• 计算机工程与应用 •

时间依赖图下的最小费用路径搜索

马 慧[1]，汤 庸[2*]，傅 瑜[1]，易 锋[1]

(1. 电子科技大学中山学院 广东 中山 528402；2. 华南师范大学计算机学院 广州 510631)

【摘要】该文提出了一种时间依赖图下最小费用路径的高效搜索算法。已有的算法从起点开始向四周扩展以发现到达终点的路径，搜索空间较大，查询耗时。本文从以下两方面减少搜索空间：首先缩小顶点的有效时间区间避免无用的计算，并且在顶点的相应的时间区间的最小费用正确计算出来之后，再计算扩展路径的费用；然后提出一种双向搜索方法，从起点和终点同时出发向四周扩展路径直到两个搜索相遇，从而控制搜索空间在以起点、终点为圆心的两个小圆内。针对路径的时变依赖性设计了双向搜索的停止条件和路径生成方法，理论上证明了方法的正确性。最后，在大规模数据集上试验证了方法的有效性。

关 键 词 双向搜索； 分段常量函数； 最短路径； 时间依赖费用

中图分类号 TP311 文献标志码 A doi:10.12178/1001-0548.2019002

Finding the Minimal Cost Path in Time-Dependent Graphs

MA Hui[1], TANG Yong[2*], FU Yu[1], and YI Feng[1]

(1. Zhongshan Institute, University of Electronic Science and Technology of China Zhongshan Guangdong 528402;
2. School of Computer Science , South Normal University Guangzhou 510631)

Abstract This paper proposes an efficient search to find a minimal cost path in a time-dependent graph. Current method discovers paths to the destination by expanding discovered paths originated from the source vertex. However, this blind search incurs large searching space, which lowers the query efficiency. The searching space is reduced in the following two ways. Firstly, during the search, the computation is confined within the valid time interval of each vertex to avoid useless computations. Besides, the costs of new expanded paths is computed not until the minimal cost of a path within a time interval of a vertex is correctly computed. Secondly, two searches are respectively invoked from the source vertex and the destination vertex, by which the searching space is roughly located in two small circles centered at the source and the destination. The bi-directional search stopping criteria as well as the way to regenerate the minimal cost path are discussed. The correctness of the proposed method is also theoretically proved. Finally, extensive experiments on large scale datasets confirm the efficiency of the proposed method.

Key words bidirectional search; piecewise constant function; shortest path; time-dependent cost

最优路径查询有着广泛的应用。近年来，学者们提出了多种最优路径查询算法。针对不同的应用场景，“最优路径”的查询目标也不一样。例如用户希望找到“距离最短”[1-3]、“最快到达”[4-9]或最条路径[10-12]等。在一些应用中，用户希望在道路上所花的费用越少越好。例如物流公司车在道路上行驶时间越长，耗油量就越大，需要找到一条耗油量最少的路径[13-14]。

上述应用可以用一个时间依赖图来刻画路网，图模型上的每条边附带一个函数 $f_{i\to j}(t)$，表示通过边 $\langle v_i,v_j\rangle$ 的费用随出发时刻 t 而变化。给定起点 v_s 和终点 v_d，查询返回路径 P，使得 P 的费用最小。允许在路径的顶点上停留，以待在合适的时间继续前行[13-20]。本问题的难点在于所研究的路径的费用具有时间依赖性。即从起点 v_s 到顶点 v_i 的最小费用不是一个常数，而是一个以时间为自变量的函数，表示最小费用依赖于时间变化。

在路网中，每一个顶点都只能通过它的入边邻

收稿日期：2019－01－21；修回日期：2020－01－09
基金项目：国家自然科学基金 (U 0009, 61772211)；广东省优秀青年教师基金 (YQ2015241)
作者简介：马慧 (1981－)，女， 副教授，主要从事数据库理论、大规模图查询方面的研究.
通信作者：汤庸 ail：ytang@m.scnu.edu.cn

居到达。若能计算出到达 v_j 的所有入边邻居 v_i 的最小费用，便可计算出到达 v_j 的最小费用。这个求值过程与最短路径的经典算法 Dijkstra算法[1] 类似。因此，文献 [13] 采用了类 Dijkstra 算法求解：从起点开始向四周扩散，按路径费用升序扩展新路径。该方法与 Dijkstra 算法的区别于：传统下，每个顶点 v_i 在 Dijkstra 搜索中仅被扩展一次，因为起点 v_s 到 v_i 的最短路距离只有一个值，本问题中每个顶点 v_i 会多次被拿出来扩展，因为对一顶点的到达时间不同，费用也不同。设 $c_\perp$ 表示满足查询条件的最小费用，所有满足条件 $g_i(t)<c_\perp$ 的路径都会被用来扩展新路径，其中 $g_i(t)$ 表示从起点 v_s 出在 t 时刻位于顶小费用。

如果把搜索过程中访问过的顶点在表示路网的图上作标记的话，搜索空间大致分布在以 v_s 为圆心、$c_\perp$ 为半径的圆内。假设平均每个顶点被扩展 k 次，搜索量大用 k 以 $c_\perp$ 为半径的圆的面积来衡量。为了减少搜索量，可分别从和终点 v_d 出发启动正向搜索和逆向搜索，两个搜索交替进行，直到“相遇”为止。“相遇”指找到一个中间点 v_i，使得正向搜索已求出从 v_s 到的最小费用路径，逆向搜索也求出到 v_d 的最小费用路径，便从已搜索的路径中生成从 v_s 到 v_d 的最小路径。需说明的是，费用路径不一定路过 v_i，细节第 4 节中讨论。直观上，搜索的搜索空间大致位于分别 v_s、v_d 为圆心，半径为 $c_\perp/2$ 的两个圆上。假设平均每个顶点被扩展了 k 次，搜索空间大致等于两个半径为 $c_\perp/2$ 的圆的面积之和的 k 倍，较单向搜索的搜大大减少。

本文优化了文献 [13-14] 的算法，并在其基础上提出了双向搜索算法。虽然双向搜索是一种常见的优化算法，但路径费用与时间相关，双向搜索算法的难点在于：路径用的时变依赖性给双向搜索的处理细节和终止增加了复杂性。

1　相关工作

最优路径查询的一个经典问题是静态图上的最短路径查询。的方法在大规模路网下，一个查询平均数毫秒的时间便可完成[2]。但实际应用中路网具有动态性，边的权值依赖于时间变化。一类时间依赖图基于离散模型，一条边的权值依赖于某些离散的出发时间点[4-6]。另一类时间依赖基于连续模型，即每一条边的权值是一个随时间变化的连续函数[7-8]。由于公共交通网路网有着天然的相似性有很多工作研究公共交通网络下的最优路径规划问题[9]。上述时间依赖图下的最优路径查询研共同点是计算“总耗时最少的路径”，即达减去出发时刻衡量路径的耗时。总时最少的路径不一定是路径上途经边用之和最小的, 因此这些方法不能解决文研究的问题。

