2021.04.13
2021夏学期ゼミ #3

利用者均衡配分モデル

M2 黛 風雅

目次

1. 交通量配分問題の前提知識

1.1 交通計画における配分問題

- **配分問題(traffic assignment)**

「**交通ネットワーク**」を対象に, …「グラフによるネットワーク表現」
「**需要OD交通量**」と …「各ODペア間の交通量」
「**配分原則**」を与件として, …「仮定する利用者の行動原理」

ネットワークの各リンクを流れる交通量を予測する問題.

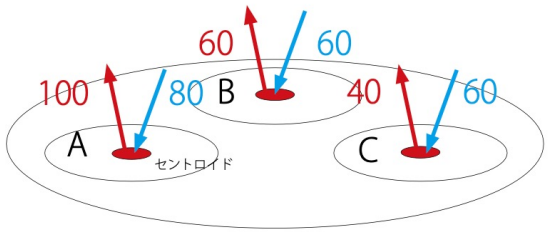
生成交通量の予測

発生・集中交通量の予測

分布交通量の予測

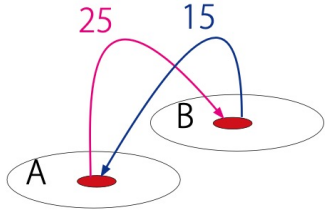
分担交通量の予測

配分交通量の予測



分布交通量

ゾーン	A	B	C	発生
A	40	25	35	100
B	15	25	20	60
C	25	10	5	40
集中	80	60	60	200



1.1 交通計画における配分問題

- **配分問題(traffic assignment)**

「**交通ネットワーク**」を対象に, …「グラフによるネットワーク表現」
「**需要OD交通量**」と …「各ODペア間の交通量」
「**配分原則**」を与件として, …「仮定する利用者の行動原理」

ネットワークの各リンクを流れる交通量を予測する問題.

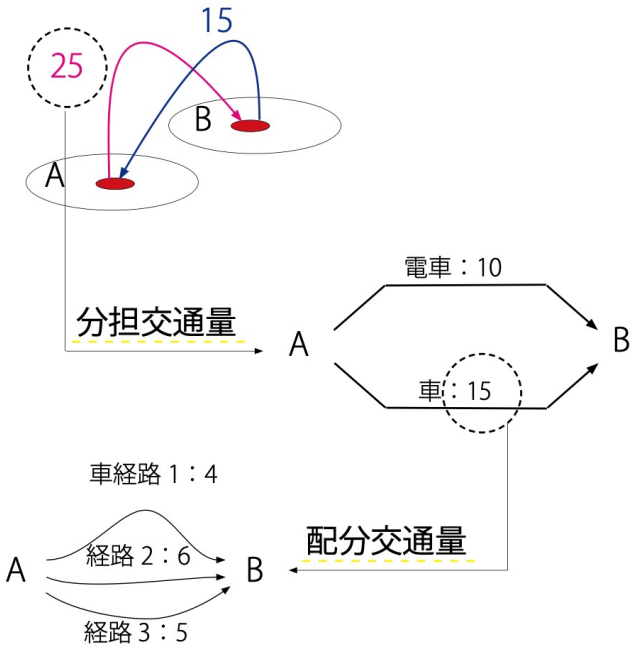
生成交通量の予測

発生・集中交通量の予測

分布交通量の予測

分担交通量の予測

配分交通量の予測



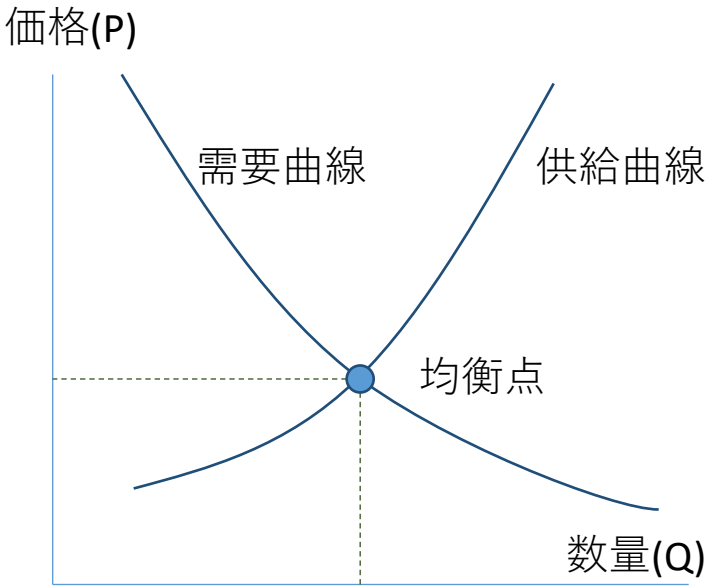
1.2 均衡の概念

- 均衡状態とは

財（サービス）の需要者と供給者の行動を，価格と取引量の関数としてそれぞれモデル化した時の交点となる価格・取引量によって記述される.

- Nash均衡概念（均衡の安定性評価）

非協力ゲームの他のプレイヤーの戦略を固定して，各プレイヤーが自己の利潤最大化行動のみ追求する場合に，他の戦略に逸脱しない戦略の組をNash均衡と言い，安定状態である.



