

# 求带多个限制条件的单源多权最短路径算法

戴树贵<sup>1,2</sup> 潘荫荣<sup>1</sup> 胡幼华<sup>1</sup> 孙 强<sup>1</sup>

<sup>1</sup>( 华东师范大学计算机科学与技术系 上海 200062 )

<sup>2</sup>( 滁州师范专科学校数学与计算机科学系 滁州 239000 )

**摘 要** 带限制条件的多权最短路径问题具有广泛的应用。本文给出一个通过按字典序生成从源顶点到目标顶点的非支配路径的方法,求出满足限制条件的最短路径的算法,并且分析了算法的时间复杂度。

**关键词** 最短路径 评价函数 非支配路径 时间复杂度

## AN ALGORITHM FOR THE MULTIPLE WEIGHTS SHORTEST-PATH PROBLEM WITH MULTIPLE CONSTRAINTS

Dai Shugui<sup>1,2</sup> Pan Yinrong<sup>1</sup> Hu Youhua<sup>1</sup> Sun Qiang<sup>1</sup>

<sup>1</sup>( Department of Computer Science and Technology ,East China Normal University ,Shanghai 200062 )

<sup>2</sup>( Department of Mathematics and Computer Science ,Chuzhou Normal College ,Chuzhou 239000 )

**Abstract** The shortest-path problem with constraints is widely practical. In this paper an algorithm of the nondominated path from the source point to the sink point at dictionary order is presented to get the shortest path with the constraints. The time complexity of the algorithm is also discussed.

**Keywords** Shortest path Valuation function Nondominated path Time complexity

## 0 引 言

最短路径问题是许多领域内经常出现的一类问题,如在交通运输和网络通信的研究中所遇到的许多问题经转化后都可以归结为最短路径问题。传统的最短路径问题是在有向图中求解从一个顶点到其它各顶点的最短路径,这称为单源最短路径问题。对于每条边上仅有一个非负权的情况, E.W. Dijkstra 于 1959 年给出了一个具有较好时间复杂度的算法<sup>[1]</sup>,其后,针对不同的应用,又产生了许多解决这一问题的算法<sup>[2]</sup>。然而从现实生活中的相当一部分问题抽象出的数学模型是:有向图中每条边上的权不是一个而是多个,并且要求从源顶点到某个顶点的最短路径中的某些权必须满足一定的限制条件,这就成了带限制条件的最短路径问题<sup>[3~6]</sup>。本文针对这一类问题给出了一个数学模型及其实现算法。

## 1 模 型

**定义 1.1** 设有向图  $G=(V,E)$ ,其中  $V=\{v_i | 1 \leq i \leq n, n$  为顶点数 $\}$ ,记  $V(G)$  为图  $G$  的顶点集; $E=\{e_i | e_i = \langle v_p, v_q \rangle, v_p, v_q \in V\}$ ,记  $E(G)$  为图  $G$  的有向边集, $w_1, w_2, \dots, w_h$  是图  $G$  的任一弧上的权,且  $w_i \geq 0 (i = 1, 2, \dots, h)$ 。

**定义 1.2** 令  $val = \sum_{k=1}^h w_k(e)$  为图  $G$  的评价函数,其中  $e \in E(G)$ ,且  $\sum_{k=1}^h w_k \geq 0$ ,并假定图  $G$  中任一有向边上的评价函数相同。

**定义 1.3** 设顶点序列  $s = v_{i_1}, v_{i_2}, \dots, v_{i_y} = t$  是从  $s$  到  $t$  的一条路径,若该路径上所有有向边上的权  $w_{j_1}, w_{j_2}, \dots, w_{j_x}$  分别满足  $\sum_{k=1}^x w_{j_k} \leq l_k (k = 1, \dots, x)$ ,其中  $l_k$  为给定的正常数,则称这一

