云南大学数学与统计学院
《算法图论实验》上机实践报告

课程名称： 算法图论实验	**年级：** 2015 级	**上机实践成绩：**
指导教师： 李建平	**姓名：** 刘鹏	**专业：** 20151910042
上机实践名称： 路的插点数问题	**学号：** 20151910042	**上机实践日期：** 2018-12-31
上机实践编号： 4	**组号：**	

一、 实验目的

1. 了解最短路的最小插点问题的实际背景；
2. 能快速写出求最短路的最小插点问题的动态规划算法。

二、 实验内容

1. 写出求最短路的最小插点问题的动态规划算法；
2. 用 C 语言实现上述算法。

三、 实验平台

Windows 10 Pro 1803；
MacOS Mojave。

四、 算法设计

4.1 问题描述

给定一个无向图$G=(V,\ E;\ w)$，边权重函数$w\colon\ E\to\mathbb{R}^+$，该函数可以视为两个节点之间进行通信的时间延迟（time delay），然后给定一个正整数$d\in\mathbb{Z}_0^+$，集合$\mathcal{P}=\{(s_i,\ t_i;\ B_i)|i=1,\ 2,\cdots,\ k\}$，其中$s_i\in V$是起始点（source node），$t_i\in V$是收点（sink node）可以把集合$\mathcal{P}$看作一个通信请求限制集合，即沿着某条从$s_i$到$t_i$路径中，延迟之和不准超过$B_i$；该问题是寻找一个$G$的子图$G'=(V',\ E')$，满足：

（1） 对于集合$\mathcal{P}$中任何一个元素$(s_i,\ t_i;\ B_i)$，在G'中都有一条从从s_i到t_i的路径P_i，且该路径的总延迟$w(P_i)=\sum_{e\in P_i}w(e)\le B_i$；

（2） 在G'中的所有边上，都插入一些新的节点（可以看作是添加一些中继器或信号放大器），保证得到的边的延迟都不超过d，显然，$\forall e\in A'$，插入的节点数量应该是$I(e)=\lceil w(e)/d\rceil-1$。

（3） 目标是找一个G'，在其中总插入的节点数量最少且满足要求（1）和（2）。

4.2 路的插点问题

给定一个连通赋权图$G=(V,\ E;\ w,\ c;\ s,\ t)$以及两个正整数$d$和$B$，这里$w\colon\ E\to\mathbb{R}^+$是边的权重函数，$c\colon\ E\to\mathbb{R}^+$是边的插点函数（指往这条边插入一个点需要花费的成本），$s,\ t$是两个固定的顶点，在图G中寻找一条从s到t的路$P_{s,\ t}$，使得路$P_{s,\ t}$的权重$w(P_{s,\ t})$不超过常数B，并且向路$P_{s,\ t}$的一些特殊边上插入若干顶点，使得新的路$P_{s,\ t}^*$中任意边的权重都不超过常数d；目标：在满足上述条件下，求出插入顶点后所产生的最小费用，如果用$\mathrm{INSERT}(e)$表示在边e中插入的点的个数，那么总插点费用为$\sum_{e\in E(P_{s,\ t})}I(e)$，即求$\min\left\{\sum_{e\in E(P_{s,\ t})}I(e)\,\middle|\,\text{any }P_{s,\ t}\right\}$

该问题是 NP-完备的。

4.3 最短路的插点问题

给定一个连通赋权图$G=(V,\ E;\ w,\ c;\ s,\ t)$以及一个正整数$d$，这里$w\colon\ E\to\mathbb{R}^+$是边的权重函数，$c\colon\ E\to\mathbb{R}^+$是边的插点函数（指往这条边插入一个点需要花费的成本），$s,\ t$是两个固定的顶点，在图G中寻找一条从s到t的最短路$P_{s,\ t}$，并且向最短路$P_{s,\ t}$的一些特殊边上插入若干顶点，使得路$P^*_{s,\ t}$中任意边的权重都不超过常数d；要求：在满足上述条件下，求出插入顶点后所产生的最小费用。

如果用INSERT(e)表示在最短路的一条边e中插入的点的个数，那么总插点费用为$\sum_{e\in E(P_{s,\ t})}I(e)$，即求$\min\left\{\sum_{e\in E(P_{s,\ t})}I(e)\ \middle|\ \text{any shortest }P_{s,\ t}\right\}$。

该问题是 P 类的。

4.4 路的插点数目问题

给定一个连通赋权图$G=(V,\ E;\ w;\ s,\ t)$以及一个正整数d和B，$c\colon\ E\to\mathbb{R}^+$是边的插点函数（指往这条边插入一个点需要花费的成本），$s,\ t$是两个固定的顶点，在图G中寻找一条从s到t的路$P_{s,\ t}$，使得路$P_{s,\ t}$的权重$w(P_{s,\ t})\le B$，并且向路$P_{s,\ t}$的一些特殊边上插入若干顶点，使得路$P^*_{s,\ t}$中任意边的权重都不超过常数d；要求：在满足上述条件下，求出插入顶点的最小数目。

