顺序经过 $v_1,v_2,\ldots$直到 v_n 的一条路径.对于该路径,目标值 $OS(p)=\sum_{i=1}^{n}OS(v_{i-1},v_i)$, 代价值 $BS(p)=\sum_{i=1}^{n}BS(v_{i-1},v_i)$. 路径对关键词的覆盖 $p.\psi=\bigcup_{v\in p}v.\psi$.

定义 3(KOR). 在给定路网图 G 下,KOR 查询可表示为 $Q=(v_s,v_t,\psi,\Delta)$,其中,v_s 表示查询起点,v_t 表示查询终点,ψ是关键词集合,Δ是代价值阈值.假定 $P_{s,t}$ 表示从 v_s 到 v_t 的所有路径的集合,满足关键词覆盖和代价阈值的路径集合为 $P_{cand}=\{p|p\in P_{s,t}\wedge Q.\psi\subseteq p.\psi\wedge BS(p)\leqslant\Delta\}$,最优路径 $P_{opt}=\arg\min_{p\in P_{cand}}OS(p)$.

2.2 算法框架

OSScalling 算法是一个高效的 KOR 查询算法,被后续研究沿用为算法框架.该算法为加快拓展速度,在预处理时计算所有顶点间的权值并保存在内存中,方便拓展时大量相加运算的快速调用.为更好地表示路径拓展中间结果并实现拓展操作,该算法设计了一种数据结构:路径标签,将路径拓展转化为标签内关键词集合的并运算和权值的相加运算.为减少无效拓展,在拓展时进行代价阈值剪枝(当前路径代价值超过代价阈值)、可行解目标值剪枝(当前路径目标值超过当前最优可行解的目标值)和支配剪枝.为将问题时间复杂度限制在多项式级别,采用目标修正策略剪去很多目标值相近的路径.为快速拓展出可行解,该算法在每轮拓展时首先计算所有待拓展顶点的优先级,然后选择其中最有可能成为可行解的路径拓展下去,既避免了盲目拓展,又加快了拓展速度.该算法步骤如下.

- 预处理阶段

(1) 修正路网图中各边目标值[19].

(2) 通过 Floyd 算法[8]计算路网图每对顶点间权值数组.

- 路径拓展阶段

(1) 为图中每个顶点维护一个标签列表存放该顶点上的标签,维护一个全局队列用来存放待拓展的路径标签.在起点建立一个路径标签,加入队列;

(2) 开始循环拓展.取出队列中局部优先度最高的标签,计算该标签所有邻边待拓展顶点中优先级最高的顶点,并拓展至该顶点,生成新标签.判别新标签是否满足剪枝条件,若满足,则用该标签剪枝该顶点上与其起点相同的标签;

(3) 循环拓展直至某一标签已覆盖所有关键词,直接拓展至终点并判别是否满足剪枝条件,若满足,则保留为可行解并更新可行解目标值.重复此过程,直到循环结束.

定义 4(路径标签). 对每条从 v_s 到 v_i 的路径 p_i^k ,在 v_i 处构建一个路径标签 $L_i^k=(\psi,SOS,OS,BS)$.其中,ψ,SOS,OS,BS 分别代表了 p_i^k 覆盖关键词集合、修正目标值、目标值和代价值.

定义 5(标签操作). 当顶点 v_i 向另一个顶点 v_j 拓展时,会根据 L_i^k 和两顶点间路径的 OS,BS 创建一个新的标签 L_j^l ,其属性如下.

(1) $L_j^l.\psi=L_i^k.\psi\cup v_j.\psi$;

(2) $L_j^l.SOS=L_i^k.SOS+SOS(v_i,v_j)$;

(3) $L_j^l.OS=L_i^k.OS+OS(v_i,v_j)$;

(4) $L_j^l.BS=L_i^k.BS+BS(v_i,v_j)$.

定义 6(局部优先度). 假设 L_i^k,L_j^l 是路径标签,则两者的局部优先度可以定义为:(1) 若两者中有一个标签已经拓展到终点,则优先度更高,即 $i=t\wedge j\neq t,L_i^k>L_j^l$;(2) 在上述条件下,若两标签都是或都不是终点标签,覆盖更多关键词的标签优先度更高,即 $|L_i^k.\psi|>|L_j^l.\psi|,L_i^k>L_j^l$;(3) 在上述条件下,在覆盖关键词个数一致时,拥有更小修

正目标值的标签优先度更高,记为 $L_i^k.SOS < L_j^l.SOS, L_i^k > L_j^l$;(4) 在上述条件下,在覆盖关键词个数和修正目标值一致的情况下,拥有更小代价值的标签优先度更高,即 $L_i^k.BS < L_j^l.BS, L_i^k > L_j^l$.

定义 7(支配). 若两条起终点相同的路径 p_1,p_2 满足 $BS(p_1)\leqslant BS(p_2)\wedge SOS(p_1)\leqslant SOS(p_2)$,则称 p_1 支配 p_2.

支配剪枝策略通过剪除被支配的路径以避免大量无效后续拓展.为进一步提高查询速度,采用完全多项式时间近似策略(fully polynomial-time approximation scheme)[20,21]对目标值相近的路径进行目标修正后,结合支配剪枝策略,进一步减少中间路径的数量.

定义 8(目标修正). 给定代价阈值Δ,图边最小代价值 $b_{\min}$,最小目标值 $o_{\min}$,最大目标值 $o_{\max}$ 和介于(0,1)的参数ε,修正比例$\theta=\lfloor \varepsilon b_{\min}o_{\min}/\Delta \rfloor$,修正目标值 $SOS(v_i,v_j)=\lfloor OS(v_i,v_j)/\theta \rfloor$.每个顶点存储个数不超过一个上界.每个顶点最多存储 $2^{\lfloor \Delta/b_{\min} \rfloor \lfloor o_{\max}\Delta/\varepsilon b_{\min}o_{\min} \rfloor}$个路径标签.

定理 1. 使用目标修正策略所得最优解与实际最优解在一定误差范围内:$OS(p_{OS})\leqslant 1/(1-\varepsilon)OS(p_{opt})$[4].