路径为满足对权  $w_{j_1}, w_{j_2}, \dots, w_{j_x}$  限制的路径。

**定称 1.4** 设  $p_1, p_2, \dots, p_m$  是从  $s$  到  $t$  的满足对权  $w_{j_1}, w_{j_2}, \dots, w_{j_x}$  限制的路径,且  $\sum val_k (k = 1, 2, \dots, m)$  分别为各条路径的评价函数和,则称  $\min(\sum val_1, \sum val_2, \dots, \sum val_m)$  为从  $s$  到  $t$  的满足对权  $w_{j_1}, w_{j_2}, \dots, w_{j_x}$  限制的最短路径。

**定称 1.5** 一条从  $s$  到  $t$  的路径  $p_s = \{s, \langle s, a_1 \rangle, \langle a_1, a_2 \rangle, \dots, \langle a_u, a_t \rangle, a_t\}$  定义:

$$val(p_s) = \sum val$$

$$w_j(p_s) = \sum w_j(e) (e \in p_s, w_j \text{ 为受限制权})$$

## 2 定义及性质

首先定义如下不等式:

$$\begin{aligned} (x_1, x_2, \dots, x_q) \leq (y_1, y_2, \dots, y_q) &\leftrightarrow \forall i: x_i \leq y_i (i = 1, \dots, q) \\ (x_1, x_2, \dots, x_q) < (y_1, y_2, \dots, y_q) &\leftrightarrow \forall i: x_i \leq y_i \text{ and } \exists i: x_i < y_i (i = 1, \dots, q) \end{aligned}$$

$$(x_1, x_2, \dots, x_q) \neq (y_1, y_2, \dots, y_q) \leftrightarrow x_1 < y_1 \text{ or } \forall j \leq i: x_j = y_j \text{ and } x_{i+1} < y_{i+1} (i = 2, \dots, q-1)$$

$$(x_1, x_2, \dots, x_q) = (y_1, y_2, \dots, y_q) \leftrightarrow \forall i: x_i = y_i (i = 1, \dots, q)$$

**定义 2.1** 设  $p_1$  是图  $G$  中一条从  $s$  到  $t$  的路径,如果图  $G$  中不存在路径  $p_2$  使:

$$(val(p_2), w_{j_1}(p_2), \dots, w_{j_x}(p_2)) < (val(p_1), w_{j_1}(p_1), \dots, w_{j_x}(p_1))$$

则称  $p_1$  是图  $G$  中一条从  $s$  到  $t$  的非支配路径,否则,称之为支配路径。如果图  $G$  中两条从  $s$  到  $t$  的路径  $p_1, p_2$  满足:

$(val(p_1), w_{j_1}(p_1), \dots, w_{j_k}(p_1)) = (val(p_2), w_{j_1}(p_2), \dots, w_{j_k}(p_2))$   
则称  $p_1, p_2$  等价。

**定义 2.2** 设  $p_1, p_2$  为两条从  $s$  到  $t$  的路径, 若  
 $(val(p_1), w_{j_1}(p_1), \dots, w_{j_k}(p_1)) \pi (val(p_2), w_{j_1}(p_2), \dots, w_{j_k}(p_2))$   
则称  $p_1$  字典序优于  $p_2$ 。

**定义 2.3** 设  $p$  为从  $s$  到  $t$  的一条路径, 若不存在路径字典序优于  $p$ , 则称  $p$  为从  $s$  到  $t$  按字典序最短路径。

**定理 2.1** 如果  $p_{st}$  是从  $s$  到  $t$  的非支配路径, 则对任何顶点  $k \in p_{st} (k \neq s, t)$ ,  $p_{st}$  的子路径  $p_{sk} (\subset p_{st})$  是从  $s$  到  $k$  的非支配路径。

**定理 2.2** 如果图  $G$  中存在一条从  $s$  到  $t$  的路径满足:  $\sum w_i \leq l_i (i = j_1, \dots, j_k)$ , 则图  $G$  中存在一条从  $s$  到  $t$  的非支配路径是所求最短路径。

### 3 算法

#### 3.1 算法思想

为了确定所需求解的最短路径, 在算法中对每个顶点  $j$  赋予若干个标号, 这些标号可分为临时标号和永久标号两类, 顶点  $j$  的永久标号集和临时标号集分别以  $R_j$  和  $S_j$  表示, 临时标号有待修改, 而永久标号不再改变, 某个顶点的一个永久标号将对应于一条从源顶点到该顶点的非支配路径。