Wardrop(1952)が経済学における均衡の概念を配分の問題として交通に持ち込み，２つの交通配分原則を提案

⇩

Wardrop均衡原理

1.3 Wardropの原則

Wardropの第一原則(等時間原則)

利用される経路の旅行時間は皆等しく、利用されない経路の旅行時間よりも小さいか、せいぜい等しい.

→利用者最適配分：利用者の視点

Wardropの第二原則(総走行時間最小化原則)

道路ネットワーク上の総旅行時間が最小となる.

→システム最適配分(SO)：道路管理者の視点

利用者均衡配分 … Wardropの第一原則(等時間原則)を配分原則とする

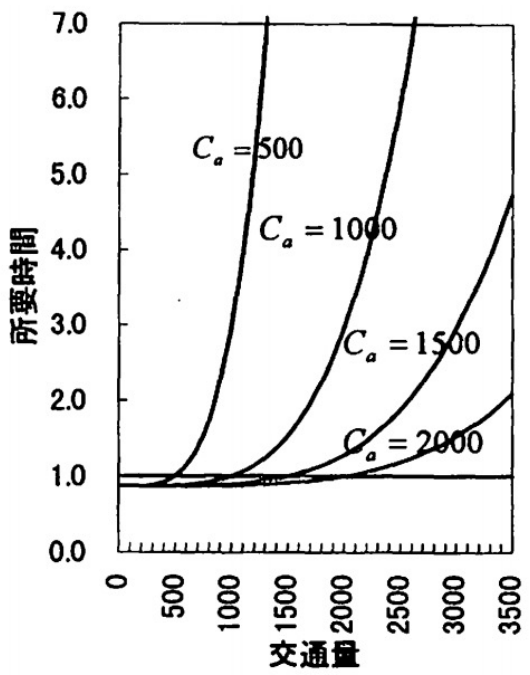
＝誰かが一方的に経路を変えることによって、
誰一人として実所要時間を改善できない状態

→リンク上のフローとリンクパフォーマンスの関係

1.4 リンクパフォーマンス

- **リンクパフォーマンス関数**

ネットワークを構成する個々のリンクのサービス水準（一般的に旅行時間）を，リンク交通量とリンク属性の関数として表わすもの．米国道路局の開発したBPR型関数が一般的に用いられる．



BPR型リンクパフォーマンス関数
交通ネットワークの均衡分析（土木学会,1998）

- **BPR関数**

$$t_a(x_a) = t_{a0}\left\{1 + \alpha\left(\frac{x_a}{C_a}\right)^{\beta}\right\}$$

- **変数の定義**

t_a …リンクaの旅行時間

t_{a0} …リンクaの自由旅行時間

x_a …リンクaの時間交通量(台/時)

C_a …リンクaの時間交通容量(台/時)

α, β …パラメータ

1.4 リンクパフォーマンス

- **リンクの性質**

flow dependent

交通量変化によるリンクコスト（所要時間や料金等）の変化を考慮する場合
＝リンク上のフロー(x_a)が旅行時間(t_a)に影響する場合

【BPR関数】

$$t_a(x_a) = t_{a0}\left\{1 + \alpha\left(\frac{x_a}{C_a}\right)^{\beta}\right\}$$

t_a …リンクaの旅行時間
t_{a0} …リンクaの自由旅行時間
x_a …リンクaの時間交通量(台/時)
C_a …リンクaの時間交通容量(台/時)
α, β …パラメータ

flow independent

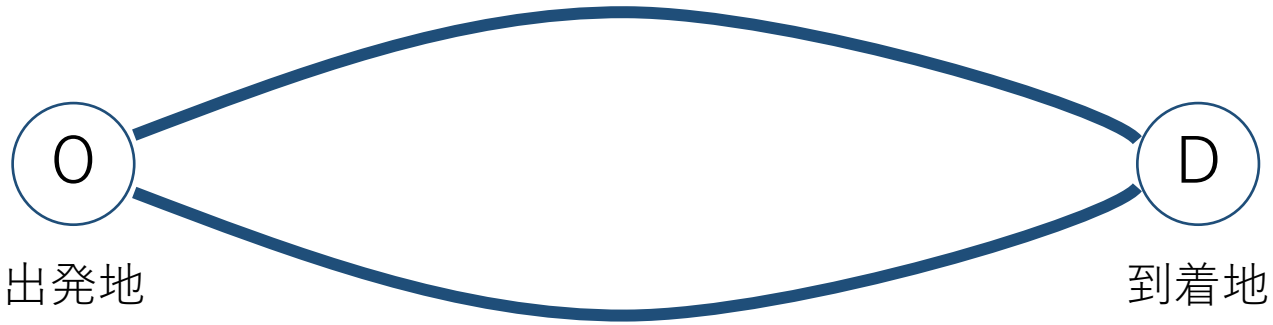
交通量変化によるリンクコストの変化を考慮しない場合
e.g. 鉄道路線網や混雑が酷くないバス

1.4 リンクパフォーマンス

■ **演習問題**

1200台の車を２通りの配分原則に基づいて２本の経路に配分してください

経路1　$t_1 = 5 + 0.1h_1$　**：リンクパフォーマンス関数**



経路2　$t_2 = 10 + 0.025h_2$

Wardropの第一原則(等時間原則)

利用される経路の旅行時間は皆等しく、利用されない経路の旅行時間よりも小さいか、せいぜい等しい.