3 KORL 算法

大规模路网图下,现有 KOR 查询算法会遇到两个问题:预处理内存开销过大、无效拓展路径过多.本文对其改进之处分别体现在预处理和路径拓展两个方面:改进预处理的关键问题是如何节省内存开销,改进拓展策略的关键问题是如何进一步剔除无效路径.

3.1 改进思路

现有算法无法解决大规模路网下 KOR 的根本原因是预处理索引结构不具有良好的可扩展性,其预处理采用单一数组保存所有顶点间权值,而数组大小会随顶点数增加呈多项式级别增长.大规模路网下,大多数机器无法承受该索引所需内存开销.因此,解决该问题的重点在于选择合适的索引结构,大幅减少内存占用且不丢失路网信息.

减少数组内存占用,必然要压缩路网信息,需要保证信息的可恢复性.这实质上是一个无损压缩问题.该问题的关键是确定可丢弃数据及其还原策略.我们注意到:计算大部分顶点间的权值需经过很多顶点,造成大量时间开销,保存这些顶点间权值又造成大量空间开销.这提示我们:把路网图划分成顶点数较少、相对更集中的子图,并将各子图间之间的关联关系体现在一张关联图中.在预处理阶段划分路网图,仅保存各子图和关联图的信息,丢弃大量不同子图顶点间权值信息,从而大幅减少内存占用;在路径拓展时,如果需要不同子图内顶点间权值信息,可以通过组合各子图与关联图上的路径实现复原.

因为预处理过程策略导致路径拓展时需要不断地做还原计算,所以查询算法必须避免大量的拓展操作产生.避免拓展主要体现在加强剪枝能力、优化拓展顺序和筛除无效待拓展顶点.在剪枝能力方面,我们采用近似支配策略剔除同源同终的目标值相近路径;在拓展顺序方面,我们优先选择导致目标值最小的待拓展顶点拓展;在筛除无效拓展顶点方面,我们仅向尚未覆盖的关键词对应的顶点拓展,同时,为了解决拓展顶点与待拓展顶点间存在多条拓展路径的问题,在预处理时计算顶点间的精简 skyline 路径集合,在拓展时仅沿这些路径拓展.

关键词顶点拓展策略还会导致一种现象:在向所有包含未拓展关键词的待拓展顶点拓展时,大部分路径都因不满足代价阈值剪枝条件而被剔除.在大规模路网下,关键词对应顶点个数也会大幅增加,这种现象更加明显.导致该现象的根本原因是:所有路径只有拓展到待拓展顶点后才能进行代价阈值剪枝,而拓展的过程涉及大量还原计算,造成大量时间开销.因此,该问题的关键是如何避免还原计算并筛除大量不满足代价阈值的待拓展顶点.我们观察到:还原路径权值的过程是将分段的路径组合并消去支配现象的过程,其中权值最大的一段路径大多是关联图上两子图间的路径,因此可以利用该部分路径权值来剪枝其他路径.为在拓展中快速调用两子图间的权值,需要在预处理中通过关联图的精简 skyline 路径集合与路网图的划分结果计算出所有子图间的最小权值数组.在每轮拓展前,通过子图间最小代价值判断有效待拓展子图,从而批量筛去大量无效待拓展顶点.

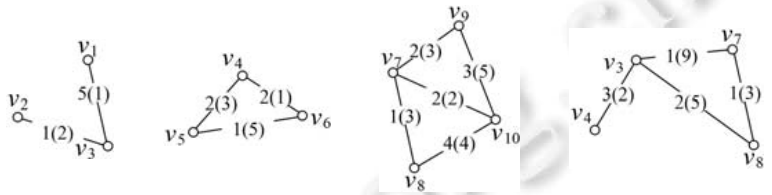
3.2 预处理

KORL 算法预处理阶段采用划分路网图,在此基础上构建树形索引的方法来组织路网信息.路网图划分对

构建树形索引有重要作用,优秀的路网图划分不仅需要保持子图大小近似一致,还需要尽量减少边界点数量.本文采用 PUNCH 算法[12]将路网划分,在划分的结果上构建该路网图的关联图[22].

定义 9(关联图). 给定一个图 $G(V,E)$,对其进行一种子图划分 $T=\{G_1,G_2,\dots,G_{|T|}\}$,其中,$G_i=(V_i,E_i)$.划分后:(1) $\bigcup_{i=1}^{|T|}V_i=V$;(2) 若 $i\neq j$,则 $V_i\cap V_j\neq\varnothing$;(3) $\forall u,v\in V_i$,若$(u,v)\in E$,则$(u,v)\in E_i$.若存在一条边(u,v),而 $u\in V_i,v\in V_j$,则称 u 是 V_i 的边界点.在一种划分下给定一个参数$\alpha(\alpha=1.1)$,关联图 $G'=(V',E')$的定义如下:(1) V'包含所有 V 中的边界点;(2) 假定 $e'\in E'$从 V_s 出发,到达 V_t,则在 G 中一定存在一条路径 p 从 V_s 到 V_t,使得 $SOS(p)=SOS(e')$,$BS(p)=BS(e')$;(3) 给定 G'中任一对顶点 V_s,V_t,一定存在一条从 V_s 到 V_t 的路径 p',近似支配原图 G 中从 V_s 到 V_t 的任意路径 p,即 $BS(p')\leqslant BS(p)$,$SOS(p')\leqslant\alpha SOS(p)$.近似支配概念将在第 3.3.1 节介绍.

假定对图 1 路网图进行一种划分 $T=\{G_1,G_2,G_3\}$,相对应的关联图及 3 个划分子图如图 3 所示.



(a) G_1 (b) G_2 (c) G_3 (d) G'

Fig.3 Subgraphs and correlation graph of the road network

图 3 路网图的子图及关联图

根据划分结果,可以将原路网图视为根节点,将各子图视为叶节点,构建树形索引结构来保存路网信息.假定对路网图进行了图 3 所示的划分,则可以构建图 4 所示的树形索引结构 RN-tree.