本算法使用由源顶点开始, 按字典序由小到大的顺序逐渐生成一条从源顶点到目标顶点的非支配路径的方法, 求得从源顶点到目标顶点的受限制最短路径。在算法执行过程中, 可以确定某些受限制路径明显不存在的情况, 但不能排除满足限制条件的路径不存在的所有情况。因此在算法执行过程中需要进行满足限制的路径是否存在的判断, 如果当目标顶点还没有永久标记之前, 所有的临时标号集均为空, 即已经没有可以转化为永久标号的临时标号了, 则满足限制条件的路径不存在。如果目标顶点最终被永久标记, 则从源顶点到目标顶点满足限制条件的路径存在。

由于在具体实现的时候, 受限制权的数目总是有限的, 为了描述的方便, 在下面的 3.2 和 3.3 节中均以有两个受限制权  $w_i$  和  $w_j$  的问题为例, 分别给出算法所使用的数据结构和算法的具体实现过程。

#### 3.2 算法的数据结构

定义一个七元组  $(val, wi, wj, j, k, i, u)$  结构, 同时使用指针将同一顶点的永久标号或临时标号连接起来:

```
struct node{float val;  
    int wi, wj;  
    int j, k, i, u;  
    struct node * next;};
```

其中:  $val$  表示该标号所对应的路径的评价函数之和;  $wi$  和  $wj$  分别表示该标号所对应的路径的受限制权  $w_i$  和  $w_j$  之和;  $j$  表示该标号所对应的顶点编号;  $k$  表示该标号所对应的路径是从源顶点到顶点  $j$  的第  $k$  条路径;  $i$  表示该标号所对应的路径是由顶点  $i$  经由有向边  $\langle i, j \rangle$  到达顶点  $j$  的;  $u$  表示该标号所对应的路径是经由顶点  $i$  的第  $u$  个标号而来的;  $next$  用于指向下一个临时标号或永久标号。

这里  $j, k, i, u$  均是为了在算法找到满足限制条件的最短路径结束后, 能够从目标顶点到源顶点回找该条最短路径。若不需要寻找该条最短路径, 则  $j, k, i, u$  可以一起省略, 这样, 算

法就只能给出最短路径的评价函数以及受限制权  $w_i$  和  $w_j$  的值。

由于程序设计语言一般不支持结构类型数据的集合, 在此使用链表结构来实现永久标号集和临时标号集, 并且为了实现方便, 在表示临时标号集链表的头结点引入一个标号的顺序号  $index$ , 数据结构定义如下:

```
struct set{unsigned index;  
    struct node * head;};
```

其中,  $index$  用于存放该顶点对应的下一个临时标号的  $k$  值,  $head$  为该顶点的标号集的头指针。

定义永久标号集和临时标号集如下:

```
struct node * R[n];  
struct set S[n];
```

其中  $R[j]$  用于存放顶点  $j$  所对应的永久标号集;  $S[j]$  用于存放顶点  $j$  所对应的临时标号集。

假设 0 为源顶点,  $n-1$  为目标顶点, 则  $R[n]$  和  $S[n]$  的结构图分别如图 1 和 2 所示。

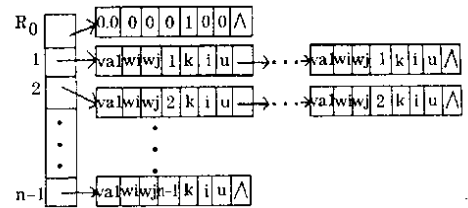


图 1 永久标号集结构示意图

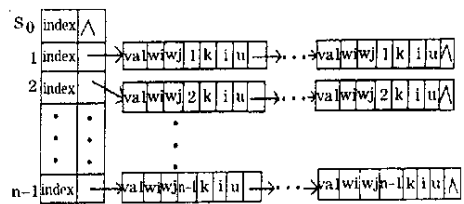


图 2 临时标号集结构示意图

为了存放评价函数和受限制权定义如下三个数组:

```
float valma[n][n];  
int w[n][n], wj[n][n];
```

其中:  $valma[i][j]$  为有向边  $\langle i, j \rangle$  的评价函数值;  $wi[i][j]$  和  $wj[i][j]$  分别为有向边  $\langle i, j \rangle$  的受限制权  $w_i$  和  $w_j$  的值。