Wardropの第二原則(総走行時間最小化原則)

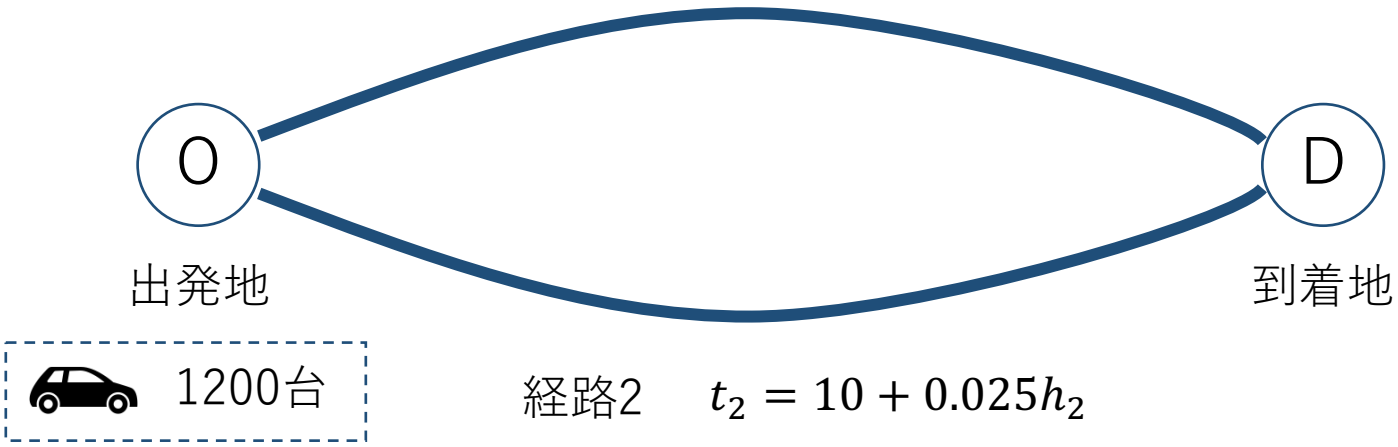
道路ネットワーク上の総旅行時間が最小となる.

1.4 リンクパフォーマンス

■ **演習問題（解答①）**

1200台の車を２通りの配分原則に基づいて２本の経路に配分してください

経路1　$t_1 = 5 + 0.1h_1$　**：リンクパフォーマンス関数**



経路2　$t_2 = 10 + 0.025h_2$

Wardropの第一原則(等時間原則)

利用される経路の旅行時間は皆等しく、利用されない経路の旅行時間よりも小さいか、せいぜい等しい.

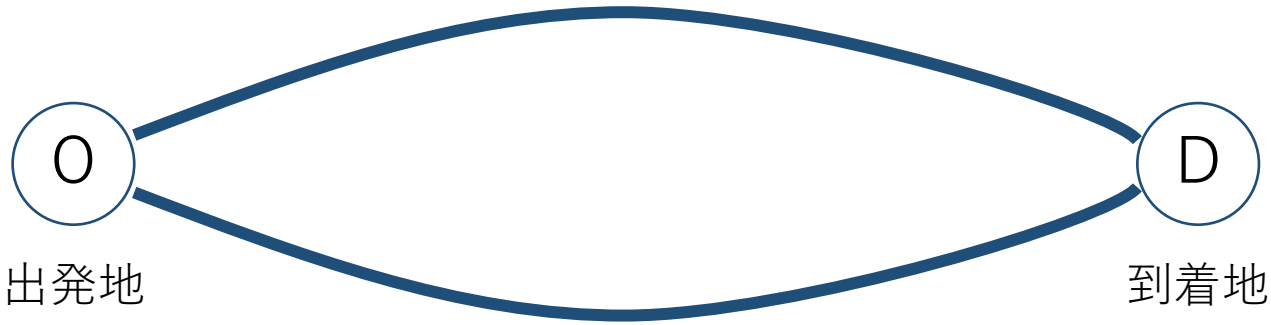
$$\begin{cases} t_1 = t_2 \\ h_1 + h_2 = 1200 \end{cases} \Rightarrow \begin{cases} t_1 = t_2 = 33, T = 39600 \\ h_1 = 280, h_2 = 920 \end{cases}$$

1.4 リンクパフォーマンス

■ **演習問題（解答②）**

1200台の車を２通りの配分原則に基づいて２本の経路に配分してください

経路1　$t_1 = 5 + 0.1h_1$　**：リンクパフォーマンス関数**



経路2　$t_2 = 10 + 0.025h_2$

Wardropの第二原則(総走行時間最小化原則)

道路ネットワーク上の総旅行時間が最小となる.

