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.

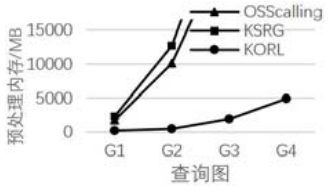

Fig.6 Preprocessing space overhead

图 6 预处理空间开销

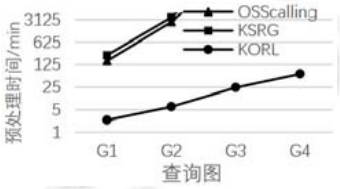

Fig.7 Preprocessing time overhead

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.

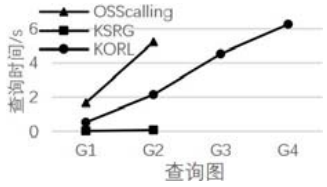

Fig.8 Querying time on different scales of query graph

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.

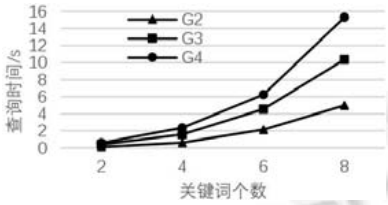

Fig.9 Querying time on different numbers of keywords

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.

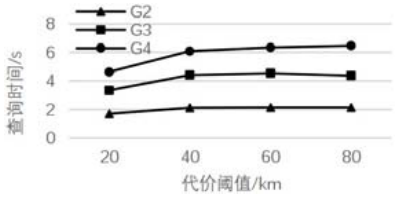

Fig.10 Querying time on different cost threshold

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.