如果用INSERT(e)表示在路的一条边e中插入的点的个数，那么总插点个数为$\sum_{e\in E(P_{s,\ t})}I(e)$，即求$\min\left\{\sum_{e\in E(P_{s,\ t})}I(e)\ \middle|\ \text{any }P_{s,\ t}\right\}$。

该问题是 P 类的。

4.5 算法设计

关于此算法已经有了详尽的分析，材料[1]给出了更为详尽和一般化的问题描述，该问题是 NP 完备的，同时该材料也针对该问题的三个特殊情况给出了近似算法或者强多项式时间复杂度算法。本实验的内容与该材料的 single-path SCRR problem 是一致的。

4.6 问题的图论描述

为了方便叙述，这里再度把这个问题描述一遍：给定一个有向图$D=(V,\ A;\ w)$，其中$w\colon\ A\to\mathbb{R}^+$是弧权重函数，给定一个正整数$d$和一个请求$(s,\ t;\ B)$，这里的$s,\ t\in V$，$B$是延迟容忍量。single-path SCRR 问题指的是要寻找一个从s到t的有向路P，其权重值不准超过B，在这种限制下最小化总插点数目$I(P)=\sum_{e\in A(P)}I(e)$，其中$I(e)=\lceil w(e)/d\rceil-1$。

4.6.1 问题形象化描述

简单说来，这个问题类似于在一个通信网络中，给定了两个节点，要求在两者之间找一条通信路线，并且在这个路线中间加一下中继器令通信稳定性达到一定标准。现在要做的就是要找一条路，在不考虑在某条边上插入点的费用的情况下，令总的插点数目最少。不考虑在每条边上分别插入节点的单位费用，相当于在任意两条边上插点的费用都是相等的。

当然，文献[1]也指出了，当每条边的插点费用与该边具有函数关系时，这个问题是 NP 完备的。

4.7 最优化结果的上下界

给定的起始点s，对于集合V中的所有其他节点u，如果$Q_{s,\ u}$是s到u的最短路，而$P_{s,\ u}$是 single-path SCRR

问题的最优路径，那么在$P_{s,u}$上的插点数不会比$Q_{s,u}$上的插点数减去$n-1$还要小。也就是说，有

$$I(Q_{s,u}) \geq I(P_{s,u}) \geq I(Q_{s,u}) - (n-1)$$

证明：对于最优解$P_{s,u}$有

$$\begin{aligned} I(P_{s,u}) &= \sum_{e \in A(P_{s,u})} \left(\left\lceil \frac{w(e)}{d} \right\rceil - 1 \right) \\ &\geq \sum_{e \in A(P_{s,u})} \left(\frac{w(e)}{d} - 1 \right) \\ &\geq \frac{w(P_{s,u})}{d} - (n-1) \\ &\geq \frac{w(Q_{s,u})}{d} - (n-1) \\ &\geq \sum_{e \in A(Q_{s,u})} \left(\left\lceil \frac{w(e)}{d} \right\rceil - 1 \right) - (n-1) \\ &= I(Q_{s,u}) - (n-1) \end{aligned}$$

得证。 □

为了设计一个优化算法来解决 single-path SCRR 问题，需要重复地利用优化算法来找到所有的最短路，然后得到一个最短路的生成子图，文献[2]根据文献[3]设计了一种时间复杂度为$O(|V|+|E|)$的算法来快速构造快速路径生成子图G_0，记为$G_0 = \mathrm{SPSG}(G;\ s,\ t)$。

Algorithm 最短路插点问题，记此算法为HSCSP

Input $G=(V,\ E)$，起点s，终点t

Output 图G最短路$Q_{s,t}$以及加细最短路$Q'_{s,t}$，记$(Q_{s,t},\ Q'_{s,t}) = \mathrm{HSCSP}(G;\ s,\ t)$

Begin

Step 1 // 构造最短路生成子图

$G_0 = \mathrm{SPSG}(G;\ s,\ t)$

for each edge $e \in G_0$:

$I(e) = \lceil w(e)/d \rceil - 1$

Step 2 // 在重新赋权的生成子图里寻找最短路

$Q_{s,t} = \mathrm{PATH}(G_0,\ s,\ t)$

output $Q_{s,t}$

Step 3 **for each** edge $e \in Q_{s,t}$:

$I(e) = \lceil w(e)/d \rceil - 1$

for $i=1$ **through** $I(e)+1$:

$v_i =$ **new** vertex
$e_i =$ **new** edge
$e_i.\mathbf{weight} = e.\mathbf{weight}/(I(e)+1)$
if $i == 1$:
    $e_i.\mathbf{rightNode} = \mathrm{v}_i$
    $e_i.\mathbf{leftNode} = e.\mathbf{leftNode}$
else if $i == I(e)+1$:
    $e_i.\mathbf{rightNode} = e.\mathbf{rightNode}$
    $e_i.\mathbf{leftNode} = e_{i-1}.\mathbf{rightNode}$
else:
    $e_i.\mathbf{rightNode} = v_i$
    $e_i.\mathbf{leftNode} = e_{i-1}.\mathbf{rightNode}$
$Q_{s,\,t}.\mathbf{addEdge}(e_i)$
if $i \neq I(e)+1$:
    $Q_{s,\,t}.\mathbf{addVertex}(v_i)$
output $Q'_{s,\,t} = Q_{s,\,t}$