総所要時間を $T = t_1h_1 + t_2h_2$ について,

$$\min T \quad s.t. \quad h_1 + h_2 = 1200$$

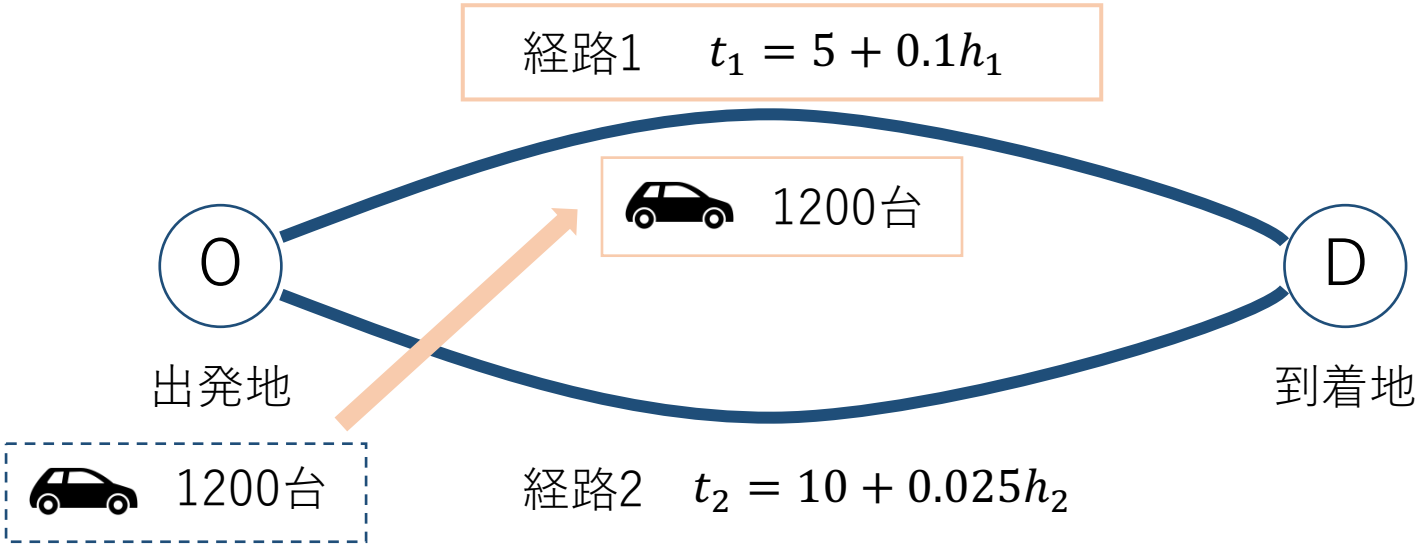
⇨

$$\begin{cases} t_1 = 31, t_2 = 33.5, T = 39550 \\ h_1 = 260, h_2 = 940 \end{cases}$$