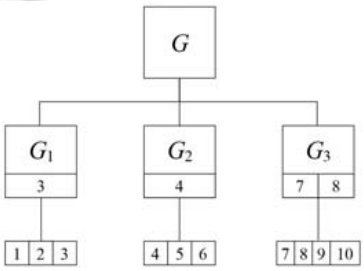


Fig.4 Tree index of road network

图 4 路网图树形索引

通过将路网图压缩成各个子图和关联图,可以很好地减少预处理时空开销.在非叶节点(如图中 G)上,只保存边界点间的权值数组;在叶节点(G_1,G_2,G_3)上,保存所有内部顶点间的权值数组.由于顶点间非支配路径有很多条,所以每个数组单元可能保存多个权值,具体计算方法将在第 3.3.4 节给出,节点信息如图 5 所示.

G_1	1	2	3
1	0	6(3)	5(1)
2	6(3)	0	1(2)
3	5(1)	1(2)	0

G_2	4	5	6
4	0	2(3)	2(1)
5	2(3)	0	1(5) 4(4)
6	2(1)	1(5) 4(4)	0

G_3	7	8	9	10
7	0	1(3)	2(3)	2(2)
8	1(3)	0	3(6)	3(5) 4(4)
9	2(3)	3(6)	0	3(5)
10	2(2)	3(5) 4(4)	3(5)	0

G'	3	4	7	8
3	0	3(2)	1(9) 3(8)	2(5)
4	3(2)	0	4(11) 6(10)	5(7)
7	1(9) 3(8)	4(11) 6(10)	0	1(3)
8	2(5)	5(7)	1(3)	0

Fig.5 Information on tree nodes

图 5 树形结构节点包含信息

3.3 现有加速策略

在路径拓展阶段,本文综合运用了各现有算法对 OSScalling 的加速策略:近似支配剪枝策略、全局优先拓展策略和关键词顶点拓展策略.以下 3 个小节分别介绍这些策略.

3.3.1 近似支配剪枝策略

近似支配剪枝策略在支配剪枝基础上引入一个近似参数,并保证所求解与实际最优解在一定误差范围内.

定义 10(近似支配). 在图 G 下,给定一个略大于 1 的参数 α(假定 α=1.1),若两条起终点相同的路径 p_1,p_2 满足 $BS(p_1)\leqslant BS(p_2)\wedge SOS(p_1)\leqslant \alpha SOS(p_2)$,则称 p_1 近似支配 p_2.

定理 2. 近似支配最优解与目标修正最优解在一定误差范围内:$OS(p_{dom})\leqslant \alpha OS(p_{OS})$[7].

定理 3. 近似支配目标修正最优解与实际最优解误差不超过 $\alpha/(1-\varepsilon)$,即 $OS(p_{dom})\leqslant \alpha/(1-\varepsilon)OS(p_{opt})$.

证明:由定理 2 可知,$OS(p_{dom})\leqslant \alpha OS(p_{OS})$,联立定理 1 可得 $OS(p_{dom})\leqslant \alpha OS(p_{OS})\leqslant \alpha/(1-\varepsilon)OS(p_{opt})$. □

3.3.2 全局优先拓展策略

全局优先拓展策略是在选取拓展标签时考虑局部优先度的基础上,引入全局优先度来保证所求解在一定误差范围内是一个最优解.全局优先度的基础是 BucketBound 思想.

定义 11(BucketBound). 一个存放部分标签的队列称为一个桶(bucket),每个桶对应一个目标值范围.给定一个参数 β(1<β<2),假设第 i 个桶 B_i 对应一个区间 $[\beta^i OS(\tau_{s,t}),\beta^{i+1}OS(\tau_{s,t}))$,$\tau_{s,t}$ 代表起终点间最小目标值路径,目标值落在该范围的路径标签存在这个桶内.不同的桶按照目标值由小到大的顺序排列,每轮拓展从最小目标值的桶内选一个标签拓展至该桶为空,第 1 个可行路径目标值近似最小,此时可近似视为最优解而放弃后续拓展.

定理 4. BucketBound 所求解与实际最优解在同一个桶内,$\beta_{i+1}OS(\tau_{s,t})\leqslant OS(p_{dom})\leqslant OS(p)\leqslant \beta_{i+2}OS(\tau_{s,t})$[4].

BucketBound 策略主要考察拓展是否会使当前路径越来越偏离最小目标路径,偏离最小的可行解即为最优解.这种偏离程度可以表示为全局优先度.

定义 12(全局优先度). 给定参数 β,标签 L_i^k 全局优先度 $gp(L_i^k)=\left\lfloor \log_\beta((L_i^k.OS+OS(\tau_{i,t}))/OS(\tau_{s,t}))\right\rfloor$.

全局优先度代表了路径标签分到哪个桶中,序号越小的桶,存储的路径标签遍历优先级越高.

3.3.3 关键词顶点拓展策略

关键词顶点拓展策略是在初始化时仅维护关键词顶点路径标签列表,在拓展时仅向尚未覆盖的关键词的顶点拓展,可有效减少查询时空开销.关键词顶点间存在多条拓展路径,因此在预处理中采用 SHARK 算法[23]计算出各子图与关联图内所有顶点间的精简 skyline 路径集合.

定义 13(关键词顶点). 对给定查询 $Q=(v_s,v_t,\psi,\Delta)$和 $kw_i\in Q.\psi$.若存在 v_m 满足 $kw_i\in v_m.\psi$,则称 v_m 是关键词 kw_i 的一个关键词顶点.所有对应于 kw_i 的关键词顶点集记为 V_{kw_i},关键词对应顶点的个数 $|V_{kw_i}|$ 称为该关键词顶点稠密度.

定义 14(精简 skyline 路径集合). 以给定两顶点 v_s,v_t 为起终点的路径中,不能被其余路径近似支配的所有路径的总集,符号表示为 $SP(s,t)$.

