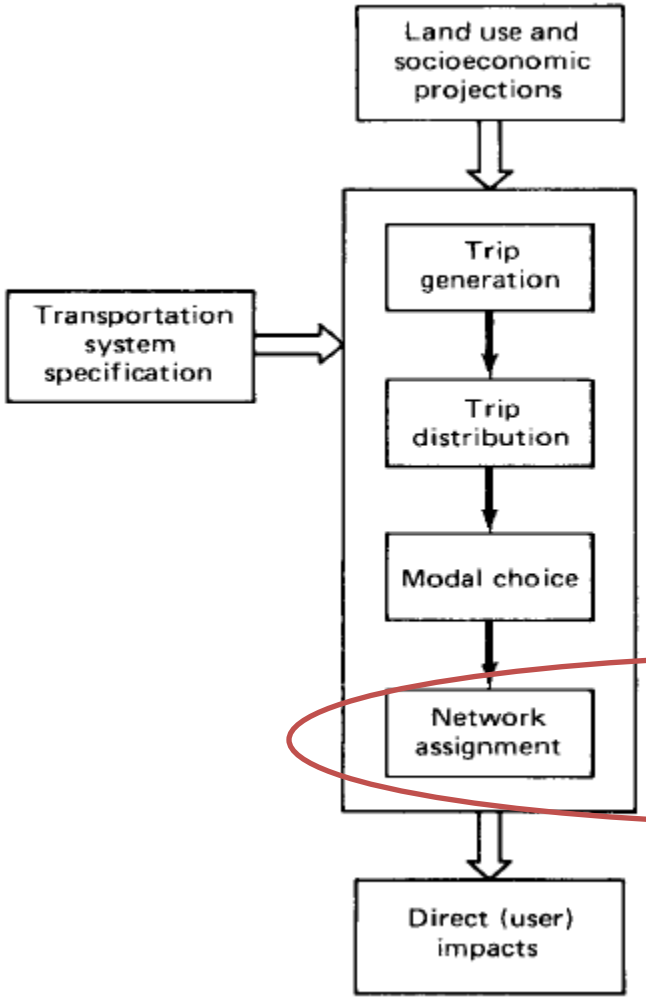
交通流分配问题
Traffic assignment

内容

1. 交通流分配问题的定义
2. 交通流分配问题的特点
3. 用户均衡
4. 交通流分配算法
   - All-or-Nothing traffic assignment
   - Capacity constraint traffic assignment
   - Incremental assignment
   - Beckmann’s transformation