同时为了在算法执行过程中, 当发现满足限制条件的路径不存在时, 可以立即结束, 引入无解标志  $nofound$ , 其初值为 0。

#### 3.3 算法

以下给出求解从源顶点  $s$  到目标顶点  $t$  受权  $w_i$  和  $w_j$  限制的最短路径算法:

(1) 分别以受限制权  $w_i$  和  $w_j$  为权求各个顶点  $j$  到目标顶点  $t$  的单权最短路径值, 并分别记为  $wi(j, t)$  和  $wj(j, t)$ , 若  $wi(s, t) > l_i$  或  $wj(s, t) > l_j$ , 则满足对权  $w_i$  和  $w_j$  限制的路径不存在, 该问题无解。  $nofound \leftarrow 1$  转 (13);

(2) 置  $S[s].head$  的值为  $(0, 0, 0, 0, s, 1, 0, 0, null)$ ;

(3) 对于所有顶点  $j \neq s$ , 置  $S[j].index$  的值为 1;  $S[j].head$  的值为  $null$ ;

(4) 若所有  $S[j].head$  为空, 则满足限制条件的路径不存在,  $nofound \leftarrow 1$  转 (13); 否则, 求所有顶点的临时标号集中按  $val$ ,

$w_i, w_j$  字典序最小的结点, 置于  $C_{min}$  中;

(5) 若  $C_{min}$  中  $j$  的值为  $i$ , 则从  $S[i].head$  中删除第一结点, 并将该结点插入  $R[i].head$  中;

(6) 若  $R[t].head \neq null$  转(13);

(7) 若存在没有作如下处理的顶点  $j \in V(G) \setminus \{i, j\} \in E(G)$  则计算:

$\{val_j \leftarrow val + valma[i][j];$  其中  $val$  为  $C_{min}$  中的  $val$ ;

$wij \leftarrow wi + wi[i][j]; w_{ij} \leftarrow wj[i][j];$  其中  $wi$  和  $wj$  分别为  $C_{min}$  中的  $wi$  和  $wj$  否则转(4);

(8) 若  $wij + w(j, t) > l_i$  或  $w_{ij} + w(j, t) > l_j$  转(12);

(9) 若  $R[j].head$  和  $S[j].head$  中存在其个结点(  $val, wi, wj, \dots$  ), 使(  $val, wi, wj$  )  $\leq$  (  $val_j, wij, w_{ij}$  ) 则转(12);

(10) 若在  $S[j].head$  中存在结点(  $val, wi, wj, \dots$  ), 满足(  $val_j, wij, w_{ij}$  )  $<$  (  $val, wi, wj$  ), 则从  $S[j].head$  删除这些结点; 并在  $S[j].head$  中插入结点(  $val_j, wij, w_{ij}, j, S[j].index, i, u, null$  ) 其中  $u$  为  $C_{min}$  中的  $k$  值, 下同), 并将  $S[j].index$  加 1, 转(12);

(11) 在  $S[j].head$  中插入结点(  $val_j, wij, w_{ij}, j, S[j].index, i, u, null$  ), 并将  $S[j].index$  加 1;

(12) 转(7);

(13) 若  $nofound = 1$  则没有满足对权  $wi$  和  $wj$  限制的路径存在, 否则满足对权  $wi$  和  $wj$  限制的路径存在, 用下述方法, 回找一条由源顶点  $s$  到目标顶点  $t$  的满足对权  $wi$  和  $wj$  限制的最短路径。