End

通过上述HSCSP算法，不仅可以解决最短路的插点问题（该问题本质上是构造最短路径子图），还可以用来解决路的插点数目问题。对任意节点u以及任意两个整数$k,\ i \geq 0$，记$\mathrm{dist}_k(u,\ i)$为从起点s到节点u的最短路$Q_{s,\,u}$的长度，而且满足限制性条件$\left|E(Q_{s,\,u})\right| \leq k$的同时，$I(P_{s,\,u}) = i$。如果这种路不存在，则记$\mathrm{dist}_k(u,\ i) = \infty$。

对于任意的i，如果满足$I(Q_{s,\,u}) - (n-1) \leq i \leq I(Q_{s,\,u})$，令$\mathrm{dist}_0(s,\ i) = 0$，$\mathrm{dist}_0(u,\ i) = \infty,\ u \neq s$，那么对于任意的$u \in V$，都有

$$\mathrm{dist}_k(u,\ i) = \mathbf{min}\left\{\mathrm{dist}_{k-1}(u,\ i),\ \mathbf{min}_{v:\,(v,\,u)\in A,\,I(v,\,u)\leq i}\{\mathrm{dist}_{k-1}\big(v,\ i - I(v,\ u)\big) + w(v,\ u)\}\right\} \qquad (*)$$

有关$\mathrm{dist}_k(u,\ i)$的计算公式有如下解释：$\mathrm{dist}_k(u,\ i)$有两种可能，第一，k值相对较大，稍小一点也没关系，这时存在$\mathrm{dist}_k(u,\ i) = \mathrm{dist}_{k-1}(u,\ i)$，另一种是$k$很合适，再小一点都会无路可走。至于后者，要在和节点u相邻的节点中，寻找$\mathrm{dist}_{k-1}\big(v,\ i - I(v,\ u)\big) + w(v,\ u)$最小的一个。与经典的矩阵连乘式的最优步骤计算问题类似，该优化问题也具有明显的最优子结构，所以该问题可以用动态规划思想来设计算法。

如果要输出回路，重点要关注$(*)$式中的$w(v,\ u)$这一项，这是令算法推进的本质步骤，因为只有这个选项引入了新的节点。

记上述计算过程为COMPUTE($\mathrm{dist}_k(u,\ i)$)，该符号还表示把计算的结果、计算的来源$(v,\ u)$存储在节点中。

Algorithm 单路径限制性插点路由问题，记此算法为SCSP

Input $G=(V,\ E;\ w)$，$(s,\ t;\ d)$，$B,\ d\in\mathbb{Z}_0^+$

Output 最优插点数目i^*以及与之相关的路径$P^*_{s,\,t}$，记为$(i^*,\ P^*_{s,\,t})=\mathrm{SCSP}(G;\ s,\ t;\ B,\ d)$

Begin

Step 1

// 用关联存储结构存储每个节点的信息

dict $L=\emptyset$

for each vertex $u\in G$:

$(Q_{s,\,u},\ Q'_{s,\,u})=\mathrm{HSCSP}(G;\ s,\ u)$

$L.\mathbf{append}(u,\ |Q_{s,\,u}|)$

Step 2

for $k=1$ **through** $n-1$:

for each node $u\in V$:

for $i=I(Q_{s,\,u})-(n-1)$ **through** $I(Q_{s,\,u})$:

$\mathrm{COMPUTE}(\mathrm{dist}_k(u,\ i))$

Step 3

$i^*=\mathbf{min}\{i\mid \mathrm{dist}_n(t,\ i)\le B\}$

Step 4

edge $temp=t.\mathbf{src}$

set $S=\emptyset$

$S.\mathbf{add}(temp)$

// src方法可以求得来源边

// edge.anotherVertex(a)方法可以求边的另一个非a顶点

while s $\notin temp.\mathbf{src}$:

vertex $tempUp=temp.\mathbf{src}.\mathbf{anotherVertex}(temp)$

$temp=tempUp.\mathbf{src}$

$S.\mathbf{add}(temp)$

$P^*_{s,\,t}=G(S)$

End

本算法涉及到用动态规划来降低存储复杂度和计算的时间复杂度，因此算法要始终存储每个节点针对每个$k,\ i$的dist数值，同时要存储计算出这个数值的来源，即$(v,\ u)$。具体到某种语言编程，该计算需要预先构建一个表格，通过查表的方式来解决计算问题。

五、 程序代码

六、 参考文献

[1] LI J, LI W, LICHEN J. The subdivision-constrained routing requests problem [J]. Journal of Combinatorial Optimization, 2014, 27(1): 152-63.

[2] 王涛, 李伟生. 最短路径子图 [J]. 北方交通大学学报, 2004, 02): 46-9.

[3] 孙强, 沈建华, 顾君忠. Dijkstra 的一种改进算法 [J]. 计算机工程与应用, 2002, 38(3): 99-101.