交通规划四阶段法（4-step method）



需求生成→ 每个区域的出行需求量

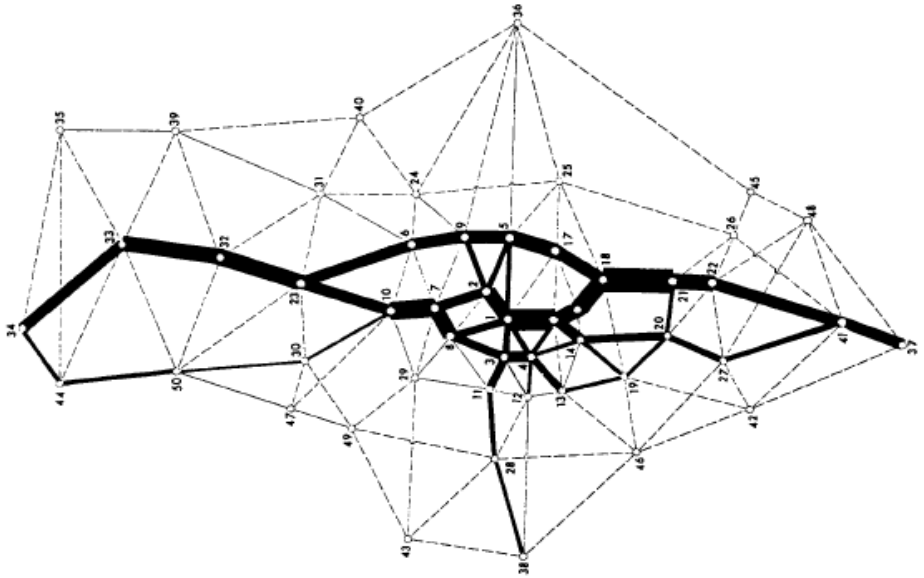
需求分配→每个OD的出行需求量

模式选择→每个OD不同模式的出行需求量

交通流分配→每条路段的流量

交通流分配问题的定义

交通流分配问题就是基于给定的路网结构、路段通行成本以及每个OD的小汽车出行需求量，确定每个路段上的小汽车流量和整个路网的通行效率。



交通流分配是交通网络设计、新建设施的交通影响分析、交通需求管理机制设计等交通规划问题的基础。

交通流分配问题的输入和输出

输入：

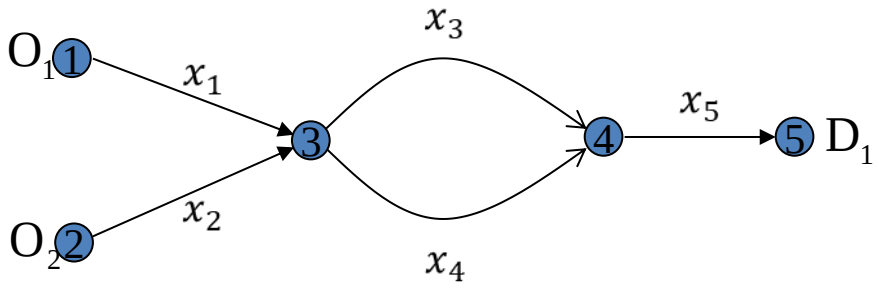
1. 路网的结构
2. 每个路段上的通行成本函数
3. 每个Origin-Destination（OD）间的出行需求

输出：

1. 每条路段上的流量和通行成本
2. 路网上总的通行时间
3. （每个OD间不同路径的流量和通行成本）

交通流分配问题的特点

路网上的流量分布是每个用户独立决策路径选择的结果



如何将每个OD间总的出行需求分配到不同的路径?

交通流分配的两个基本准则

- Two principals proposed by Wardrop:
  - First principal: User equilibrium (UE)用户均衡

  用户均衡描述的是当每个用户按照自己的意愿选择路径时，所达到的均衡状态的路网流量分配。

  - Second principal: System optimum (SO)系统最优

  系统最优描述的是当每个用户都按照中央指令来选择路径时，达到社会福利最大化的情况下的路网流量分配。

交通流分配的第一准则-用户均衡

用户均衡的基本假设

1. 所有用户都是理性用户，总是试图最小化自己的出行成本
2. 所有用户都对路网掌握完全信息

用户均衡状态的定义

- **Wardrop均衡的定义（Wardrop, 1952）：**

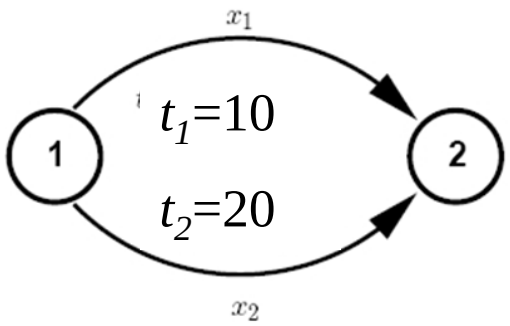
  No driver can unilaterally reduce his/her travel costs by shifting to another route.