1.5 配分手法

- **需要配分（all-or-nothing配分(AON)）**

flow independentな状況において，最短経路に全ての交通量を割り当てる．
各種配分方法/アルゴリズムの各プロセスにおいて用いられる．

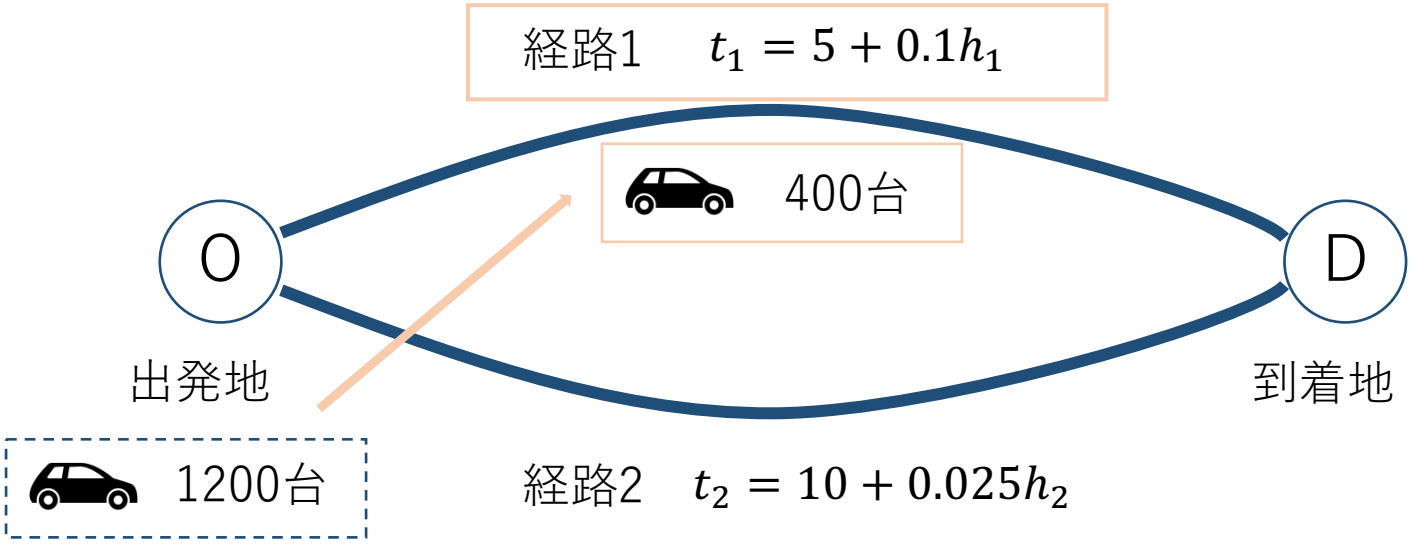


経路1に1200台全てを割り当てる→ $t_1 = 5 + 0.1 \times 1200 = 125$

1.5 配分手法

- **実際配分（分割配分法(IA)）** Incremental Assignment Method

all-or-nothing配分から発展させた方法．一度に交通量を配分するのではなく，これを n 回に分割して，各回ごとにリンクパフォーマンスを更新しながら最短経路探索を行い配分する．



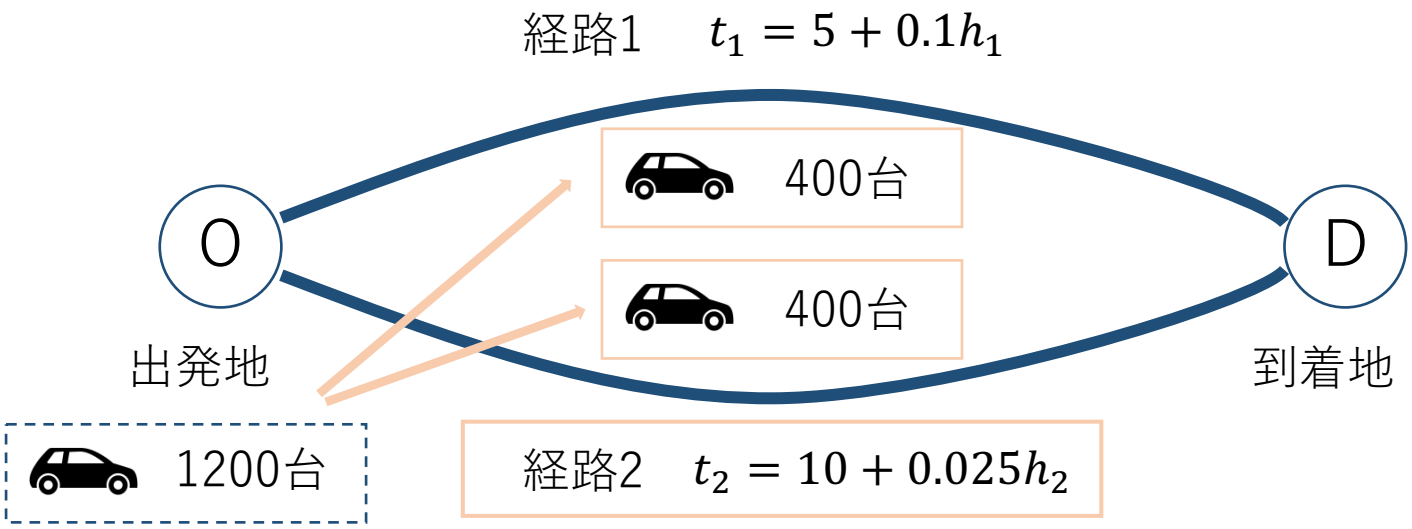
3等分割の場合）

経路1に400台を割り当てる→ $t_1 = 5 + 0.1 \times 400 = 45 > 10$

1.5 配分手法

- **実際配分（分割配分法(IA)）** Incremental Assignment Method

all-or-nothing配分から発展させた方法．一度に交通量を配分するのではなく，これを n 回に分割して，各回ごとにリンクパフォーマンスを更新しながら最短経路探索を行い配分する．



3等分割の場合）

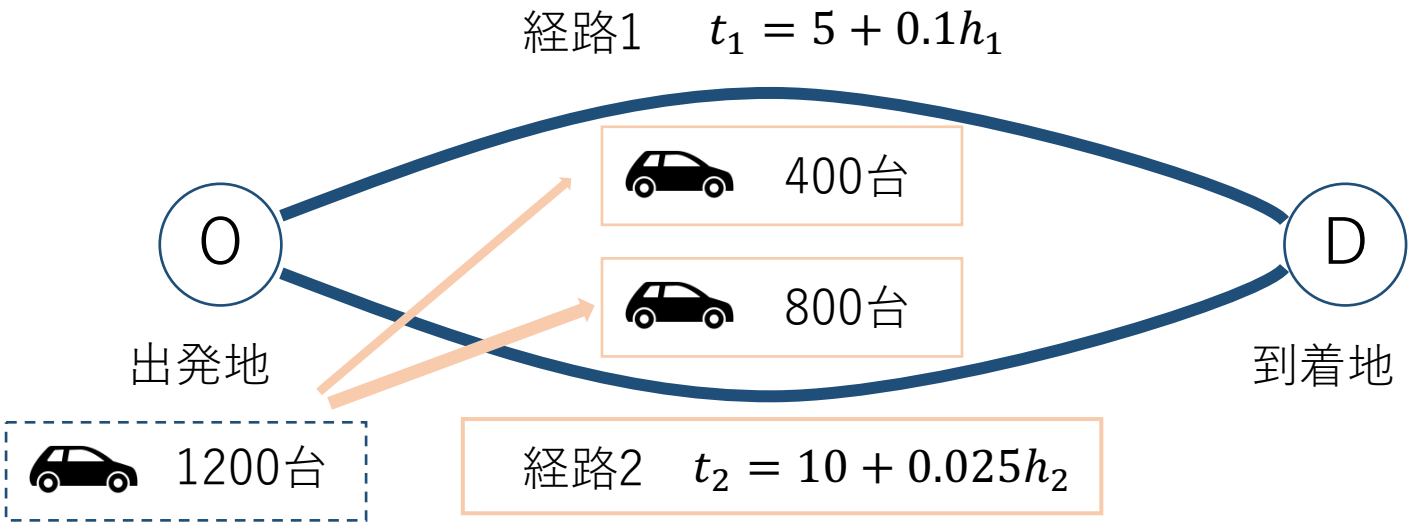
経路1に400台を割り当てる→ $t_1 = 5 + 0.1 \times 400 = 45 > 10$