本文的研究基于文献 [13] 提出型：图中每条边 $\langle v_i,v_j\rangle$ 附有一个值 $w_{i\to j}$，表示通过 $\langle v_i,v_j\rangle$ 的耗时；另有一个费用函数 $f_{i\to j}(t)$，表示通过 $\langle v_i,v_j\rangle$ 的费用依赖于顶点 v_i 的出刻 t，路径允许在顶点上等待，以实现费用最小。在此模型的基础文献 [14] 提出了一种两阶段搜索方法 Two-Step，将文献 [13] 中的每条边的 $w_{i\to j}$ 扩展成时间依赖函数，并对文献 [13] 的方法做了改进，包括 $w_{i\to j}$ 是依赖函数的场景。文献 [15] 提出的问题模型与文献 [13-14] 相似，每条边 $\langle v_i,v_j\rangle$ 只有一个函数 $f_{i\to j}(t)$，$f_{i\to j}(t)$ 表示在 t 时刻通过 $\langle v_i,v_j\rangle$ 的耗时，求解路上耗时 (on-road travel time) 最小的路径，允许路径在顶点上等待。文献 [16] 在文献 [15] 础上采辆的历史轨迹数据来构造时间依赖图，并提出了一种求近似解的算法，加快了查询速度。另有工作研究基散时间模型的允许在顶点上等待的最小费用路径查询[18, 20]。

2　问题描述

本文沿用文献 [13] 的时间依赖图模型和最小费用路径查询问题的定义。

定义 1　时间依赖图。时间依赖图是一个有向图，记作 $G_T=(V, E, W, F)$。$V=\{v_i\}$ 是顶点的集合，$E\subseteq V\times V$ 是边的集合。$\langle v_i,v_j\rangle\in E$ 表示一条从顶点 v_i 到顶点 v_j 的有向边。每条边 $\langle v_i,v_j\rangle$ 附有一个权值 $w_{i\to j}\in W$，表示通过 $\langle v_i,v_j\rangle$ 的耗时；另有函数 $f_{i\to j}(t)\in F$ 表示在 t 时刻通过 $\langle v_i,v_j\rangle$ 的费用。$f_{i\to j}(t)$ 段常量函数表示：

$$f_{i\to j}(t)=\begin{cases}c_1 & t_0\leqslant t<t_1\\ c_2 & t_1\leqslant t<t_2\\ \cdots & \\ c_k & t_{k-1}\leqslant t<t_k\end{cases}$$

为了表述清晰，假设通过 $\langle v_i,v_j\rangle$ 的时间 $w_{i\to j}$ 是一个非负常数。需说明的是，本文的算法也适用于当 $w_{i\to j}$ 是一个关于 t 的函数的情形，只要 $w_{i\to j}(t)$ 满足 FIFO 性质：$t_1<t_2\leftrightarrow t_1+w_{i\to j}(t_1)<t_2+w_{i\to j}(t_2)$，具体改动方法参见文献 [14]。

同文献 [13-14] 一样，本文假设 $\forall f\in F$ 的所有分段区间均为左闭右开区间。当任意一端改成开区间或闭区间时，本文讨论的算法依然有效。

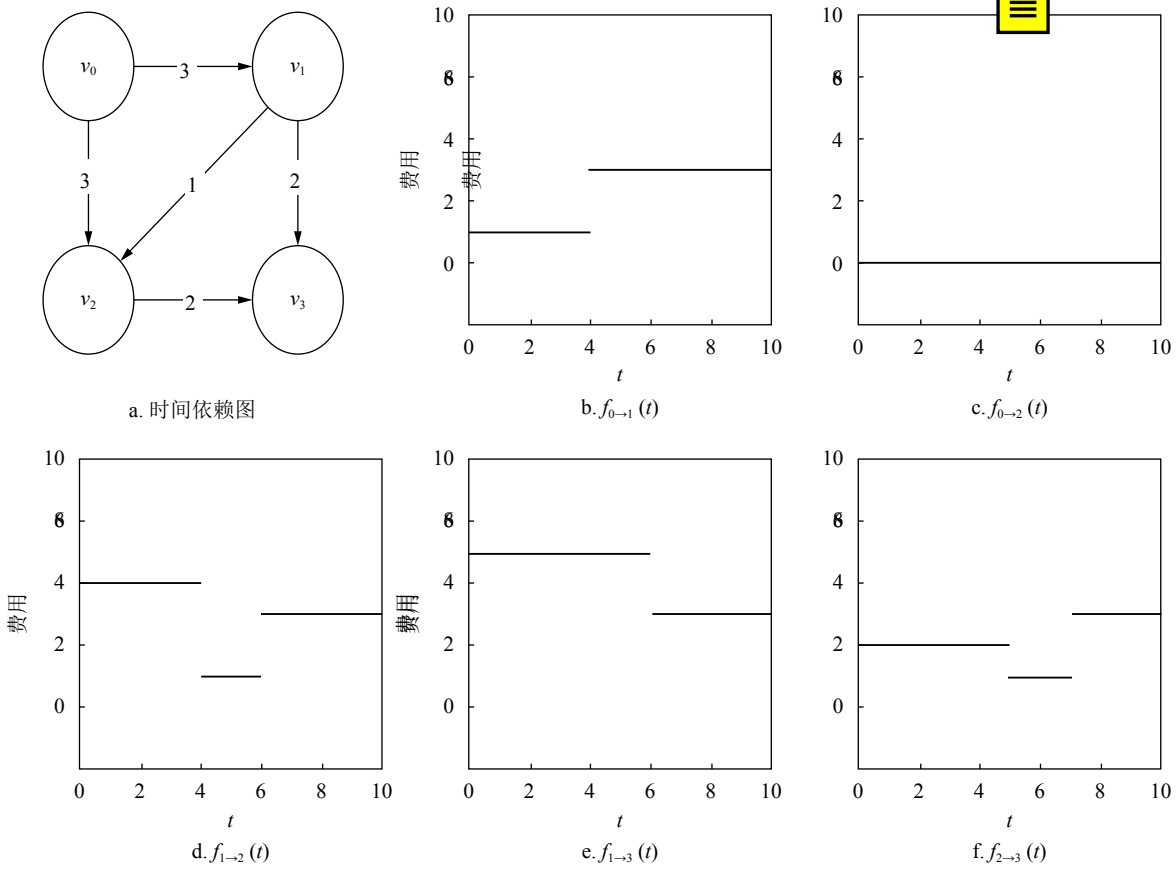
定义 2　路径 $P=v_1@t_1\to v_2@t_2\to\ldots\to v_k@t_k\to v_{k+1}$

是图 G_T 的一条路径，其中 $\langle v_i,v_{i+1}\rangle \in E(1\leqslant i\leqslant k)$。$P$ 表示在 t_1 时刻从顶点 v_1 出发，在 $t_1+w_{1\to 2}$ 时刻到达 v_2，然后在 t_2 时刻从 v_2 出发，…，到达 v_k 后在 t_k 时刻出发，最后到达 v_{k+1}。路径上允许到达顶点 v_i 后，等待一段时间再出发，以期待 P 的费用更小。P 上的时间约束满足 $t_i+w_{i\to i+1}\leqslant t_{i+1}$。$P$ 的费用记为：

$$\mathrm{cost}(P)=f_{1\to 2}(t_1)+f_{2\to 3}(t_2)+\cdots+f_{k\to k+1}(t_k)=\sum_{i=1}^{k} f_{i\to i+1}(t_i)$$