  - 没有司机能够通过单方面改变自己的出行路线选择来减少出行成本。

- 在Wardrop均衡下，任意被使用的路径，其路径成本等于该OD间用户的最小出行成本。

例1（不考虑拥堵效应）

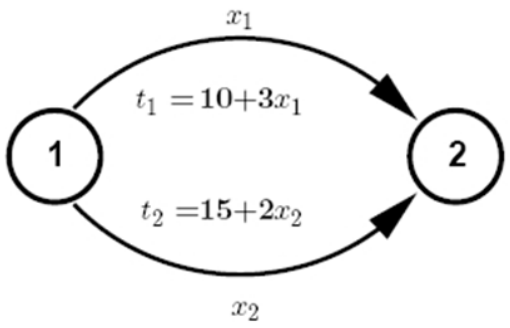
考虑如下只包含单个OD和两条平行弧的简单网络。假设节点1和2之间共存在12个单位的出行需求，每条平行弧上的通行时间分别为10和20，如下图所示。请问用户均衡状态下，两条平行弧上的流量各是多少？



$x_1 + x_2 = 12$

例2（考虑拥堵效应）

（通行时间是流量的函数）考虑如下只包含单个OD和两条平行弧的简单网络。假设节点1和2之间共存在12个单位的出行需求，每条平行弧上的通行时间是其流量的函数，如下图所示。请问用户均衡状态下，两条平行弧上的流量各是多少？



$x_1 + x_2 = 12$

三种可能的情况：

1. 两条路径都有人用$(x_1 > 0, x_2 > 0)$，此时：

$$t_1(x_1) = t_2(x_2)$$

2. 路径1没有人用，路径2有人用$(x_1 = 0, x_2 > 0)$，此时：

$$t_1(x_1) \geq t_2(x_2)$$

3. 路径1有人用，路径2没有人用$(x_1 > 0, x_2 = 0)$，此时：

$$t_1(x_1) \leq t_2(x_2)$$

例2

三种可能的情况：

1. $x_1 > 0, x_2 > 0, t_1(x_1) = t_2(x_2)$:

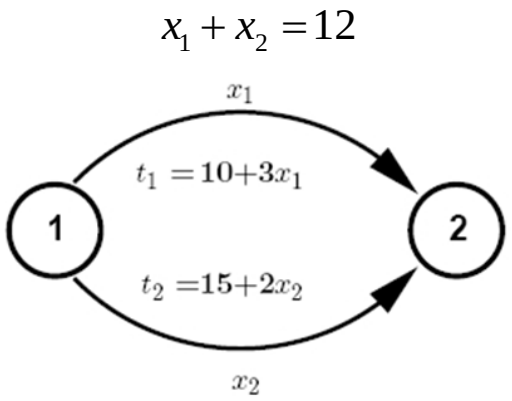
$10 + 3x_1 = 15 + 2x_2, x_1 + x_2 = 12 \quad \rightarrow x_1 = 5.8, x_2 = 6.2$ √

2. $x_1 = 0, x_2 > 0, t_1(x_1) > t_2(x_2)$:

$x_1 = 0, x_2 = 12, t_1 = 10 < t_2 = 39$ ×

3. $x_1 > 0, x_2 = 0, t_1(x_1) < t_2(x_2)$:

$x_1 = 12, x_2 = 0, t_1 = 46 > t_2 = 15$ ×



用户均衡的数学表达

数学符号:

W：OD的集合

R_w：OD $w \in W$间的路径集合

$c_{r,w}$：OD $w \in W$间路径 $r \in R_w$的通行成本

μ_w：OD $w \in W$间的最小通行成本，$\mu_w = \min\left\{c_{r,w}, r \in R_w\right\}$

$f_{r,w}$：OD $w \in W$间路径 $r \in R_w$上的流量

用户均衡条件:

If $c_{r,w} - \mu_w = 0$，then $f_{r,w} \geq 0$

If $c_{r,w} - \mu_w > 0$, then $f_{r,w} = 0$,

$$\sum_{r \in R_w} f_{r,w} = d_w, w \in W$$

$\Longleftrightarrow$

$$\left(c_{r,w} - \mu_w\right) f_{r,w} = 0,\ r \in R_w, w \in W$$

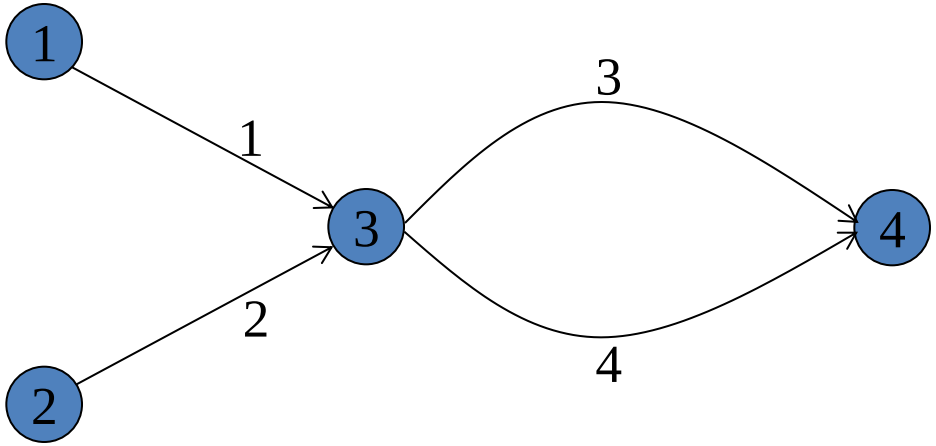
$$c_{r,w} - \mu_w \geq 0,\quad f_{r,w} \geq 0,\ r \in R_w, w \in W$$

$$\sum_{r \in R_w} f_{r,w} = d_w, w \in W$$

路径和弧之间的变量转化

- A link-path incident matrix defines the relationship between paths and links:
  - $\Delta = [\delta_{a,r}, a \in A, r \in R]$
  - $\delta_{a,r} = 1$ if path $r \in R$ traverses link $a \in A$, $\delta_{a,r} = 0$ otherwise
- Using link-path incident matrix, it is convenient to describe path costs with link costs, and describe link flows with path flows:
  - $c_{r,w} = \sum_{a \in A} \delta_{a,r} t_a(v_a), r \in R_w, w \in W$
  - $v_a = \sum_{w \in W} \sum_{r \in R_w} \delta_{a,r} f_{r,w}, a \in A$

Example



O-D trips from 1 to 4: path 1 (links 1, 3); path 2 (links 1,4)

O-D trips from 2 to 4: path 3 (links 2, 3); path 4 (links 2,4)

$$\Delta \triangleq \left(\delta_{a,r}\right) = \begin{array}{c} \\ \text{link} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} \overbrace{\text{path}}^{\text{O-D 1-4}} & & \overbrace{}^{\text{O-D 2-4}} & \\ 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ & & & \\ & & & \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

$$c_{1,1} = t_1\delta_{1,1} + t_2\delta_{2,1} + t_3\delta_{3,1} + t_4\delta_{4,1}$$
$$= t_1 + t_3$$
$$x_3 = f_{1,1}\delta_{3,1} + f_{2,1}\delta_{3,2} + f_{3,2}\delta_{3,3} + f_{4,2}\delta_{3,4}$$
$$= f_{1,1} + f_{3,2}$$

用户均衡

$$\left(c_{r,w}-\mu_w\right) f_{r,w}=0,\ \ r\in R_w, w\in W$$

$$c_{r,w}-\mu_w\geq 0,\quad f_{r,w}\geq 0,\ \ r\in R_w, w\in W$$

$$\sum_{r\in R_w} f_{r,w}=d_w, w\in W$$

$$x_a=\sum_{w\in W}\sum_{r\in R_w} f_{r,w}\delta_{a,r}, a\in A$$

$$c_{r,w}=\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r}, r\in R_w, w\in W$$

$$\left(\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r}-\mu_w\right) f_{r,w}=0,\ \ r\in R_w, w\in W$$

$$\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r}-\mu_w\geq 0,\quad f_{r,w}\geq 0,\quad r\in R_w, w\in W$$

$$\sum_{r\in R_w} f_{r,w}=d_w,\ \ w\in W$$

$$x_a=\sum_{w\in W}\sum_{r\in R_w} f_{r,w}\delta_{a,r},\quad a\in A$$

用户均衡下的交通流分配算法

- All-or-Nothing traffic assignment
- Capacity constraint traffic assignment
- Incremental assignment