先由  $R[t].head$  的  $i$  和  $u$ , 找到其前一个结点  $i$  及  $R[i].head$  中满足  $k = u$  的结点, 再由该结点的  $i$  和  $u$  找到其前一个结点  $i, \dots$  这样一直执行下去, 直到某次所得到的结点中的  $i$  等于  $s$ 。这样所得到的顶点序列的倒序即为由源顶点  $s$  到目标顶点  $t$  满足对权  $wi$  和  $wj$  限制的最短路径所对应的顶点序列, 这个最短路径的评价函数之和为  $R[t].head$  的  $val$  值, 受限制权  $wi$  和  $wj$  之和分别为  $R[t].head$  中的  $wi$  和  $wj$  的值。

#### 4 算法的可行性证明

**定理 1** 若(  $val_j, w_{1j}, \dots, w_{xj}$  )  $\in R[j]$ , 则永久标号(  $val_j, w_{1j}, \dots, w_{xj}$  )对应了图  $G$  中一条从源顶点  $s$  到顶点  $j$  非支配路径  $P_{sj}$ , 且  $val(p_{sj}) = val_j, w_{j1}(p_{sj}) = w_{1j}, \dots, w_{jx}(p_{sj}) = w_{xj} \leq l_x$ 。

**定理 2** 假设(  $val_j, w_{1j}, \dots, w_{xj}$  )是从  $S[j]$  中第一个被确定到  $R[j]$  中的永久标号, 则(  $val_j, w_{1j}, \dots, w_{xj}$  )对应的非支配路径必定是从源顶点  $s$  到顶点  $j$  的受权  $w_{j1}, w_{j2}, \dots, w_{jx}$  限制的最短路径。

**定理 3** (  $val_t, w_{1t}, \dots, w_{xt}$  )  $\in R[t]$ , 则(  $val_t, w_{1t}, \dots, w_{xt}$  )对应的非支配路径是从源顶点  $s$  到目标顶点  $t$  的满足对权  $w_{j1}, \dots, w_{jx}$  限制的路径, 其评价函数之和为  $val_t$ , 受限制权  $w_{j1}, w_{j2}, \dots, w_{jx}$  的和分别为  $w_{j1t}, \dots, w_{jxt}$ 。

#### 5 实例及性能分析

如图 3 所示的有向图中, 每条有向边上的三元组  $w_1, w_2, w_3$  为该边的三个权值, 其中  $w_2$  和  $w_3$  为受限制权, 下面对于给定的限制条件, 给出了算法的执行结果, 并对算法在最坏情况下的时间复杂度进行了分析。(定义评价函数  $val = k_1 \times w_1 + k_2 \times w_2 + k_3 \times w_3$ , 其中  $k_1 = 0.6, k_2 = 0.3, k_3 = 0.1$ )

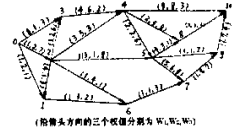


图 3 带两个限制条件的多权网络

算法首先求各顶点到目标顶点 10 的以权  $w_2$  和  $w_3$  为权的单权最短路径值, 如下:

$w_2(0, 10) = 7$	$w_3(0, 10) = 6$	$w_2(1, 10) = 20$
$w_3(1, 10) = 16$	$w_2(2, 10) = 5$	$w_3(2, 10) = 6$
$w_2(3, 10) = 12$	$w_3(3, 10) = 5$	$w_2(4, 10) = 6$
$w_3(4, 10) = 3$	$w_2(5, 10) = 4$	$w_3(5, 10) = 5$
$w_2(6, 10) = 17$	$w_3(6, 10) = 14$	$w_2(7, 10) = 16$
$w_3(7, 10) = 11$	$w_2(8, 10) = 1$	$w_3(8, 10) = 4$
$w_2(9, 10) = 7$	$w_3(9, 10) = 4$	$w_2(10, 10) = 0$
$w_3(10, 10) = 0$		

##### 5.1 实例分析

求权  $w_2$  满足  $\sum w_2 \leq 20$  且权  $w_3$  满足  $\sum w_3 \leq 9$  限制时, 顶点 0 到顶点 10 间的受限制最短路径运行结果如下表所示:

$w_2$	$w_3$	$val$	$j$	$k$	$i$	$u$
0	0	0.00	0	1	0	0
1	3	5.40	3	1	0	1
7	5	9.80	4	1	3	1
15	8	17.30	1	1	4	1