経路2に400台を割り当てる→ $t_2 = 10 + 0.025 \times 400 = 20 < 45$

1.5 配分手法

- **実際配分（分割配分法(IA)）** Incremental Assignment Method

all-or-nothing配分から発展させた方法．一度に交通量を配分するのではなく，これを n 回に分割して，各回ごとにリンクパフォーマンスを更新しながら最短経路探索を行い配分する．



3等分割の場合）

経路1に400台を割り当てる→ $t_1 = 5 + 0.1 \times 400 = 45 > 10$

経路2に400台を割り当てる→ $t_2 = 10 + 0.025 \times 400 = 20 < 45$

経路2に400台を割り当てる→ $t_2 = 10 + 0.025 \times 800 = 30 < 45$

1.5 配分手法

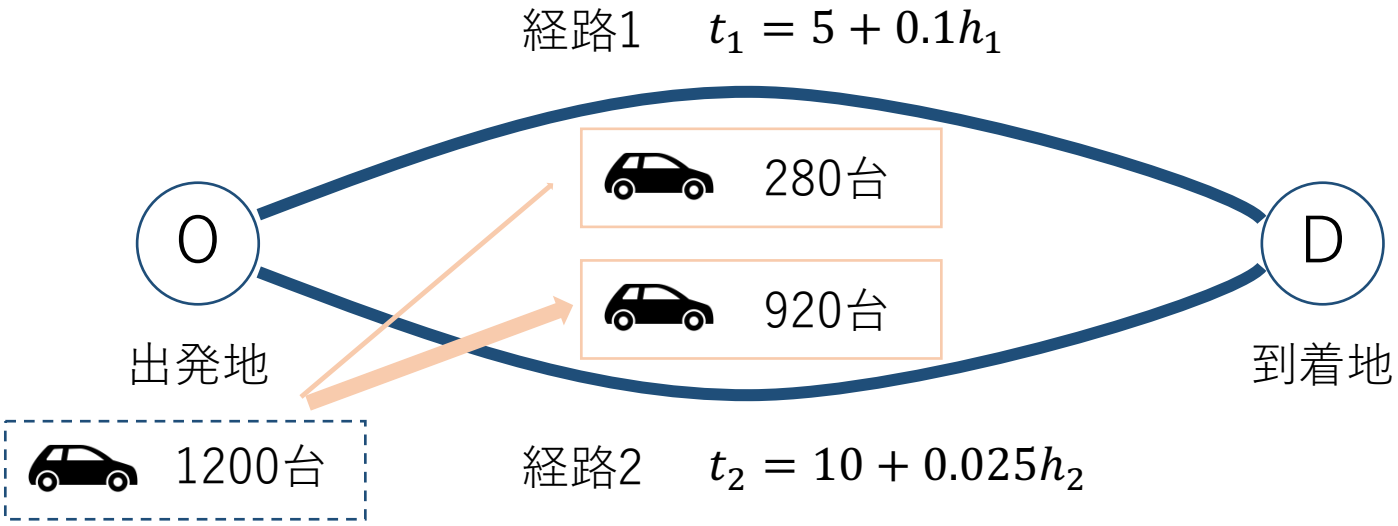
- **確定的利用者均衡配分**（需要固定型利用者均衡配分）

  User **E**quilibrium assignment with Fixed Demand(**UE**/FD)

Wardropの第一原則を配分原則とした均衡原理を実際のネットワークに当てはめた場合のモデル．OD需要を固定した利用経路間の均衡問題．

利用者の行動選択に関する仮定
①常に旅行時間を最小とするように行動する．
②利用可能な経路に関する完全な情報を得ている．
③同一の評価基準で経路を選択する．

※ OD分布と交通量配分を同時決定する需要変動型(UE/VD)もある



1.5 配分手法

- **確率的利用者均衡配分**　Stochastic **U**ser **E**quilibrium assignment(**SUE**)

UEが前提とする条件②や条件③の仮定を排し，利用者の経路選択の多様性や不確実性をランダム効用理論に基づいて考慮するモデル.