启发式

- Beckmann’s tranformation

All-or-Nothing Traffic Assignment

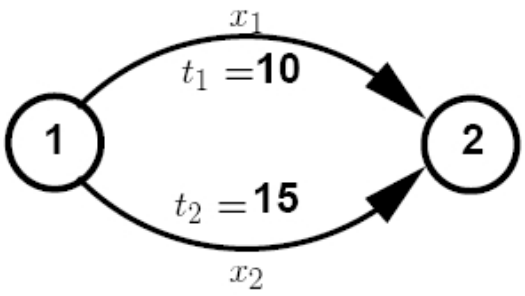
- "All-or-Nothing" is a commonly used procedure for network loading. This does not recognize the dependence between flows and travel time (travel time is assumed to be fixed).

- In this method the trips from any origin zone to any destination zone are loaded onto a single, minimum cost, path between them.

- It involve the technique of network loading- the process of assigning the O-D entries to the network for specific link travel times by following the route choice criterion.

Example

To demonstrate how this assignment works, an example network is considered. This network has two nodes having two paths as links. Let us suppose a case where travel time is not a function of flow as shown in other words it is constant as shown in the figure below.



- This model is unrealistic as only one path between every O-D pair is utilized even if there is another path with the same or nearly same travel cost;
- Traffic on links is assigned without consideration of whether or not there is adequate capacity or heavy congestion;
- Travel time is a fixed input and does not vary depending on the congestion on a link.

Capacity Constraint Traffic Assignment

- As traffic volume increases, speed decreases.
- BPR (Bureau of Public Roads) function

$$t_a = t_0\left(1+0.15\left(\frac{x_a}{C_a}\right)^4\right)$$

t_a travel time at traffic flow x_a

t_0 zero-flow or free flow travel time

x_a traffic flow on link a (veh/hr)

C_a practical link capacity (veh/hr)

α, β parameters

Capacity Constraint Traffic Assignment

- Incorporating capacity restraint. The most common method is to load the network and adjust assume link speeds after each loading to reflect volume/ capacity restraints.
- These loading and adjustments are introduced until a balance is obtained.
- Experience shows that a balance can be normally achieved after 3-4 loadings.
- This process attempt to capture the equilibrium nature of the traffic assignment problem. It is a more realistic representation of traffic and is in widespread use.

Capacity Constraint Traffic Assignment

Algorithm

Step 0 *Initialization*. Perform All-or-Nothing based on $t_a^0 = t_a(0), \forall a \in A$, and obtain link flow $\{x_a^0\}$. Set iteration counter $n = 1$.

Step 1 *Updating*. Set $t_a^0 = t_a\left(x_a^{n-1}\right), \forall a \in A$.

Step 2 *Network loading*. Assign all trips to the network using all-or-nothing based on the travel time $\{t_a^n\}$. This yields a set of link flows $\{x_a^n\}$.

Step 3 *Convergence test*. If $\max_{a \in A}\left\{\left|x_a^n - x_a^{n-1}\right| \le \kappa\right\}$, stop. Otherwise, let $n := n+1$ and go to Step 1.