由上表可知, 从顶点 0 到顶点 10 间的  $w_2$  满足  $\sum w_2 \leq 20$  且  $w_3$  满足  $\sum w_3 \leq 9$  的受限制最短路径为(图中黑体标明了路标):  $0 \rightarrow 3 \rightarrow 4 \rightarrow 10$ ; 该路径上权  $w_2, w_3$  和评价函数  $val$  的值分别为 15、8 和 17.30。

##### 5.2 时间复杂度分析

在受限制权和它们的限制值均为整数的情况下, 一个顶点的最大标号为  $l_1 \times l_2 \times \dots \times l_x$  (记为  $\Pi l_i$ ), 所有顶点的永久标号集和临时标号集中至多有  $(n-1)\Pi l_i$  个标号, 算法的第(4)~(12)步每循环一次确定一个永久标号, 因此该算法最多重复执行  $(n-1)\Pi l_i$  次即结束;

在(1)中, 由 Dijkstra 算法的时间复杂度分析可知, 该步骤的时间复杂度为  $O(xn^2)$ 。

在(4)中, 要取各临时标号集中按(  $val_j, w_{1j}, \dots, w_{xj}$  )字典序最小的标号, 则由于在随后的临时标号插入时, 都是按(  $val_j, w_{1j}, \dots, w_{xj}$  )字典序进行的, 因此可以依次取每个  $S_j (j = 0, 1, \dots, n-1)$  的第一个元素 (即按字典序最小的元素) 进行比较, 从中按字典序选出最小者, 这样, 该步骤至多需要  $(x+1)(n-1)$  次比较。

在(7)~(12)的循环中, 由于每个顶点的出度至多为  $(n-1)$ , 因此, 首先至多进行  $(x+1)(n-1)$  次加法, 然后在(8)中最多执行  $x(n-1)$  次比较, 在(9)中由于顶点  $j$  的永久标号集和临时标号集中至多有  $\Pi l_i$  个标号, 因此在其中查找某个标号, 则至多需要  $(x+1)\Pi l_i$  次比较, 同样, 在顶点  $j$  的临时标号集中, 至多有  $\Pi l_i$  个临时标号, 将一个标号按字典序插入顶点  $j$  的临时标号集中至多需要  $(x+1)\Pi l_i$  次比较, 故在(10)和(11)中每次把一个新的临时标号按字典序插入到顶点  $j$  的临时标号集中至多需  $(x+1)\Pi l_i$  次操作, 每循环一次至多产生  $(n-1)$  个新的临时标

号。因此该步骤每循环一次共需  $n(x+1)\Pi l_i$  次比较就可以把新的临时标号按字典序插入到顶点  $j$  的临时标号集中。所以对于每次循环  $(7) \sim (12)$  最大时间复杂度为  $O(nx\Pi l_i)$ ;

在 (13) 中,由于一条路径至多有  $n$  个顶点,而每个顶点至多有  $\Pi l_i$  个标号,则在每个顶点的标号集中找到指定的标号,需要  $(x+1)\Pi l_i$  次比较,因而从目标点到源点回找所求得的最佳解最多需要  $(n-2)(x+1)\Pi l_i$  次比较(因为对于源点和目标点并不需要查找)。

因此,该算法的时间复杂度为  $O(n^2x\Pi l_i^2)$ 。

6 结束语

本文给出一个通过按(评价函数,受限制权 1,受限制权 2,...)字典序生成从源顶点到目标顶点的非支配路径的方法,求出满足多个限制条件的最短路径的算法,算法的理论时间复杂度为  $O(n^2x\Pi l_i^2)$ 。但是由于一般情况下顶点的最大出度  $k$  远小于顶点数  $n$ ,同时受限制权的数量  $x$  也是一个较小的量,所以可以认为该算法的时间复杂度为  $\max\{O(xn^2),O(n\Pi l_i^2)\}$ 。和传统的通过求解第  $k$  短路径以解决该类问题的方法相比,当受限制权数较少时,算法具有较小的时间复杂度,是一个可以接受的实用算法。