求出各图精简 skyline 路径集合后,按照目标值从小到大的顺序排序,并在树形索引中相应子节点上维护集合信息.将 v_i,v_j 两顶点间最小 $OS(p)$的路径记为 $\tau_{i,j}$,最小 $BS(p)$(即最大 $OS(p)$)的路径记为 $\sigma_{i,j}$.

3.4 最小代价剪枝策略

由于预处理阶段仅保存了各子图和关联图的精简 skyline 路径集合,在路径拓展阶段调用不同子图顶点间权值时需要实时计算,在此提出不同子图顶点间精简 skyline 路径的计算方式.

假定查询起点 v_s,起点所在子图边界点 $v_s'\in V_s'$,终点 v_t,终点所在子图边界点集为 $v_t'\in V_t'$.任一起终点间路径可划分为 3 段:起点至某一起点所在子图边界点(v_s,v_s')、该起点所在子图边界点至某一终点所在子图边界点(v_s',v_t')、该终点所在子图边界点至终点(v_t,v_t').不同子图顶点间精简 skyline 路径是由相应的 3 个精简 skyline 路径集合中路径组成的不被其他路径支配的路径集合:

$$SP_{cand}(v_s,v_t)=\{p\mid p=(p_1,p_2,p_3)\wedge p_1\in SP(v_s,v'_s),p_2\in SP(v'_s,v'_t),p_3\in SP(v'_t,v_t)\wedge(v'_s,v'_t)\in(V'_s,V'_t)\},$$

$$SP(v_s,v_t)=\{p\mid\forall p'\in SP_{cand}(v_s,v_t),\nexists SOS(p')<SOS(p)\wedge BS(p')<BS(p)\}.$$

定理 5. 不同子图顶点间所有精简 skyline 路径都由起点至起点所在子图边界点集、终点至终点所在子图边界点集、起终点边界点集间的精简 skyline 路径组合构成.

证明:由于起终点不在同一子图中,所以起终点间的路径必定经过起点所在子图边界点集 V'_s 和终点所在子图边界点集 V'_t.不妨假设 $p\notin SP(v_s,v'_s)$,则一定存在一条路径 p',使得 $BS(p')\leqslant BS(p),SOS(p')\leqslant\alpha SOS(p)$.假设 p 和 $SP(v'_s,v_t)$ 中一条路径 p_1 构成最终路径 p_{opt},则一定存在一条由 p' 和 p_1 构成的路径 p'_{opt},满足 $BS(p'_{opt})\leqslant BS(p_{opt})$,$SOS(p'_{opt})\leqslant\alpha SOS(p_{opt})$,即 p_{opt} 被 p'_{opt} 近似支配,则 p_{opt} 不是精简 skyline 路径集合中的路径. □

通过对实际路网图的观察与分析,可以认为关联图中所有顶点间权值往往大于各子图内部的权值,这种现象在大规模路网下尤为明显[24].利用路网图这种特性,可以加快不同子图顶点间精简 skyline 路径集合的计算速度.首先,在预处理过程中寻找到任意两子图 G_i,G_j 所包的含边界点间最小目标值 $OS(\tau_{G_i,G_j})$ 和最小代价值 $BS(\sigma_{G_i,G_j})$,构建子图最小目标矩阵 $O(\tau_G)$和最小代价矩阵 $B(\sigma_G)$,存放在 RN-tree 根节点上.在查询阶段,若计算的是 $v_i,v_j(v_i\in G_i,v_j\in G_j)$间精简 skyline 路径集合,首先找出 $\tau_{G_i,G_j}(\sigma_{G_i,G_j})$,在此基础上,沿着最小目标(代价)路径双向拓展至 v_i 和 v_j,所得路径记为 $\tau'_{i,j}(\sigma'_{i,j}).\tau'_{i,j}(\sigma'_{i,j})$ 在大多数情况下就是实际最小目标(代价)路径 $\tau_{i,j}(\sigma_{i,j})$.依据这个结论,可以利用近似支配策略剪去两子图间大量无效路径和通过这些路径生成的最终路径,从而极大加速了计算过程.

为解决关键词顶点稠密度增加导致大量不符合代价阈值的顶点被拓展到的问题,本文在上述对路网图观察和分析的基础上,提出最小代价剪枝策略.在每轮拓展前计算剩余代价,与 $B(\sigma_G)$对比后得到当前子图所对应的所有有效待拓展子图.在拓展时,若待拓展顶点不属于这些子图对应的节点,则不予拓展.

3.5 算法详述

综合上文,第 3.5.1 节给出 KORL 算法步骤和运行示例,第 3.5.2 节对 KORL 时间、空间复杂度和近似度作出分析.

3.5.1 算法步骤

算法 1. KORL.

Input:RN-tree,$Q=(v_s,v_t,\psi,\Delta),\beta$.

Output:L_{opt}.

1: set B_0=*null*, L_{opt}=*null*, *Found*=false, get V_{kw} where $kw\in Q.\psi-v_t.\psi$

2: create $L_s^0=(v_s.\psi\cap Q.\psi,0,0,0)$ at v_s, enqueue L_s^0 on B_0, get $\tau_{s,t}$=argmin$_{p\in SP(s,t)}OS(p)$

3: **while** *Found*=false **do**

4: get the first non-empty queue B_r

5: **if** all queues are empty **then**

6: **return** no feasible route

7: dequeue L_i^k which has the highest local priority

8: **if** L_i^k covers all keywords **then**

9: **for each** $sp\in SP(i,t)$ **do**

10: $L_t^l=(L_i^k.\psi\cup v_t.\psi,L_i^k.SOS+SOS(sp),L_i^k.OS+OS(sp),L_i^k.BS+BS(sp))$

11: **if** $L_t^l.BS\leqslant\Delta$ **then**

12: enqueue L_t^l on B_{gp} where $gp=\left\lfloor\log_\beta(L_t^l.OS/OS(\tau_{s,t}))\right\rfloor$