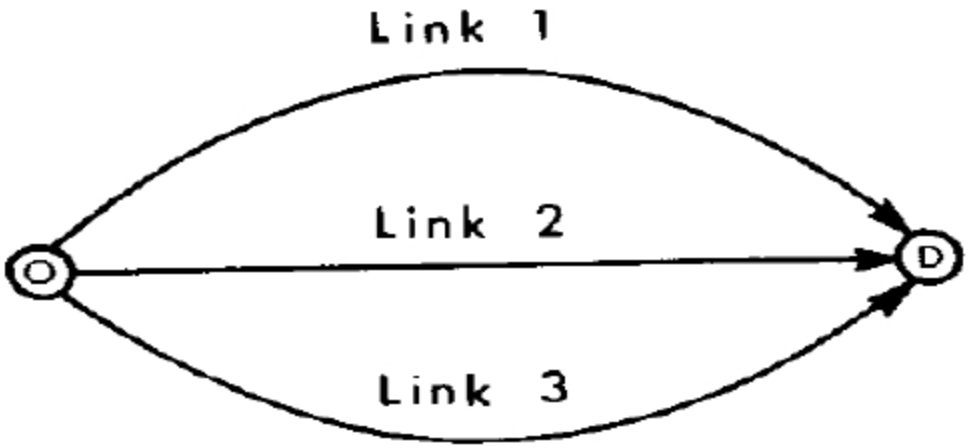


Figure 4.18b

$$t_1 = 10\left[1 + 0.15\left[\frac{x_1}{2}\right]^4\right] \text{ time units}$$

$$t_2 = 20\left[1 + 0.15\left[\frac{x_2}{4}\right]^4\right] \text{ time units}$$

$$t_3 = 25\left[1 + 0.15\left[\frac{x_3}{3}\right]^4\right] \text{ time units}$$

$$x_1 + x_2 + x_3 = 10 \text{ flow units}$$

Capacity Restraint Algorithm Applied to the Network in Figure 5.1

Iteration Number	Algorithmic Step	Link 1	Link 2	Link 3
0	Initialization	$t_1^0 = 10$ $x_1^0 = 10$	$t_2^0 = 20$ $x_2^0 = 0$	$t_3^0 = 25$ $x_3^0 = 0$
1	Update Loading	$t_1^1 = 947$ $x_1^1 = 0$	$t_2^1 = 20$ $x_2^1 = 10$	$t_3^1 = 25$ $x_3^1 = 0$
2	Update Loading	$t_1^2 = 10$ $x_1^2 = 10$	$t_2^2 = 137$ $x_2^2 = 0$	$t_3^2 = 25$ $x_3^2 = 0$
3	Update Loading	$t_1^3 = 947$ $x_1^3 = 0$	$t_2^3 = 20$ $x_2^3 = 10$	$t_3^3 = 25$ $x_3^3 = 0$
		⋮	⋮	⋮

- This algorithm may not converge;

- To remedy this converge problem, first, in stead of using the travel time obtained in the previous iteration for the new loading, a combination of the last two travel times obtained is used. This introduces a “smoothing” effect. Second, the iteration is made to stop after N iterations. The equilibrium flow pattern is taken to be the average flow for each link over the last four iterations. (N should never be less than 4.)

Step 0: Initialization. Perform all-or-nothing assignment based on $t_a^0 = t_a(0), \forall a$. Obtain a set of link flows $\{x_a\}$. Set iteration counter n=1

Step 1: Update. Set $\tau_a^n = t_a(x_a^{n-1}), \forall a$

Step 2: Smoothing. Set $t_a^n = 0.75 t_a^{n-1} + 0.25 \tau_a^n, \forall a$

Step 3: Network Loading. Assign all trips to the network using all-or-nothing based on travel times $\{t_a^n\}$. This yields a set of link flows $\{x_a^n\}$.

Step 4: Stopping rule. If $n = N$, go to stop 5. Otherwise, set n=n+1 and go to step 1.

Step 5: Averaging. Set $x_a^* = \frac{1}{4}\sum_{l=0}^{3} x_a^{n-l}$, $\forall a$ and stop. $\{x_a^*\}$ are the link flows at equilibrium.

- Note that this modified capacity restraint approach does not produce an equilibrium pattern.