参 考 文 献

[1] E. W. Dijkstra, A note on Two Problem in connexion with graphs, Numeric Mathematics 1, 269 ~ 271( 1959 ).  
[2] Narshingh Deo and Chi-yin Pang. Shortest-Path Algorithms :Taxonomy and Annotation. Networks Vol. 14( 1984 ), 275 ~ 323.  
[3] H. C. Joksch. The shortest Route Problem with Constraints. Journal of Mathematical Analysis and Applications 14( 1966 ), 191 ~ 197.  
[4] Mohsen M. D. Hassan. Network Reduction for the Acyclic Constrained Shortest Path Problem. European Journal of Operational Research. 63( 1992 ), 124 ~ 132.  
[5] Gabriel Y. Handler and Israel Zang. A Dual Algorithm for the Constrained Shortest Path Problem. Network Vol. 10( 1980 ), 293 ~ 310.  
[6] 周经伦、吴唤群,“受顶点数限制的最短路径问题及其算法[J]”《系统工程》,第 14 卷第 5 期( 1996 ), 37 ~ 44.

(上接第 24 页)

```
Hour _ Per _ Student    dec( 4 , 1 ) ,
Primary key( OID )
Foreign key( OID )reference PERSON
On delete cascade
On update restrict );
```

```
·Insert :
Person :    与重叠单继承相同
Faculty :   与重叠单继承相同
Student :   与重叠单继承相同
Faculty _ Student :
Insert Into PERSON
Values( &newID &newName &newAddress );
Insert Into FACULTY
Values( &newID &newWage );
Insert Into STUDENT
Values( &newID &newInto _ Score );
```

```
Insert Into FACULTY _ STUDENT
Values( &newID &newHour _ Per _ Week );
·Delete :
Person :    与重叠单继承相同
Faculty :   与重叠单继承相同
Student :   与重叠单继承相同
Faculty _ Student :
Delete form PERSON
Where PERSON. ID = &newfaculty _ student _ ID ;
```

PERSON

OID	Name	Address
1	李杰辉	江岸小区 45 幢 408
2	方 雄	白马小区 145 幢 101
3	蒋云研	阳光小区 5 幢 508
4	许小东	安吉花园 32 幢 709
5	李燕芦	大学花园 7 幢 213
6	代顺达	友爱小区 8 幢 408
7	冯秋华	朝阳花园 1 幢 203
8	林凤桥	江周路 42 号

FACULTY

OID	Wage
1	1056.8
3	2006.7
6	1978.5

STUDENT

OID	Into _ Score
2	658
4	763.5
5	780
6	698
7	709

FACULTY \_ STUDENT

OID	Hour _ Per _ Week
6	34

关系模式 FACULTY \_ STUDENT 反映的是 FACULTY \_ STUDENT 类的基类是 FACULTY 类和 STUDENT 类。由于关系数据库的参照完整性只反映两个表之间的参照关系,在表 FACULTY \_ STUDENT 的 Create 语句中,我们把 PERSON 表作为其参照表。向表 FACULTY \_ STUDENT 插入记录必须遵循一定的顺序,即先从根表 PERSON 开始,然后是 FACULTY 表和 STUDENT 表,最后才是 FACULTY \_ STUDENT 表。


4 结束语

由于问题域的多样性和复杂性,对继承关系转换为关系表没有一种统一的方法。本文描述了面向对象概念模型中的继承关系向关系表转换的一种方法。在该方法中,我们把单继承划为重叠单继承、非相交单继承和划分单继承,并且给出了各种继承关系转换为关系表的 SQL 语句实现。我们的转换方法既考虑了不同类层次之间的关系,又考虑了实现的可能性,该转换方法是对现有转换方法有益的补充。

参 考 文 献

[1] G. Booch, J. Rumbaugh, J. Jacobson, The Unified Modeling Language User Guide, Addison Wesley.  
[2] Y. Masunaga, in :W. Kim( ED. ), Object Identity, Equality and Relational Concept, Deductive and Object \_ Oriented Databases, 1990, pp. 185 ~ 202.  
[3] N. Black, Database Optimisation Techniques Microsoft Tech \_ Ed 95[ for Microsoft Access V2.0 ], Microsoft Corporation, 1995.  
[4] P. Misshra, M. H. Eich, Join processing in relational databases, ACM Comput. Surv., 1992, 24( 1 ) 63 ~ 113.  
[5] (美)兰博等著,姚淑珍等译, UML 参考手册,北京 :机械工业出版社, 2001.1( Rational 技术丛书 ).  
[6] 丁宝康、董健全、施伯乐,数据库实用教程,北京 :清华大学出版社, 2001.