13: **else** delete L_t^l

14: **if** B_{gp} doesn't exists **then**

15: create B_{gp} and enqueue L_t^l on B_{gp}
16: **if** B_{gp}=B_r **then**
17: *Found*=true, $L_{opt}=L_t^l$
18: **else**
19: **for each** $v_j \in V_{kw}$, $kw \in (Q.\psi - v_t.\psi - L_i^k.\psi) \wedge v_i \in G_i, v_j \in G_j, BS(\sigma_{G_i,G_j}) \leqslant \Delta - L_i^k.BS$ **do**
20: **for each** $sp \in SP(i,j)$ do
21: create $L_j^l = (L_i^k.\psi \cup v_j.\psi, L_i^k.SOS + SOS(sp), L_i^k.OS + OS(sp), L_i^k.BS + BS(sp))$ on v_j
22: **if** L_j^l isn't dominated by other labels on v_j and $L_j^l.BS + BS(\sigma_{j,t}) \leqslant \Delta$ **then**
23: enqueue L_j^l on B_{gp} where $gp = \lfloor \log_\beta((L_j^l.OS + OS(\tau_{j,t}))/OS(\tau_{s,t})) \rfloor$
24: **else delete** L_j^l
25: **if** B_{gp} doesn't exists **then**
26: create B_{gp} and enqueue L_j^l on B_{gp}
27: **for each** L on v_j **do**
28: **if** L is dominated by L_j^l **then**
29: **delete** L

算法第 1 行、第 2 行给出了各变量初始化结果,计算出算法中经常使用的一些定量.第 3 行~第 6 行表示从第 1 个非空队列中取出局部优先度最高的标签.若所有队列为空,表示无可行解.第 7 行~第 17 行表示路径标签覆盖所有关键词后如何拓展至终点并近似求得最优路径,其中,第 7 行~第 10 行代表沿着标签所在顶点到终点的不同精简 skyline 路径拓展新标签;第 11 行~第 15 行表示若新标签满足剪枝条件,则将其放入对应队列;第 16 行、第 17 行表示若新标签所入队列与当前队列一致,则近似输出最优结果.第 18 行~第 29 行介绍路径标签未覆盖全部关键词时的相应操作,其中,第 18 行~第 21 行表示从拓展时满足剪枝条件的子图中,取出尚未拓展的关键词对应的顶点,沿着不同的精简 skyline 路径拓展,创建新标签;第 22 行~第 26 行判别新标签是否满足剪枝条件,满足则将其放入对应队列;第 27 行~第 29 行利用新标签剪裁标签所处顶点中原有标签.

算法运行示例:以图 1 路网图、图 2 关键词倒排列表、图 3 的划分为例.查询 Q=(v_1,v_{10},〈kw_1,kw_2〉,11),修正参数ε=0.5,修正相当于将目标值放大 22 倍,近似支配参数α=1.1,全局优先度参数β=1.1.

经计算$\tau_{s,t}$=(v_1,v_3,v_7,v_{10}),初始化队列 B_0,起点处初始化路径标签 $L_1^1=(\phi,0,0,0)$ 并放入 B_0,从 B_0 中取出 L_1^1,向未覆盖关键词、并不在距起点所在子图目标值代价值超标的子图中的关键词顶点拓展,待拓展顶点有 v_3,v_7,v_8,v_9.对每一个顶点进行拓展过程中,沿着不同的精简 skyline 路径,得到的新的标签: $L_3^1=(\langle kw_1\rangle,110,5,1), L_7^1=(\langle kw_1\rangle, 132,6,10), L_7^2=(\langle kw_2\rangle,176,8,9), L_8^1=(\langle kw_2\rangle,154,7,6), L_9^1=(\langle kw_2\rangle,176,8,13), L_9^2=(\langle kw_2\rangle,220,10,12)$.在各自顶点存储这些路径标签,并筛去不满足剪枝条件的标签 L_7^1, L_9^1, L_9^2 .将剩余标签分别放入对应队列, L_3^1 放入 B_0, L_8^1 放入 B_2, L_7^2 放入 B_2.第 1 轮迭代并无满足所有条件路径.

从 B_0 中按局部优先度取出(B_0 目前只包含一个标签) L_3^1,按照拓展规则向 v_8,v_9 拓展.得到新标签 $L_8^2=(\langle kw_1, kw_2\rangle,154,7,6), L_9^3=(\langle kw_1,kw_2\rangle,176,8,13), L_9^4=(\langle kw_1,kw_2\rangle,220,10,12)$.抛弃被剪枝的 L_9^3 和 L_9^4 .将 L_8^2 放入 B_2.因为当前处于 B_0 而非 B_1 队列,故即使 L_8^2 已满足其余条件,仍不能结束遍历.

由于 B_0 中已无其他标签,故从 B_2 中取出局部优先级最高的标签 L_8^2,对其进行拓展.由于所有关键词已包含,故直接向终点拓展,产生两个新标签 $L_{10}^1=(\langle kw_1,kw_2,kw_3\rangle,220,10,11), L_{10}^2=(\langle kw_1,kw_2,kw_3\rangle,242,11,10)$.$L_{10}^1$ 对应 B_2, L_{10}^2 对应 B_3.因为 L_{10}^1 所处队列与当前队列一致,故判定 L_{10}^1 为最优路径标签.

3.5.2 算法分析

- 时间复杂度

由于 KORL 算法是一种近似算法,查询时间存在着很大的随机性,所以需按照算法最坏情况来计算时间复杂度,即:忽略掉全局优先策略,按照无剪枝效果的 OSScalling 算法来计算时间复杂度.

根据第 3.4 节的描述,顶点间所有精简 skyline 路径是由被分割的 3 段 skyline 路径组合并剪枝所得.在子图或关联图中,顶点间精简 skyline 路径集合最大元素个数$|sp|_{\max}=\log_{\alpha}(|V|o_{\max}/o_{\min})$[13].起(终)点所在子图平均有$U|V'|/|V|$个边界点,其中,$U$ 代表图划分时规定每个子图的最大顶点数.故任意顶点间精简 skyline 路径数量为$|sp|_{\max}^{3}\cdot(U^2|V'|^2/|V|^2+2U|V'|/|V|)$,记为$|SP|$.