Modified Restraint Algorithm Applied to the Network in Figure 5.1

Iteration Number	Algorithmic Step	Link 1	Link 2	Link 3
0	Initialization	$t_1^0 = 10$ $x_1^0 = 10$	$t_2^0 = 20$ $x_2^0 = 0$	$t_3^0 = 25$ $x_3^0 = 0$
1	Update	$\tau_1^1 = 947$	$\tau_2^1 = 20$	$\tau_3^1 = 25$
	Smoothing	$t_1^1 = 244$	$t_2^1 = 20$	$t_3^1 = 25$
	Loading	$x_1^1 = 0$	$x_1^1 = 10$	$x_3^1 = 0$
2	Update	$\tau_1^2 = 10$	$\tau_2^2 = 137$	$\tau_3^2 = 25$
	Smoothing	$t_2^2 = 186$	$t_2^2 = 49$	$t_3^2 = 25$
	Loading	$x_1^2 = 0$	$x_2^2 = 0$	$x_3^2 = 10$
3	Update	$\tau_1^3 = 10$	$\tau_2^3 = 20$	$\tau_3^3 = 488$
	Smoothing	$t_1^3 = 142$	$t_2^3 = 42$	$t_3^3 = 141$
	Loading	$x_1^3 = 0$	$x_2^3 = 10$	$x_3^3 = 0$
Average		$x_1^* = 2.5$ $t_1^* = 13.7$	$x_2^* = 5.0$ $t_2^* = 27.3$	$x_3^* = 2.5$ $t_3^* = 26.8$

Incremental assignment

- Assigns a portion of the O-D entries at each iteration.
- Travel times are then updated and an additional portion of the O-D matrix is loaded onto the network.

Step 0: Preliminaries. Divide each origin-destination entry into N equal portions. (i.e. set $q_{rs}^{n} = q_{rs} / N$). Set n=1 and $x_a^0 = 0, \forall a$

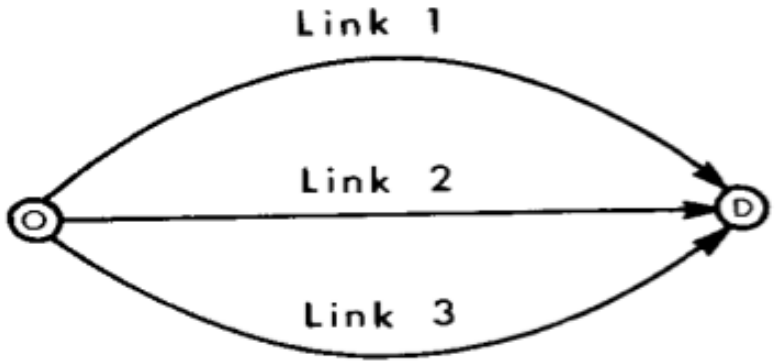
Step 1: Update. Set $t_a^n = t_a(x_a^{n-1}), \forall a$

Step 2: Incremental loading. Perform all-or-nothing assignment based on $\{t_a^n\}$, but using only the trip rates q_{rs}^n for each O-D pair. This yields a flow pattern $\{w_a^n\}$

Step 3: Flow summation. Set $x_a^n = x_a^{n-1} + w_a^n, \forall a$

Step 4: Stopping rule. If $n = N$, stop (the current set of link flows is the solution); otherwise, set n=n+1 and go to step 1.