定义 3 时间依赖图上的具有时间限制的最小费用路径查询。给定一个时间依赖图 G_T，起点 v_s，终点 v_d，时间 td、ta，G_T 上的具有时间限制的费用最小路径查询返回路径 P，使得：1)P 在 td 或之后从 v_s 出发，在 ta 或之前到达 v_d；2) 所有满足条件 1) 的路径中，P 的费用最小。

图 1a 显示了一个时间依赖图，边上的数字表示 $w_{i\to j}$。图 1b～图 1f 为各条边的费用函数。若查询从 v_0 到 v_3 的最小费用路径，td=0，ta=10，则 $P=v_0@2\to v_1@5\to v_3$ 是其中一个解，cost(P)=5。



a. 时间依赖图　b. $f_{0\to 1}(t)$　c. $f_{0\to 2}(t)$

d. $f_{1\to 2}(t)$　e. $f_{1\to 3}(t)$　f. $f_{2\to 3}(t)$

图 1　一个时间依赖图及各条边的费用函数

3　逆向搜索

表 1　文中常用符号的含义

符号	含义
N_i^+, N_i^-	分别表示顶点v_i的出边邻居和入边邻居的集合
v_s,v_d,td,ta	查询的起点、终点、最早时刻、最晚到达时刻
$w_{i\to j}$	通过边$\langle v_i,v_j\rangle$的时间
$f_{i\to j}(t)$	在t时刻从v_i出发，通过边$\langle v_i,v_j\rangle$的费用
$g_i(t)$	t时刻位于v_i，到达v_d的最小费用
edt_i	在td时刻从v_s出发，从v_i出发的最早时刻
ldt_i	在ta时刻到达v_d，从v_i出发的最晚时刻
$g_{i\to j}(t)$	在t时刻位于v_i，途经$\langle v_i,v_j\rangle$到达v_d的最小费用
$g_i^{\rightarrow}(t)$	从v_s出发，在t时刻位于v_i的最小费用
$g_i^{\leftarrow}(t)$	在t时刻位于v_i，到达v_d的最小费用

文中变量在表 1 中给出具体定义。

3.1　出发时间–最小代价函数

因为路径的费用与时间相关，所以定义顶点 v_i 的出发时间–最小代价函数为 $g_i(t)$，表示若在 t 时刻位于 v_i，则在 ta 时刻或之前到达 v_d 的最小费用。在 t 时刻位于 v_i，指在 t 时刻出发，或者在 v_i 上等待至 t 时刻后的某个时刻出发。

查询的起始顶点是 v_s，故搜索的目标是计算 $g_s(t)$ 的最小值。

3.2　更新出发时间–最小费用函数

本节讨论在已知顶点 v_i 的邻居的出发时

间−最小费用函数的情况下，如何更新 $g_i(t)$。用 N_i^+表示 v_i 的出边邻居的集合。设 $v_j \in N_i^+$，$g_{i\to j}(t)$ 表示在 t 时刻位于 v_i 途经$\langle v_i,v_j\rangle$再从 v_j 到达 v_d 的最小费用。由于从 v_i 出发的路径只能通过 N_i^+中的顶点到达 v_d，所以 $g_i(t)$ 的值取决于 $g_{i\to j}(t)$ ($v_j \in N_i^+$)：

$$g_i(t) = \min\{g_{i\to j}(t) | v_j \in N_i^+\} \tag{1}$$

$g_i(t)$ 的求解过程可以分成两步：1) 求 $g_{i\to j}(t)$；2) 按式 (1) 更新 $g_i(t)$。第 1) 步的计算如下。

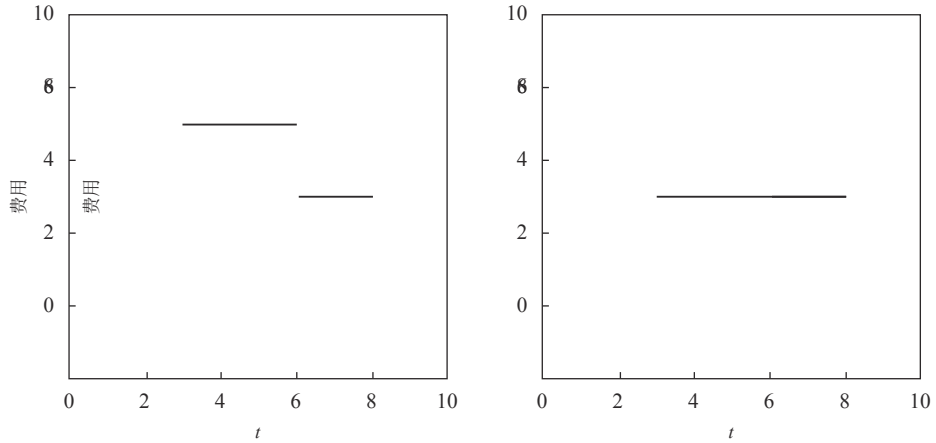
$g_i($定义域为 $[\mathrm{edt}_i, \mathrm{ldt}_i]$。各顶点的 edt_i 值可以在以 $w_{i\to j}$ 为边$\langle$的权值图上，从 v_s 出发调用一次 Dijkstra 搜索得；同理，各点的 ldt_i 值也可以从 v_d 出发调用一次逆向 Dijkstra 搜索求得。

若不考虑在 v_i 上等待，有：

$$g_{i\to j}(t) = f_{i\to j}(t) + g_j(t+w_{i\to j}) \tag{2}$$

若考虑上在 v_i 上等待，则 $g_{i\to j}(t)$ 是一个单调递增函数：若 $t_1<t_2$，则 $g_{i\to j}(t_1)\leqslant g_{i\to j}(t_2)$。这是因为：若在 t_2 时出发的费用小，则在 t_1 时可以选择等待至 t_2 时才出发。结合式 (1) 可得，$g_i(t)$ 也是一个单调递增函数。

可见 $g_{i\to j}(t)$ 的计算分成两步：首先通过式 (2) 计算 $g_{i\to j}(t)$，然后将 $g_{i\to j}(t)$ 变成单调函数。图 2 求 $g_{1\to 3}(t)$ 的计算过程，已知 $g_3(t)=0$。



a. 通过式 (2) 计算　　b. 变成单调递增函数

图 2　求 $g_{1\to 3}(t)$ 的过程

引理 1　若已知 $g_j(t)$，上述方法计算的 $g_{i\to j}(t)$ 是在 t 时刻位于 v_i，途经边$\langle v_i,v_j\rangle$到达 v_d 的最小费用函数。

3.3　增量式计算 $g_s(t)$

回顾查询的目标是求 $g_s(t)$ 的最小值，并非 $g_s(t)$ 在整个定义域上的值。已知 $g_j(t)$ 求 $g_{i\to j}(t)$，对 $g_j(t)$ 的定义域按 g_j 的取值分成若干段，按定义域上的取值从大的顺序求 $g_{i\to j}(t)$在各分段的值。以在搜索过程中，也可以先后用 $g_j(t)$ 的某个分段的值计算 $g_{i\to j}(t)$，不需要在完整计算出 $g_j(t)$ 后才计算 $g_{i\to j}(t)$。增量式计算的思想是：为了减少计算量，当 $g_j(t)$ 的某个段的值已被正确计算出来后，才用来计算 $g_{i\to j}(t)$。计算过程如算法 1 所示。