根据定义 8,每个关键词顶点最多拥有$2^{\lfloor|\psi|\Delta/b_{\min}\rfloor}o_{\max}\Delta/\varepsilon b_{\min}o_{\min}\rfloor$个路径标签,记为 $L_{\max}$.若最大关键词稠密度为 $V_{\max}$,则待拓展关键词顶点总数$|V_{kw}|=\sum_{kw_i\in Q.\psi}|V_{kw_i}|\leqslant|\psi|V_{\max}$.因此最多有$|\psi|V_{\max}L_{\max}$ 个标签待拓展,每次从队列中取一个标签并进行$|SP||\psi|V_{\max}$ 次拓展.

最坏情况下,向队列中取标签的复杂度为 $O(\lg(|\psi|V_{\max}L_{\max}))$,判别新标签与顶点中其他标签是否支配复杂度为 $O(L_{\max})$,故总时间复杂度为$O(|SP||\psi|^2V_{\max}^2L_{\max}\lg(|\psi|V_{\max}L_{\max})+|SP||\psi|^2V_{\max}^2L_{\max}^2)$.

- 空间复杂度

KORL 算法空间消耗主要是顶点列表和队列中存储的路径标签,由于最坏情况下需要遍历所有关键词顶点,加上起终点会产生路径标签,队列最多存储$|\psi|V_{\max}L_{\max}$ 个路径标签,所以空间复杂度为 $O((2|\psi|V_{\max}+2)\cdot L_{\max})$.

- 近似度分析

假定最终解为 p,OSScalling 算法(只采用近似支配和目标修正策略)所求解为 p_{dom},实际最优解为 p_{opt}.若 P 对应队列为 B_i,根据定理 4 可知,$OS(p_{dom})\geqslant\beta^iOS(\tau_{s,t})$,$OS(p)\leqslant\beta^{i+1}OS(\tau_{s,t})$,结合定理 3,可得:

$$\frac{OS(p)}{OS(p_{opt})}=\frac{OS(p)}{OS(p_{dom})}\frac{OS(p_{dom})}{OS(p_{opt})}<\frac{\alpha\beta^{i+1}OS(\tau_{s,t})}{(1-\varepsilon)\beta^iOS(\tau_{s,t})}=\frac{\alpha\beta}{1-\varepsilon}.$$

故最终解近似度为 $\frac{\alpha\beta}{1-\varepsilon}$.

4 实验分析与验证

本节通过实验对 KORL 算法进行相关验证与分析.第 4.1 节介绍实验数据集与环境.第 4.2 节介绍实验的设计思路.第 4.3 节对比 KORL 算法与现有算法预处理阶段的性能表现.第 4.4 节分析 KORL 算法在不同查询因素下的查询效率.

4.1 数据集与环境

本文所有实验运行在 Intel(R) Xeon(R) CPU E5-2609 v4@1.70GHz×8 的服务器上,内存大小为 16G.所有算法均在 Java 上实现.

本文实验采用 9th DIMACS Implementation Challenge[11]所提供的路网图数据,路网图边集具有路径长度和路径耗时两个属性.本实验采用不同规模的 4 个查询图,其中,G_1~G_3 是通过程序从纽约市路网图中截取 3 个子图,将路径长度视为 KOR 中的代价值,路径耗时视为目标值,并采用 1 000 个关键词随机为查询图中各顶点生成关键词描述[7];G_4 是在完整的纽约市路网图上采用 10 000 个关键词随机生成的查询图.数据集具体信息见表 1.

Table 1 Query graph

表 1 查询图

查询图	顶点数	边数	顶点平均关键词数	查询图	顶点数	边数	顶点平均关键词数
G_1	10K	22K	4.3	G_3	100K	266K	3.7
G_2	25K	63K	4.4	G_4	264K	730K	8.2

4.2 实验设计

为验证 KORL 算法非常适用于大规模路网图,需要分别验证其在预处理和查询阶段的性能.为了验证 KORL 算法在预处理阶段具有良好的可扩展性,我们设置了实验 1.由于现有算法无法运行在大规模查询图上,因此我们仅验证不同查询因素对算法效率的影响.针对 3 个查询因素:查询图规模、关键词个数和代价阈值,我们设置了实验 2~实验 4,并且为了验证 KORL 算法在查询阶段拥有不错的表现,我们在实验 2 中,现有算法可运行的查询图上加入与现有算法的对比.

- 实验 1:预处理时空开销实验

为了验证 KORL 算法在预处理表现上的优越性,需要对比 KORL 算法与现有算法在预处理上的时空开销.由于现有算法预处理方法大致相同,因此选取最近提出的算法 KSRG[7]作为对比.实验运行在本机环境(16G 内存)下,采用 G_1~G_4 作为查询图,评价标准是两种算法在预处理阶段的内存占用空间和预处理计算时间.

- 实验 2:查询图规模影响实验

为了判断查询图规模对算法效率的影响,需要控制关键词个数与代价阈值相同的前提下改变查询图规模,评价标准是平均查询时间.实验设定关键词个数为 6,代价阈值为 60km,分别采用 G_1~G_4 作为查询图.为了验证 KORL 算法具有不错的查询表现,需要和现有算法作对比.实验采用 OSScalling[4],KSRG[7]和 KORL 算法,随机提交 10 个查询并对查询时间取平均.

- 实验 3:关键词个数影响实验

为了判断关键词个数对算法效率的影响,需要控制数据集与代价阈值相同的前提下改变关键词个数,评价标准是平均查询时间.实验采用 G_2~G_4 作为查询图,代价阈值设定为 60km,关键词个数分别取 2,4,6,8.实验采用 KORL 算法,随机提交 10 个查询并对查询时间取平均.

- 实验 4:代价阈值影响实验

为了判断代价阈值对算法效率的影响,需要控制数据集与关键词个数相同的前提下改变查询代价阈值,评价标准是平均查询时间.实验采用 G_2~G_4 作为查询图,关键词个数设定为 6,代价阈值分别取 20,40,60,80km.实验采用 KORL 算法,随机提交 10 个查询并对查询时间取平均.