Example



$$t_1 = 10\left[1 + 0.15\left[\frac{x_1}{2}\right]^4\right] \text{ time units}$$

$$t_2 = 20\left[1 + 0.15\left[\frac{x_2}{4}\right]^4\right] \text{ time units}$$

$$t_3 = 25\left[1 + 0.15\left[\frac{x_3}{3}\right]^4\right] \text{ time units}$$

$$x_1 + x_2 + x_3 = 10 \text{ flow units}$$

Incremental Assignment Algorithm (N=4)

Iteration	Algorithmic Step	output	Link		
			1	2	3
	updating	t	10	20	25
1	incremental loading	w	2.5	0	0
	summation	x	2.5	0	0
	updating	t	13.66211	20	25
2	incremental loading	w	2.5	0	0
	summation	x	5	0	0
	updating	t	68.59375	20	25
3	incremental loading	w	0	2.5	0
	summation	x	5	2.5	0
	updating	t	68.59375	20.45776	25
4	incremental loading	w	0	2.5	0
	summation	x	5	5	0
	travel time at convergence		68.59375	27.32422	25

- The order of loading the O-D pair is important.

- This is not an equilibrium assignment. With large number of iterations, the final flow pattern may approximate an UE pattern. But this is not guaranteed.

内容

1. 交通流分配问题的定义
2. 交通流分配问题的特点
3. 用户均衡
4. 交通流分配算法
   - All-or-Nothing traffic assignment
   - Capacity constraint traffic assignment
   - Incremental assignment
   - Beckmann’s transformation

用户均衡

$$\left(c_{r,w}-\mu_w\right) f_{r,w}=0,\ \ r\in R_w, w\in W$$

$$c_{r,w}-\mu_w\geq 0,\quad f_{r,w}\geq 0,\ \ r\in R_w, w\in W$$

$$\sum_{r\in R_w} f_{r,w}=d_w, w\in W$$

$$x_a=\sum_{w\in W}\sum_{r\in R_w} f_{r,w}\delta_{a,r}, a\in A$$

$$c_{r,w}=\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r}, r\in R_w, w\in W$$

$$\left(\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r}-\mu_w\right) f_{r,w}=0,\ \ r\in R_w, w\in W$$

$$\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r}-\mu_w\geq 0,\quad f_{r,w}\geq 0,\quad r\in R_w, w\in W$$

$$\sum_{r\in R_w} f_{r,w}=d_w,\ \ w\in W$$

$$x_a=\sum_{w\in W}\sum_{r\in R_w} f_{r,w}\delta_{a,r},\quad a\in A$$

贝克曼转化（**Beckmann's transformation**）

- Assumption:

All link travel time functions t(x) are continuous, differentiable, increasing, convex and separable.

Convex: $d^2 t_a(x_a)/dx_a^2 \geq 0,\ x_a \geq 0,\ a \in A$

Separable: $\frac{dt_a(x_a)}{dx_a} \geq 0,\ \frac{dt_a(x_a)}{dx_b} = 0,\ a \neq b,\ a,b \in A$

贝克曼转化（**Beckmann's transformation**）

If all link travel time functions t(x) are continuous, differentiable, increasing, convex, separable and additive, then the user equilibrium flow pattern is identical to the solution of the following optimization problem.

$$\left(\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r} - \mu_w\right) f_{r,w} = 0,\ \ r \in R_w, w \in W$$

$$\sum_{a\in A} t_a\left(x_a\right)\delta_{a,r} - \mu_w \geq 0,\ \ f_{r,w} \geq 0,\ \ \ r \in R_w, w \in W$$

$$\sum_{r\in R_w} f_{r,w} = d_w,\ \ w \in W$$

$$x_a = \sum_{w\in W}\sum_{r\in R_w} f_{r,w}\delta_{a,r},\ \ \ a \in A$$

$\Longleftrightarrow$

$$\min \sum_{a\in A}\int_0^{x_a} t_a\left(w\right)dw \qquad (1)$$

s.t.

$$x_a = \sum_{w\in W}\sum_{r\in R_w} f_{r,w}\delta_{a,r},\ \ \ a \in A \qquad (2)$$

$$\sum_{r\in R_w} f_{r,w} = d_w,\ \ w \in W \qquad (3)$$

$$f_{r,w} \geq 0,\ \ \ r \in R_w, w \in W \qquad (4)$$

This equivalence can be demonstrated by proving that first-order conditions for the minimization program are identical to the equilibrium conditions.