算法 1 Reverse Search

输入：G_T，起点 v_s、终点 v_d、最早出发时刻 td、最晚到达时刻 ta。

输出：最小费用。

1) 分别从 v_s 和 v_d 调用 Dijkstra 搜索，用 td、ta 求各顶点 v_i 的 edt_i 和 ldt_i 值。
2) $\forall v_i \in V-\{v_d\}$, $g_i(t)=+\infty$; $g_d(t)=0$; $T_i=\mathrm{edt}_i$;
3) 将元组 $(0, v_d, \mathrm{edt}_d, \mathrm{ldt}_d)$ 加进 H
4) while (!H.empty()) do
5)　从 H 弹出元组 $(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$
6)　if (v_j==v_s) then return c_j
7)　for each $v_i \in N_j^-$ do
8)　　$g_{i\to j}(t-w_{i\to j}) = f_{i\to j}(t-w_{i\to j}) + c_j$, $\mathrm{td}_j \leqslant t < \mathrm{ta}_j$
9)　　将 $g_{i\to j}(t)$ 变成单调递增函数，更新 g_i;
10)　　$c_i = \min\{g_i(t) \mid T_i \leqslant t\}$
11)　　设 $g_i(t)$ 取值 c_i 的区间右端点是 ta_i;
12)　　将元组 $(c_i, v_i, T_i, \mathrm{ta}_i)$ 加进 H
13)　$T_j = \mathrm{ta}_j$;
14)　$c_j = \min\{g_j(t) \mid t \geqslant \mathrm{ta}_j\}$
15)　设 $g_j(t)$ 取值 c_j 的区间的右端点是 ta_j;
16)　将元组 $(c_j, v_j, T_j, \mathrm{ta}_j)$ 加进 H;
17) return $+\infty$;

除 v_d 以外，各顶点 v_i 的 $g_i(t)=+\infty$，表示 $g_i(t)$ 在定义域 $[\mathrm{edt}_i, \mathrm{ldt}_i]$ 上的取值未确定，T_i 的取值将在下文中。算法迭代地选取某个顶点 v_j 和相

应的时间区间扩展，更新 v_j 的入边邻居的费用，直到找到从 v_s 出发的最小费用路径为止。

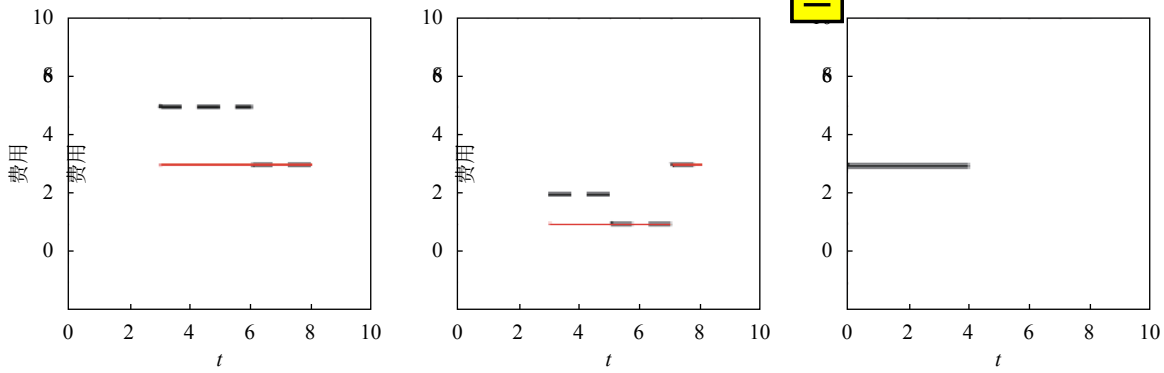
算法维护一个最小堆 H，H 中的每个元素是一个四元组 $(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$，表示顶点 v_j 的函数 $g_j(t)$ 在区间 $[\mathrm{td}_j, \mathrm{ta}_j)$ 的取值为 c_j。H 中的元组以 c_j 为关键字排序。初始时 H 仅包含一个元组 $(0, v_s, \mathrm{edt}_d, \mathrm{ldt}_d)$(第 3 行)。当 $(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$ 弹出时，$g_j(t)$ 在 $[\mathrm{td}_j, \mathrm{ta}_j)$ 区间上的取值已被正确计算出来，为 c_j，这一点将在定理 1 中证明。当 v_s 首次弹出时，最小费用路径已找到(第 6 行)。

若 v_j 不是 v_s，用 v_j 在 $[\mathrm{td}_j, \mathrm{ta}_j)$ 区间上的取值 c_j 计算 $g_{i\to j}(t)$(第 7～12 行，详细的计算过程将在下一段讨论)。每个顶点 v_j 附带一个值 T_j，表示 $g_j(t)$ 的已确定值的时间的右端点。初始时，$T_j=\mathrm{edt}_j$。当 v_j 首次从 H 弹出时，由定理 1 可知，c_j 是 $g_j(t)$ 的最小值，ta_j 是 $g_j(t)=c_j$ 的时间区间的右端点，接下来用 $g_j(t)$ 在 $[\mathrm{edt}_j, \mathrm{ta}_j)$ 这一段的值计算 $g_{i\to j}(t)$。由于 $g_j(t)$ 是单调递增函数，所以 $g_j(t)$ 的下一个最小值的区间的左端点是 ta_j，于是在第 13 行令 $T_j=\mathrm{ta}_j$，表示下一步将用 $g_j(t)$ 在 $[T_j, \cdot)$ 上的最小值更新 $g_{i\to j}(t)$。因此，$(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$ 扩展完毕后，继续找出 $g_j(t)$ 在区间在 $[T_j, \cdot)$ 上的最小值 c_j，将 v_j、c_j 和 c_j 取值对应的时间区间加进 H。

扩展 v_j 的详细过程如下。第 8) 行计算得到的 $g_{i\to j}(t)$ 并非单调函数，需要把 $g_{i\to j}(t)$ 变成单调递增函数。按式 (1) 更新 $g_i(t)$，之后需要重新找出 $g_i(t)$ 在未确定取值的时间区间上的最小值 c_i(第 10) 行)。第 12) 行中，若 H 中没有包含顶点是 v_i 的四元组，则将 $(c_i, v_i, T_i, \mathrm{ta}_i)$ 加进 H；若 H 包含顶点是 v_i 的四元组 $(c_i', v_i', \mathrm{td}_i', \mathrm{ta}_i')$，而且费用大于 c_i，则用 $(c_i, v_i, T_i, \mathrm{ta}_i)$ 代替 $(c_i', v_i', \mathrm{td}_i', \mathrm{ta}_i')$。

计算过程如图 3 所示。求 v_0 到 v_3 的最小费用路径，td=0, ta=10。算各顶点的最早出发时刻和最晚出发时刻，得到顶点 v_i 的 $g_i(t)$ 函数的定义域分别是：$v_0[0,5]$, $v_1[3,8]$, $v_2[3,8]$, $v_3[5,10]$。初始时，$T_0=0$, $T_1=3$, $T_2=3$, $T_3=5$。首先，$g_3(t)=0$，H 包含四元组 $(0,v_3,5,10)$。