4.3 预处理开销对比

实验 1 的预处理空间开销如图 6 所示,时间开销如图 7 所示.

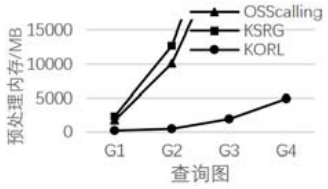


Fig.6 Preprocessing space overhead

图 6 预处理空间开销

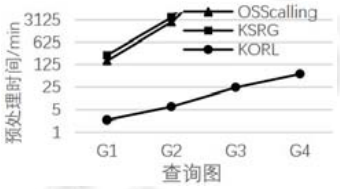


Fig.7 Preprocessing time overhead

图 7 预处理时间开销

从图 6 中可以看到,当查询图为 G_2(顶点数为 25 000)时,KSRG 算法内存开销已经接近 16GB.对于 G_3,G_4 而言,内存开销已远远超出 16GB,故不在图中标明.而 KORL 算法处理 G_4 仅需 5GB 左右的内存,对比可见,KORL 算法预处理空间开销远小于以往算法.

在时间开销方面,KSRG 算法对 G_2 的预处理已经超过 50h,后续查询图因时间太长不再标明.而 KORL 算法对 G_4 的预处理仅需 1h.由此体现出 KORL 算法预处理在大规模路网图下性能远超以往.

4.4 不同查询因素对算法效率的影响

本小节分别给出实验 2~实验 4 的结果并进行分析.

4.4.1 查询图规模对算法效率的影响

实验 2 结果如图 8 所示.从图 8 中可以看出,当查询图规模扩大时,查询时间增长会变缓.这是因为查询图规模越大,代价阈值的剪枝能力就相对越强.若在很小的图中,最优路径的代价值还未达到代价阈值,需要遍历整个解空间才能得到最优解;而当图的规模越大时,越多的顶点会被最小代价剪枝策略筛去,增长速度也会放缓.这种放缓趋势会延续下去,直到查询图大到很多顶点间的代价值已超过代价阈值时,此时在第 1 轮拓展中就会把无关顶点筛去,导致查询时间基本不会变化.

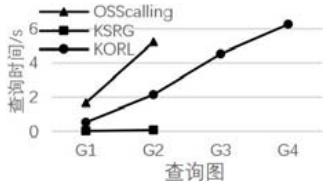


Fig.8 Querying time on different scales of query graph

图 8 不同查询图规模下的查询时间

在本实验中,同样观察了两种以往算法在大规模路网图上的表现.由于以往算法对 G_3,G_4 的预处理内存开销超过本实验环境,故仅标出其在 G_1,G_2 上的查询时间.对比 3 个算法在查询阶段的表现,KORL 算法查询时间介于 OSScalling 与 KSRG 算法之间.虽然 KORL 算法需要在路径拓展中动态计算拓展路径,但由于多种加速策略的使用,导致其查询时间并未显著增加.通过预处理阶段和查询阶段的良好表现,验证了 KORL 算法良好的可扩展性.

4.4.2 关键词个数对算法效率影响

实验 3 的结果如图 9 所示.从图 9 中 G_2~G_4 这 3 个查询图上查询时间随关键词个数别的变化可以看出,当关键词达到 6 个以上时,查询时间会显著上升.这是由于关键词增多时,遍历的深度也会大幅上升,即使采用了关键词顶点拓展策略,也会产生大量中间结果,导致查询时间的显著上升.

当图的规模增大时,查询时间随关键词个数增加而增加的速度会上升.这是因为在更大规模的图中,关键词个数增加会导致最优路径更难被拓展出来,BucketBound 算法筛去后续拓展的效率变慢.因此在越大的图中,关键词个数的增加会造成查询时间显著增加.

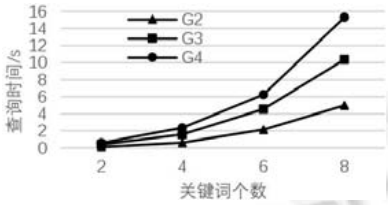


Fig.9 Querying time on different numbers of keywords

图 9 不同关键词个数下的查询时间

4.4.3 代价阈值对算法效率影响

实验 4 结果如图 10 所示.从图 10 中 G_2~G_4 这 3 条曲线的共同变化趋势可以看出,查询时间呈先增长后平稳的趋势.这是由于在代价阈值很小的时候,很多待拓展顶点会被最小代价剪枝手段筛去;当代价阈值增大到实际最小目标路径的代价值时,BucketBound 近似策略会使 KORL 算法尽快地返回近似最优解,不再进行后续拓展;当代价阈值继续增大时,实际最优解不会发生改变,故查询时间不再有显著变化.

当图的规模增大时,查询时间随代价阈值增加而增加的速度会上升.因为在越大规模的查询图中,相对越小的代价阈值对查询算法的剪枝能力就越弱,查询时间增长的增长速度就会上升.

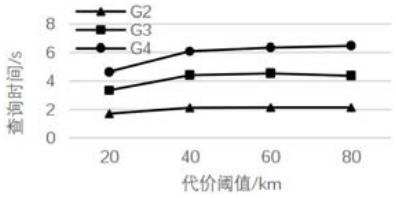


Fig.10 Querying time on different cost threshold

图 10 不同代价阈值下的查询时间

5 总结与展望

本文针对以往 KOR 算法预处理无法适用于大规模路网的问题,提出一种利用划分路网和构建树形索引的预处理方法来组织路网信息,在此基础上提出相应的查询算法 KORL 并做出改进.通过大规模查询图上的实验证明 KORL 算法良好的可扩展性.对 KOR 查询的优化一方面要继续寻找更好的剪枝策略,另一方面探索利用执行过的 KOR 查询的子路径也是一个重要方向.