利用者の行動選択に関する仮定
①常に旅行時間を最小とするように行動する.
~~②利用可能な経路に関する完全な情報を得ている.~~
~~③同一の評価基準で経路を選択する.~~

←認知の誤差を考慮

SUEの均衡状態は「どの利用者も自分が経路を変更することによって自分の経路費用を減少させることが出来ないと信じている状態」として表現される

- **動的利用者均衡配分(DUE)**　**D**ynamic **U**ser **E**quilibrium assignment (**DUE**)

任意の時刻に出発した車が、目的地に到着するまでの経路の旅行時間に対して、常にWardropの均衡が成立する.

2. 確定的利用者均衡配分(UE)とその解法

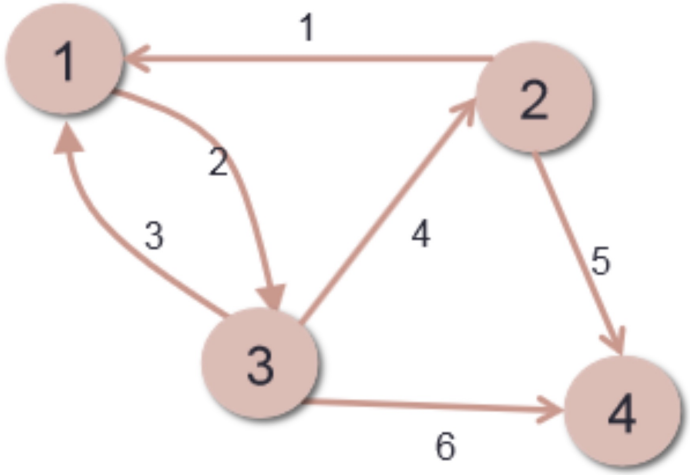
2.1 UEの定式化

- **現実規模の交通網を扱えるネットワーク表現の精緻化**
  - ・現実の道路網ではOD間の経路は非常に多い
  - ・複数のODペア間の交通需要が各リンクの交通量を構成

・ネットワークのグラフ表現

ノードとリンクの集合として記述

ノード集合：N = {1,2,3,4}
(有向)リンクの集合：A = {1,2,3,4,5,6}



・始点/終点セントロイド

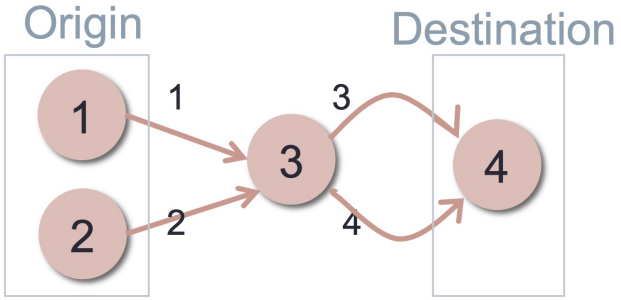
フローが発生/集中するノード

・パス

Origin, Destinationを結ぶ経路.
一般に各ODペアに対して複数のパスが存在

2.1 UEの定式化

- **ネットワークを記述する変数の定義**



a：リンク
r：出発地
s：目的地
k：経路

$A = \{1,2,3,4\}$：リンク集合
$\Omega = \{(1,4),(2,4)\}$：OD集合
$\forall a \in A, \forall rs \in \Omega$

q_{rs}：ODペアrs間の総交通量
$\delta_{a,k}^{rs}$：ダミー変数
リンクaがODペア(r,s)を結ぶパスkに含まれる.
TRUE = 1, FALSE= 0

OD表(行列**q**)

$$\mathbf{q} = \begin{pmatrix} q_{11} & \cdots & q_{1s} \\ \vdots & \ddots & \vdots \\ q_{r1} & \cdots & q_{rs} \end{pmatrix}$$

- **リンク単位の集計とパス単位の集計との関係**

c_k^{rs}：ODペアrsを結ぶ経路kにおける旅行時間
t_a：リンクaにおける旅行時間(BPR関数)

$$c_k^{rs} = \sum_{a \in A} t_a \delta_{a,k}^{rs}$$

f_k^{rs}：ODペアrsを結ぶ経路kにおけるフロー数
x_a：リンクaにおけるフロー数

$$x_a = \sum_{rs \in \Omega} \sum_{k \in K_{rs}} f_k^{rs} \delta_{a,k}^{rs}$$

2.1 UEの定式化

- **Wardropの利用者均衡の数学的定式化**

Wardropの第一原則が成立している状態を以下のように表現

【Wardropの第一原則】

$f_k^{rs} > 0$ のとき　$c_k^{rs} - c_{min}^{rs} = 0$

$f_k^{rs} = 0$ のとき　$c_k^{rs} - c_{min}^{rs} \geq 0$

「利用される経路の旅行時間は皆等しく、
利用されない経路の旅行時間よりも小さいか、
せいぜい等しい」

$s.t$　$\sum_k f_k^{rs} - q_{rs} = 0$　（フローの保存則）

$f_k^{rs} \geq 0$　（フロー非負）

現実規模のネットワークでこの解を得るのは非常に困難

→Beckmann & Winsten (1956) が数理最適化問題として変換.