一次迭代，$(0,v_3,5,10)$ 从 H 中弹出。扩展 $\langle v_1,v_3\rangle$，得到的 $g_1(t)$ 如图 3a 所示，细实线表示将 $g_1(t)$ 变成单调递增函数，下此时 $g_1(t)$ 的最小值是 5，所以将元组 $(5,v_1,3,8)$ 加进 H。同理，扩展 $\langle v_2,v_3\rangle$，得到 $g_2(t)$ 如 b 所示，将 $(3,v_2,3,7)$ 加进 H。



a. 第一次迭代 $g_1(t)$　　b. 第一次迭代 $g_2(t)$　　c. 第二次迭代 $g_0(t)$

图 3　v_0 到 v_3 的最小费用路径的计算过程

第二次迭代，$(3,v_2,3,7)$ 从 H 中弹出。扩展 $\langle v_0,v_2\rangle$，得到 $g_0(t)$ 如图 3c 所示，将元组 $(5, v_0, 0, 4)$ 加进 H。扩展 $\langle v_1,v_2\rangle$ 得到的 $g_1(t)$ 无更新。$g_2(t)$，$(3,v_2,3,7)$ 扩展完毕，将 T_2 改成 7。$g_2(t)$ 在 $t\geq 7$ 上的最小值是 5，所以将 $(5,v_2,7,8)$ 加进 H。

第三次迭代，$(5,v_0,6,8)$ 从 H 中弹出，至此已找到 v_0 到 v_3 的最小费用 5。

3.4　算法 Reverse Search 的改进与正确性证明

算法 Reverse Search 与 Two-Step 算法相比，在两方面缩小了搜索空间：1) Reverse Search 将 $g_i(t)$ 的定义域缩小至 $[\mathrm{edt}_i,\mathrm{ldt}_i]$，而 Two-Step 把正向搜索的定义域限定在 $[\mathrm{edt}_i,\mathrm{ta}]$。实际上 $[\mathrm{ldt}_i, \mathrm{ta}]$ 区间上的计算是无意义的。尤其对于路径上靠近起点的顶点 v_i，其 ldt_i 的值可能比 ta 小很多，且在 $[\mathrm{ldt}_i, \mathrm{ta}]$ 上的费用也比小 (由于 v_i 靠近起点)，这意味着搜索初期用了 $[\mathrm{ldt}_i, \mathrm{ta}]$ 中的一些子区间扩展路径，而这部分扩展是不可能生成在 ta 前到达终点的路径的；2) 在第 8) 行，Reverse Search 仅用 $[T_j, \mathrm{ta}_j)$ 区间计算 $g_{i\to j}(t)$，剩余区间 $[\mathrm{ta}_j, \cdot)$ 不用。因为此时 $g_j(t)$ 在 $[\mathrm{ta}_j, \cdot)$ 上的取值尚未确

定，仍可能被 v_j 的出边邻居更新，用未确定的值来计算 $g_{i\to j}(t)$ 也是无意义的。

证明方法如下。

定理 1 在 Reverse Search 中，当 $(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$ 弹出时，$g_j(t)$ 函数在 $[\mathrm{td}_j, \mathrm{ta}_j)$ 上的取值已确定，即为 c_j。

证明：以示区分，用 $g_j(t)$ 表示正确的出发时间−最小费用函数；用 $g_j^*(t)$ 表示元组 $(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$ 弹出时，当前计算出的值。证明 $g_j^*(t)=g_j(t)$，$\mathrm{td}_j\leqslant t<\mathrm{ta}_j$。

采用数学归纳法。对迭代 $(0, v_d, \mathrm{edt}_i, \mathrm{ldt}_d)$ 弹出时，$g_d^*(t)=g_d(t)=0$。假设在前 k 次迭代中命题成立，证明第 $k+1$ 次迭代时命题仍然成立。不妨设存在 $t_j\in[\mathrm{td}_j, \mathrm{ta}_j)$，使得 $g_j^*(t_j)\neq g_j(t_j)$。由于 $g_j(t_j)$ 表示正确的最小费用，所以有 $g_j^*(t_j)>g_j(t_j)$。

设在 t_j 时刻从 v_j 出发的最小费用路径 $P_j=v_j@t_j\to\cdots v_u@t_u\to v_w@t_w\to\cdots\to v_d$。由于：$g_j^*(t)\neq g_j(t_j)$ 且 $g_d^*(t)=g_d(t)=0$，所以在 P_j 上，存在两个相邻的顶点 v_u 和 v_w，v_u 在时刻 t_u 出发，v_w 在时刻 t_w 出发，使得在前 k 次迭代中，v_w 的包含 t_w 的时间区间的元组曾经从 H 中弹出，v_u 的包含 t_u 的时间区间的元组未从 H 中弹出。分别记 $P_u(P_w)$ 为 P_j 上在 t_u 时刻出发 (在 t_w 时刻从 v_w 出发) 的子路径。

每条边上的费用都是非负数，所以 $\mathrm{cost}(P_u)\leqslant\mathrm{cost}(P_j)$。又因为：$\mathrm{cost}(P_u)=f_{u\to w}(t_u)+\mathrm{cost}(P_w)$ 且 P_j 是最小费用路径，即 $\mathrm{cost}(P_j)=g_j(t_j)$，所以：

$$f_{u\to w}(t_u)+\mathrm{cost}(P_w)\leqslant g_j(t_j) \tag{3}$$

$g_w(t_w)$ 是在时刻 t_w 位于 v_w、到达 v_d 的最小费用，所以 $g_w(t_w)\leqslant\mathrm{cost}(P_w)$。结合式 (3)，有:

$$f_{u\to w}(t_u)+g_w(t_w)\leqslant g_j(t_j) \tag{4}$$

v_w 的包含 t_w 的时间区间的元组从 H 弹出时，曾用 $f_{u\to w}(t)$ 计算 $g_{u\to w}(t)$，接着用 $g_{u\to w}(t)$ 更新 $g_u^*(t)$，因此 $g_u^*(t_u)\leqslant f_{u\to w}(t_u)+g_w^*(t_w)$。按归纳，$v_w$ 的包含 t_w 的时间区间的元组从 H 弹出时，$g_w^*(t_w)=g_w(t_w)$。因此：

$$g_u^*(t_u)\leqslant f_{u\to w}(t_u)+g_w(t_w) \tag{5}$$

结合式 (4) 和式 (5)，可得 $g_u^*(t_u)\leqslant g_j(t_j)<g_j^*(t_j)$。由于 v_u 的包含 t_u 的时间区间的四元组也未从 H 弹出，这与“第 $k+1$ 次迭代时选取 v_j 的包含 t_j 的四元组”矛盾。证毕。

4 双向搜索

正向搜索计算顶点 v_i 的到达时间−费用函数：$g_i(t)$ 表示在 td 时刻或之后从 v_s 出发，若在 t 时刻位于 v_i 时的最小费用。搜索过程参考第 3 节的讨论。