# 求带多个限制条件的单源多权最短路径算法

作者：[戴树贵](#)，[潘荫荣](#)，[胡幼华](#)，[孙强](#)  
作者单位：[戴树贵\(华东师范大学计算机科学与技术系, 上海, 200062; 滁州师范专科学校数学与计算机科学系, 滁州, 239000\)](#)，[潘荫荣, 胡幼华, 孙强\(华东师范大学计算机科学与技术系, 上海, 200062\)](#)  
刊名：[计算机应用与软件](#)   
英文刊名：[COMPUTER APPLICATIONS AND SOFTWARE](#)  
年，卷(期)：2004, 21(12)  
被引用次数：10次

## 参考文献(6条)

1. [E W Dijkstra](#) [A note on Two Problem in connexion with graphs](#) 1959
2. [Narshingh Deo; Chi-yin Pang](#) [Shortest-Path Algorithms: Taxonomy and Annotation](#) 1984
3. [H C Joksch](#) [The shortest Route Problem with Constraints](#) 1966
4. [Mohsen M; D. Hassan](#) [Network Reduction for the Acyclic Constrained Shortest Path Problem](#) 1992
5. [Gabriel Y; Handler; Israel Zang](#) [A Dual Algorithm for the Constrained Shortest Path Problem](#) 1980
6. [周纶伦; 吴唤群](#) [受顶点数限制的最短路径问题及其算法](#) 1996(05)

## 本文读者也读过(3条)

1. [彭中](#) [网络图的计算机算法和显示方法的研究](#)[学位论文] 2004
2. [冯德民. 谢娟英. FENG De-min. XIE Juan-ying](#) [带单一限制条件的单源多权最短路径算法及其实现](#)[期刊论文]-[西南师范大学学报\(自然科学版\)](#) 2000, 25(3)
3. [戴树贵. 孙强. 潘荫荣](#) [带限制条件的多权最短路径算法](#)[期刊论文]-[上饶师范学院学报](#) 2002, 22(3)

## 引证文献(10条)

1. [邓礼礼, 孙强](#) [求图中顶点之间所有最短路径的一种算法](#)[期刊论文]-[计算机应用与软件](#) 2009(10)
2. [李汉轩, 李志华, 吕春生](#) [一种多约束条件下路径规划算法研究](#)[期刊论文]-[电子设计工程](#) 2012(10)
3. [徐盛, 雷定猷, 张英贵](#) [超限超重货物运输路径决策模型和算法](#)[期刊论文]-[铁道货运](#) 2009(08)
4. [王卫强, 孙强](#) [求图中受顶点数限制的所有最短路径的算法](#)[期刊论文]-[计算机工程与设计](#) 2008(07)
5. [况超](#) [求图中每对顶点间的所有最短路径算法的分析与研究](#)[学位论文] 硕士 2010
6. [何建军, 王丽芳](#) [求所有最小点成本最短路径算法](#)[期刊论文]-[软件导刊](#) 2015(04)
7. [吴先锋](#) [基于结构元理论的路径综合优化矩阵算法研究](#)[学位论文] 硕士 2012
8. [王卫强](#) [求图中受顶点数限制的所有最短路径的算法分析研究](#)[学位论文] 硕士 2007
9. [徐盛](#) [超限超重货物运输管理系统的设计与开发](#)[学位论文] 硕士 2009
10. [裴岩](#) [机器学习理论研究及其在车载导航系统中的应用](#)[学位论文] 硕士 2008

引用本文格式：[戴树贵. 潘荫荣. 胡幼华. 孙强](#) [求带多个限制条件的单源多权最短路径算法](#)[期刊论文]-[计算机应用与软件](#) 2004(12)