References:

[1] Gao YJ, Qin X, Zheng BH. Efficient reverse top-*k* Boolean spatial keyword queries on road networks. IEEE Trans. on Knowledge and Data Engineering, 2015,27(5):1205−1218. [doi: 10.1109/TKDE.2014.2365820]

[2] Cao X, Cong G, Jensen CS, Beng CO. Collective spatial keyword querying. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011). 2011. 373−384. [doi: 10.1145/1989323.1989363]

[3] Choudhury FM, Culpepper JS, Sellis T, Cao X. Maximizing bichromatic reverse spatial and textual *k* nearest neighbor queries. Proc. of the VLDB Endowment, 2016,9(6):456−467. [doi: 10.14778/2904121.2904122]

[4] Cao X, Chen LS, Cong G, Xiao XK. Keyword-aware optimal route search. Proc. of the VLDB Endowment, 2012,5(11):1136−1147. [doi: 10.14778/2350229.2350234]

[5] Cao X, Chen LS, Cong G, Guan JH, Phan NT, Xiao XK. KORS: Keyword-aware optimal route search system. In: Proc. of the ICDE 2013. 2013. 1340−1343. [doi: 10.1109/ICDE.2013.6544939]

[6] Jin PF, Niu BN, Zhang XZ. KSRG: An efficient optimal route query algorithm for multi-keyword coverage. Journal of Computer Applications, 2017,37(2):352−359 (in Chinese with English abstract). [doi: 10.11772/j.issn.1001-9081.2017.02.0352]

[7] Jin PF. Research on efficient keyword-aware optimal route query processing method [MS. Thesis]. Jinzhong: Taiyuan University of Technology, 2017 (in Chinese with English abstract).

[8] Floyd RW. Algorithm 97: Shortest path. Communications of the ACM, 1962,5(6):345−346. [doi: 10.1145/367766.368168]

[9] Xiao XY, Luo Q, Li ZS, Xie X, Ma WY. A large-scale study on map search logs. ACM Trans. on the Web (TWEB), 2010,4(3):8–9. [doi: 10.1145/1806916.1806917]

[10] Zhong RC, Li GL, Tan KL, Zhou LZ, Gong ZG. G-tree: An efficient and scalable index for spatial search on road networks. IEEE Trans. on Knowledge and Data Engineering, 2015,27(8):2175−2189. [doi: 10.1109/TKDE.2015.2399306]

[11] 2018. http://www.dis.uniroma1.it/challenge9/index.shtml

[12] Delling D, Goldberg AV, Razenshteyn I, Werneck RF. Graph partitioning with natural cuts. In: Proc. of the IEEE Int'l Conf. on Parallel and Distributed Processing Symp. (IPDPS). 2011. 1135−1146. [doi: 10.1109/IPDPS.2011.108]

[13] Li RH, Qin L, Yu JX, Mao R. Optimal multi-meeting-point route search. IEEE Trans. on Knowledge and Data Engineering, 2016, 28(3):770−784. [doi: 10.1109/TKDE.2015.2492554]

[14] Zhang JZ, Wen J, Meng XF. Multi-tag route query based on order constraints in road networks. Chinese Journal of Computers, 2012,35(11):2317−2326 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2012.02317]

[15] Bao JL, Yang XC, Wang B, Wang JY. An efficient trip planning algorithm under constraints. Journal of Chinese Computer Systems, 2013,34(12):429−434 (in Chinese with English abstract). [doi: 10.1109/WISA.2013.87]

[16] Bao JL, Wang B, Yan SC, Yang XC. Multi-constrained optimal path search algorithms. In: Proc. of the 16th Asia-Pacific Web Conf. Springer-Verlag, 2014. 355−366. [doi: 10.1007/978-3-319-11116-2_31]

[17] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 1998,20(1):359−392. [doi: 10.1137/S1064827595287997]

[18] Meyerhenke H, Monien B, Sauerwald T. A new diffusion-based multilevel algorithm for computing graph partitions. Journal of Parallel and Distributed Computing, 2009,69(9):750−761. [doi: 10.1016/j.jpdc.2009.04.005]

[19] Tsaggouris G, Zaroliagis C. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. Theory of Computing Systems, 2009,45(1):162−186. [doi: 10.1007/s00224-007-9096-4]

[20] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 3rd ed. London: MIT Press, 2009. 1048−1128. [doi: 10.1002/9783527649846.ch17]

[21] Dumitrescu I, Boland N. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. Networks, 2003,42(3):135−153. [doi: 10.1002/net.10090]

[22] Wang SB, Xiao XK, Yang Y, Lin WQ. Effective indexing for approximate constrained shortest path queries on large road networks. Proc. of the VLDB Endowment, 2016,10(2):61−72. [doi: 10.14778/3015274.3015277]

[23] Delling D, Wagner D. Pareto paths with SHARC. In: Proc. of the Int'l Symp. on Experimental Algorithms. Springer, 2009. 125−136. [doi: 10.1007/978-3-642-02011-7_13]

[24] Abeywickrama T, Cheema MA, Taniar D. *K*-nearest neighbors on road networks: A journey in experimentation and in-memory implementation. Proc. of the VLDB Endowment, 2016,9(6):492−503. [doi: 10.14778/2904121.2904125]

附中文参考文献:

[6] 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.计算机应用,2017,37(2):352−359. [doi: 10.11772/j.issn.1001-9081.2017.02.0352]

[7] 金鹏飞.基于关键词的最优路径查询高效处理方法的研究[硕士学位论文].晋中:太原理工大学,2017.

[14] 张金增,文洁,孟小峰.路网环境下访问序列受限的多标签路线查询算法.计算机学报,2012,35(11):2317−2326. [doi: 10.3724/SP.J.1016.2012.02317]

[15] 鲍金玲,杨晓春,王斌,王佳英.一种支持约束关系的高效的行程规划算法.小型微型计算机系统,2013,34(12):2702−2707. [doi: 10.1109/WISA.2013.87]

郝晋瑶(1993－),男,硕士,主要研究领域为空间数据查询.

康家兴(1994－),男,硕士,主要研究领域为空间数据库,大数据分析.

牛保宁(1964－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据管理与分析,数据库系统性能管理,区块链数据管理.