双向搜索的思路是：分别从 v_s 和 v_d 启动正向搜索和逆向搜索，两个搜索交替进行，直到两个搜索相遇为止。具体地，用 $c_\perp$ 表示当前找到的路径的最小费用，初始时，$c_\perp=+\infty$。用 $g_i^{\rightarrow}(t)$ 表示顶点 v_i 的到达时间−最小费用函数，$g_i^{\leftarrow}(t)$ 表示 v_i 的出发时间−最小费用函数。分别用 $H^{\rightarrow}$ 和 $H^{\leftarrow}$ 表示正向搜索和逆向搜索的最小堆。假设当前四元组 $(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$ 从 $H^{\leftarrow}$ 弹出，v_i 是 v_j 一个入边邻居，用算法 1 更新 $g_i^{\leftarrow}(t)$ 后，利用 $g_i^{\rightarrow}(t)$ 更新 $c_\perp$:

$$c_\perp=\min\{c_\perp, g_i^{\rightarrow}(t_f)+g_i^{\leftarrow}(t_b)\mid t_f\leqslant t_b\} \tag{6}$$

双向搜索的终止条件与重构最小费用路径：在搜索过程中，$c_\perp$ 的值不断地更新。当正向 (或逆向) 搜索弹出顶点 v_{mid} 得 v_{mid} 的正向搜索的已确定值的时间区间中存在时间点 t_f，且 v_{mid} 的逆向搜索的已确定值的时间区间中存在时间点 t_b，满足 $t_f\leqslant t_b$，搜索终止，$c_\perp$ 即为所求的最小费用。最小费用路径并不一定途经 v_{mid}，而是途经令 $c_\perp$ 取得最小值的顶点 v_i。

设当前从最小堆中弹出顶点 v，如何判断 v 存在时间点 t_f 与 t_b，$t_f\leqslant t_b$ 且 $g_i^{\rightarrow}(t_f)$ 和 $g_i^{\leftarrow}(t_b)$ 的值均已确定回顾在逆向搜索中，每个顶点 v_i 附带一个值 $T_i^{\leftarrow}$，表示已确定值的时间区间的右端点；对称地，在正向搜索中，每个顶点 v_i 附带一个值 $T_i^{\rightarrow}$，表示已确定值的时间区间的左端点。在搜索过程中，$T_i^{\leftarrow}$ 值从 edt_i 开始递增；$Ti^{\rightarrow}$ 从 ldt_i 开始递减。当 $T_i^{\rightarrow}\leqslant T_i^{\leftarrow}$ 时表明存在 t_f 和 t_b 如上述描述。

定理 2 当双向搜索算法终止时，所得的 $c_\perp$ 即为最小费用。

证明：首先，当四元组 $(c_j, v_j, \mathrm{td}_j, \mathrm{ta}_j)$ 从正向搜索的最小堆 $H^{\rightarrow}$ 弹出时，$g_j^{\rightarrow}$ 在 $[\mathrm{td}_j, \mathrm{ta}_j)$ 区间上的取值已确定为 c_j。证明过程与定理 1 同理。

当前 (或逆向) 搜索从最小堆中弹出顶点 v_{mid}，使得 v_{mid} 在正向和逆向搜索的已确定值的时间区间存在时间点 t_f 和 t_b，满足 $t_f\leqslant t_b$，搜索终止。假设正确的最小费用是 c，分两种情况：1) $c=g_{\mathrm{mid}}^{\rightarrow}(t_f)+g_{\mathrm{mid}}^{\leftarrow}(t_b)$；2) $c<g_{\mathrm{mid}}^{\rightarrow}(t_f)+g_{\mathrm{mid}}^{\leftarrow}(t_b)$。

1) 若 $c=g_{\mathrm{mid}}^{\rightarrow}(t_f)+g_{\mathrm{mid}}^{\leftarrow}(t_b)$，说明 v_{mid} 在一条最小费用路径 P 上，在 t_f 时刻到达 v_{mid}，在 t_b 时刻离开 v_{mid}。设 P 上 v_{mid} 的前驱顶点是 v_x，v_{mid} 的后继顶点是 v_y。若 v_x 在正向搜索中先出堆，v_y 在逆向搜索中后出堆，则在 v_x 出堆时，由[illegible]知，能通

过扩展边$\langle v_x, v_{mid}\rangle$正确计算出 $g_{mid}^{\rightarrow}(t_f)$ 的值；之后在 v_y 出堆时，能通过扩展边$\langle v_{mid}v_y\rangle$正确计算出 $g_{mid}^{\leftarrow}(t_b)$。再通过式(6)计算出 $c_{\perp}=g_{mid}^{\rightarrow}(t_f)+g_{mid}^{\leftarrow}(t_b)$。对称地，若 v_y 先向搜索中先出堆，v_x 在正向搜索中后出堆，也能正确计算出 c。

2) $g_{mid}^{\rightarrow}(t_f)+g_{mid}^{\leftarrow}(t_b)$。设 P 是最小费用路径，即 cost(P)=c。v_k 是 P 上的一个顶点，用 $P_k^{\rightarrow}$表示 P 上从 v_s 到 v_k 的那一截前缀路径，$P_k^{\leftarrow}$表示 P 上从 v_k 到 v_d 的那一截后缀路径。必定存在两个相邻的顶点 v_x、v_y，即 $P=v_s@t_s\rightarrow\cdots\rightarrow v_x@t_x\rightarrow v_y@t_y\rightarrow\cdots\rightarrow v_d$，使得：

$$\text{cost}(P_x^{\rightarrow}) < g_{mid}^{\rightarrow}(t_f) \qquad \text{cost}(P_y^{\rightarrow}) \geqslant g_{mid}^{\rightarrow}(t_f) \tag{7}$$

设 $P_x^{\rightarrow}$在时刻 ta_x 到达 v_x，由 $g_x^{\rightarrow}(t)$ 的定义得：

$$g_x^{\rightarrow}(ta_x) \leqslant \text{cost}(P_x^{\rightarrow}) < g_{mid}^{\rightarrow}(t_f) \tag{8}$$

由于 cost(P)=cost($P_y^{\rightarrow}$)+cost($P_y^{\leftarrow}$)，结合本节的假设 cost(P)<$g_{mid}^{\rightarrow}(t_f)+g_{mid}^{\leftarrow}(t_b)$，得：

$$\text{cost}(P_y^{\rightarrow}) - g_{mid}^{\rightarrow}(t_f) < g_{mid}^{\leftarrow}(t_b) - \text{cost}(P_y^{\leftarrow}) \tag{9}$$

结合式(7)和式(9)，cost($P_y^{\leftarrow}$)<$g_{mid}^{\leftarrow}(t_b)$。按$g_y^{\leftarrow}(t_y)$的定义，$g_y^{\leftarrow}(t_y)\leqslant c(P_y^{\leftarrow})$。于是，$g_y^{\leftarrow}(t_y)<g_{mid}^{\leftarrow}(t_b)$。说明在逆向搜索中，$v_y$ 的包含 t_y 的时间区间的元组比 v_{mid} 的包含 t_b 的时间区间的元组先出堆。结合定理 1 和式(8)，说明在搜索终止之前 1) 的包含 t_y 的时间区间曾从 $H^{\leftarrow}$中弹出；2) v_x 的包含 ta_x 的时间区间的元组曾从中 $H^{\rightarrow}$中弹出。不妨设 2) 先发生，1) 后发生。在 1) 发生时，$g_x^{\rightarrow}(ta_x)$已正确计算出来，且在逆向搜索扩展$\langle v_x,v_y\rangle$时，也正确计算出 $g_x^{\leftarrow}(t_x)$，此时利用式(6)可算出正确的最小费用 $c_{\perp}$。当 1) 先发生，2) 发生，也能得出相同的结论。证毕。

5 实 验

实验采用 Visual Studio 2017Community C++编写，Windows 10 平台运行，机器配置 Intel(R) Core (TM) i78700K CPU 和 64 GB 内存。

5.1 数据集

本文采用两组路网数据集测试 (http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm)。OL 描述了奥尔登堡的路网，包括 6 105 个顶点和 7 035 条边；CA 描述了加利福尼亚州的路网，包括 20 147 个顶点和 21 692 条边。同文献 [13-14] 的测试方法，本文在路网的基础上生成时间依赖图。图中每条边$\langle v_i,v_j\rangle$的时间代价 $w_{i\rightarrow j}$ 取路网的边的权值。费用函数 $f_{i\rightarrow j}(t)$ 的定义域设为 [0, 20 000]。将 $f_{i\rightarrow j}(t)$ 的定义域随机分成 k 段，每一段上随机生成一个 [20, 100] 内的整数。k 分别取 10 和 20。

分别在 OL 和 CA 上随机生成查询集。查询集共包含 10 000 个查询，随机生成每个查询的起点和终点。首先计算出每个查询的从起点到终点的最快到达时间，近似地用来衡量起点到终点的远近。将这 10 000 个查询按最快到达时间升序排序，每 1 000 个查询分成一组，分别将这 10 组查询记为 $Q1$,$Q2$, …, $Q10$。于是，$Q1$ 组内的查询的起点和终点相距较近，$Q10$ 组内的相距较远。为了保证大部分查询有解，每个查询的 td 限定在 [0, 10 000] 范围，ta 在 [10 000, 20 000] 范围内随机生成。

实验分别在以下 4 方面测试算法：1) Reverse Search 算法与文献 [13-14] 的 Two-Step 算法的效率对比；2)～4) 从起点和终点之间的远近、图的规模和费用函数上分段数量 k 这 3 方面对比单向 (正向) 搜索和双向搜索的性能。以示公平，单、双向搜索均使用了第 3 节提出的增量式搜索方法。算法考察的性能指标为平均每个查询的时间，以 ms 为单位。

5.2 实验结果

1) Reverse Search 与 Two-Step 算法效率对比。图 4 显示了 OL 数据集 k=10 的时间依赖图下，Reverse Search 和 Two-Step 对 10 组查询集的平均耗时。起止点相距越远的查询效率差越大。这是因为 Reverse Search 与 Two-Step 相比，缩小了每个顶点的有效时间区间，而且等某个顶点在某时间段出发的最小费用正确计算出来以后，才用来计算扩展路径的费用。这两个优化能减少部分无用的计算。

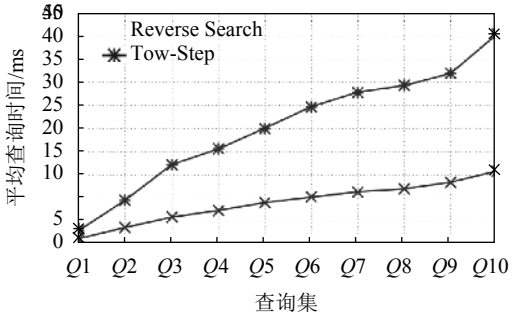


图 4 k=10 下的平均查询时间

2) 起点和终点之间的远近对单向搜索和双向搜索的效率影响。图 5 显示了 OL 和 CA 在 k=10 的时间依赖图下，查询集 $Q1$,$Q2$,…,$Q10$ 每个查询

的平均花费时间，y 轴用对数的比例显示。图例标记的前两个字母表示路网名称，S 表示单向搜索，B 表示双向搜索。随着起点与终点的距离增加，两者的效率差尤为明显。例如在 OL 数据集下的 $R1$ 查询集中，单向、双向搜索平均每个查询花费 1.25 ms 和 0.98 ms；在 $R10$ 查询集中，单向、双向搜索平均每个查询花费 15.87 ms 和 4.18 ms，双向搜索耗时仅是单向搜索的 26.3%。造成较大的效率差是因为双向搜索能减少搜索过程中访问的顶点数，而由于搜索过程中一个顶点可能多次被多次访问，每次访问对应不同的时间区间。因此，双向搜索通过减少访问的顶点数提高了查询效率。留意到实验中的单向搜索已经使用了第 3 节提出的增量式搜索优化。与 Two-Step 相比，对于查询相距较远的顶点，双向搜索的平均每个查询耗时约是 Two-Step 的 10%。

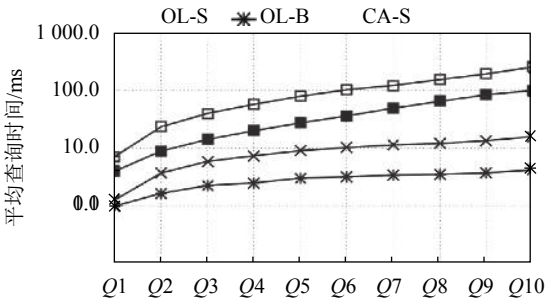


图 5　OL 和 CA 的 k=10 下的平均查询时间

3) 图的规模对算法效率的影响。从图 5 显示的数据看出，CA 的顶点数和边数约是 OL 的 4 倍，但平均查询时间 CA 约是 OL 的 10 倍，而且随着查询起止点的距离增大，两个图上的查询的效率差尤为明显。这是因为图规模变大后，两点之间的可选的路径数暴涨，所以查询也更耗时。

4) 函数上的分段数量 k 的影响。图 6 以 CA 为例显示了 k=10 和 k=20 时的平均查询时间。总体来说，不管用单向搜索还是双向搜索，k 越大，平均每个查询时间越长。这是因为在搜索过程中每个顶点会反复从最小堆中弹出，每次弹出的顶点附带不同的时间区间，分段越多顶点上的时间区间也越多，因此顶点反复弹出次数越多。图 7 从另一个角度展示了当路网与查询集固定的情况下，k=10 和 k=20 下的两种搜索的查询时间对比。有趣的是，不管使用哪种搜索，k 变大时引起的平均查询时间增加的比率相对稳定。

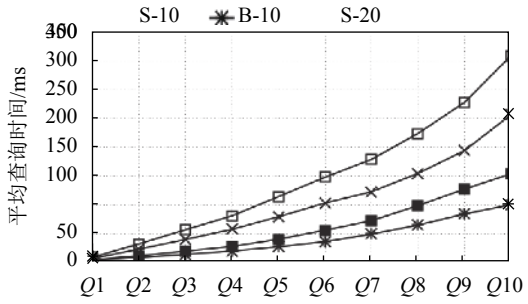


图 6　k=10, 20 下两种搜索的平均查询时间

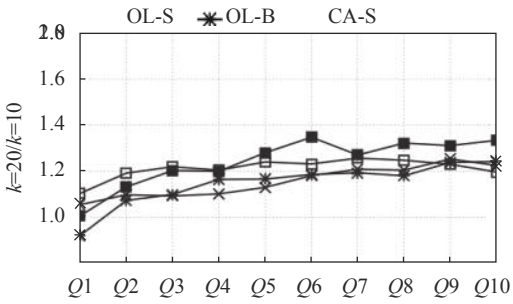


图 7　平均查询时间对比

6　结 束 语

本文提出了一种时间依赖图下的最小费用路径查询的高效算法。首先通过缩小顶点的有效时间区间与延迟计算路径费用的方法减少部分无意义的计算；然后提出双向搜索的方法削减了搜索空间，提高了算法的效率。理论上证明了方法的正确性，实践上用真实路网验证了方法的有效性。下一步工作拟进一步将大规模图查询的索引方法引入本问题中，研究在时间依赖图下的最小费用路径索引算法。

参　考　文　献

[1] DIJKSTRA E W. A note on two problems in connection with graphs[J]. Numerische Mathematics, 1959, 1(1): 269-271.

[2] WU L, XIAO X, DENG D, et al. Shortest path and distance queries on road networks: An experimental evaluation[J]. Proceedings of the VLDB Endowment, 2012, 5(5): 406-417.

[3] 刘贵松, 解修蕊, 黄海波, 等. 基于最短路径信任关系的推荐项目方法[J]. 电子科技大学学报, 2014, 43(2): 162-166.
LIU Gui-song, XIE Xiu-rui, HUANG Hai-bo, et al. Fast computing method for items recommendation based on shortest-path trust relationship[J]. Journal of University of Electronic Science and Technology of China, 2014, 43(2): 162-166.

[4] WU H, CHENG J, HUANG S, et al. Path problems in temporal graphs[J]. Proceedings of the VLDB Endowment,

2014, 7(9): 721-732.
[5] WANG S, LIN W, YANG Y, et al. Efficient route planning on public transportation networks: A labelling approach[C]// International Conference on Management of Data. Melbourne, Australia: [s. n.], 2015: 967-982.
[6] STRASSER B, WAGNER D. Connection scan accelerated[C]//Proceedings of the meeting on Algorithm Engineering and Experimentation. [S. l.]: [ACM], 2014: 125-137.
[7] IDRI A, OUKARFI M, BOULMAKOUL A, et al. A new time-dependent shortest path algorithm for multimodal transportation network[J]. Procedia Computer Science, 2017, 109: 692-697.
[8] BATZ G V, GEISBERGER R, NEUBAUER S, et al. Time-dependent contraction hierarchies and approximation[C]// Symposium on Experimental and Efficient Algorithms. Heidelberg, Berlin: Springer, 2010: 166-177.
[9] BAST H, DELLING D, GOLDBERG A V, et al. Route planning in transportation networks[J]. Data Structures and Algorithms, 2016, 9220: 19-80.
[10] LIU G, QIU Z, QU H, et al. Computing k shortest paths using modified pulse-coupled neural network[J]. Neurocomputing, 2015, 149: 1162-1176.
[11] LIU G, QIU Z, QU H, et al. Computing k shortest paths from a source node to each other node[J]. Soft Computing, 2015, 19(8): 2391-2402.
[12] HU X B, CHIU Y C. A constrained time-dependent k shortest paths algorithm addressing overlap and travel time deviation[J]. International Journal of Transportation Science and Technology, 2015, 4(4): 371-394.
[13] 杨雅君, 高宏, 李建中. 时间依赖代价函数下的最优路查询问题[J]. 计算机学报, 2012, 35(11): 2247-2264.
YANG Ya-jun, GAO Hong, LI Jian-zhong. Finding the optimal path under time-dependent cost function on graphs[J]. Chinese Journal of Computers, 2012, 35(11): 2247-2264.
[14] YANG Y, GAO H, YU J X, et al. Finding the cost-optimal path with time constraint over time-dependent graphs[J]. Proceedings of the VLDB Endowment, 2014, 7(9): 673-684.
[15] LI L, WEN H, DU X, et al. Minimal on-road time route scheduling on time-dependent graphs[J]. Proceedings of the VLDB Endowment, 2017, 10(11): 1274-1285.
[16] LI L, ZHENG K, WANG S, et al. Go slow to go fast: Minimal on-road time route scheduling with parking facilities using historical trajectory[J]. Very Large Data Bases, 2018, 27(3): 321-345.
[17] KOMAI Y, NGUYEN D H, HARA T, et al. kNN Search utilizing index of the minimum road travel time in time-dependent road networks[C]//Symposium on Reliable Distributed Systems. [S. l.]: IEEE, 2014: 131-137.
[18] BISWAS S, GANGULY A, SHAH R, et al. Restricted shortest path in temporal graphs[C]//Database and Expert Systems Applications. Hanoi, Vietnam: [s. n.], 2015: 13-27.
[19] DEAN B C. Algorithms for minimum-cost paths in time-dependent networks with waiting policies[J]. Networks, 2004, 44(1): 41-46.
[20] CAI L Y, SHA D. Shortest paths subject to time-varying stochastic transit times[C]//International Conference on Service Systems and Service Management. [S. l.]: IEEE, 2011: 1-2.

编 辑 叶 芳

(上接第 457 页)

[3] GONZELEZ-PELCIC N, ALI A, VA V, et al. Millimeter-wave communication with out-of-band information[J]. IEEE Commun Mag, 2017, 55(12): 140-146.
[4] HASHEMI M, KOKSAL C E, SHROFF N B. Out-of-band millimeter wave beamforming and communications to achieve low latency and high energy efficiency in 5G systems[J]. IEEE Trans Commun, 2018, 66(2): 875-888.
[5] JIANG Z, MOLISCH A F, CAIRE G. Achievable rates of FDD massive MIMO systems with spatial channel correlation[J]. IEEE Transactions on Wireless Communications, 2015, 14(5): 2868-2882.
[6] PETER M, AKI K, KATSUYUKI H. Measurement campaigns and initial channel models for preferred suitable frequency ranges[EB/OL]. [2016-03-20]. https://bscw.5g-mmmagic.eu/pub/bscw.cgi/d94832.
[7] ROH W, SEOL J Y, PARK J, et al. Millimeter-wave beamforming as an enabling technology for 5G cellular communications: Theoretical feasibility and prototype results[J]. IEEE Commun Mag, 2014, 52(2): 106-113.
[8] SEMIARI O, SAAD W, BENNIS M, et al. Integrated millimeter wave and Sub-6GHz wireless networks: a roadmap for ultra-reliable low-latency communications [EB/OL]. [2018-02-12]. https://arxiv.org/abs/1802.03837.
[9] WIPF D P, RAO B D. An empirical Bayesian strategy for solving the simultaneous sparse approximation problem[J]. IEEE Trans Signal Process, 2007, 55(7): 3704-3716.
[10] LIN Xin-cong, WU Sheng, JIANG Chun-xiao, et al. Estimation of broadband multiuser millimeter wave massive MIMO-OFDM channels by exploiting their sparse structure[J]. IEEE Trans Wireless Commun, 2018, 17(6): 3959-3973.
[11] XIU Y, WANG W, WU J, et al. Massive MIMO downlink channel estimation based on improved CAMP-MMV algorithm[C]//2017 9th International Conference on Advanced Infocomm Technology (ICAIT). [S.l.]: IEEE, 2017: 114-119.
[12] RANGAN S. Generalized approximate message passing for estimation with random linear mixing[C]//2011 IEEE International Symposium on Information Theory Proceedings. St Petersbug, Russia: IEEE, 2011: 2168-2172.

编 辑 漆 蓉