最適化問題はある程度解法が確立しているため均衡解を効率よく導けるように.

2.1 UEの定式化

【UE/FD-Primal】

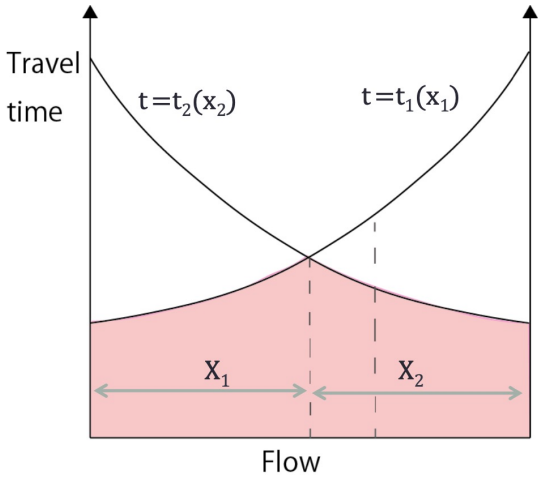
$$\min Z(\boldsymbol{x}) = \sum_a \int_0^{x_a} t_a(\boldsymbol{\omega})\, d\omega$$

リンクaにおけるフロー数がωのときの旅行時間

ネットワーク上の全リンクの旅行時間の和

$$s.t \quad \sum_k f_k^{rs} - q_{rs} = 0 \quad x_a = \sum_{rs\in\Omega}\sum_{k\in K_{rs}} f_k^{rs}\delta_{a,k}^{rs}$$

$$f_k^{rs} \geq 0 \quad x_a \geq 0 \quad \forall a \in A,\ \forall rs \in \Omega$$

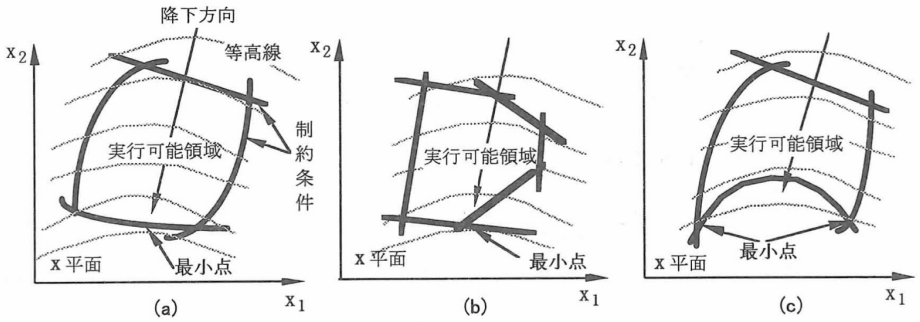


目的関数の解釈：
積分の値の最小値を与える点を探索

- **Wardropと等価であることの証明**

→Lagrangian関数として再定義する

※解の唯一性を担保する重要な性質
①制約条件式の凸性
　∵制約条件式が全て線形
②目的関数の狭義凸性
　∵ Hessian $\nabla^2 Z$ が正定値行列（一次元関数においては 2 階微分が正であること）
　∵ t_aが x_a について単調増加：リンクパフォーマンス関数の性質



図−8　実行可能領域の凸性と解の唯一性との関係

2.1 UEの定式化

- **Lagrangian関数の導入**

【一般的な制約条件付き最適化問題】

$$\min Z(x)$$

$$s.t \quad g_n(\boldsymbol{x}) - d_n \geq 0, \quad \forall n \in \zeta$$

$$x_m \geq 0$$

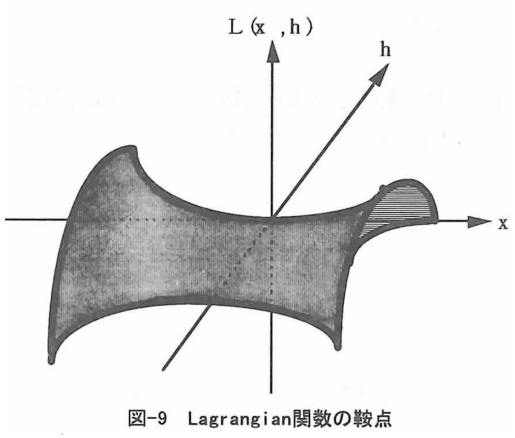


図-9 Lagrangian関数の鞍点

Lagrangian関数として制約条件を関数内に取り込み

【Lagrangian関数】

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = Z(x) + \sum_n \lambda_n (d_n - g_n(\boldsymbol{x}))$$

$$s.t \quad x_m \geq 0$$

相補性条件：
効いている制約条件において$\lambda_n \neq 0$

$L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$は，
$\boldsymbol{x}$について$L(\boldsymbol{x}, \boldsymbol{\lambda})$を最小化
$\boldsymbol{\lambda}$について$L(\boldsymbol{x}, \boldsymbol{\lambda})$を最大化

※図中のhはここでは$\boldsymbol{\lambda}$として定義
$\boldsymbol{\lambda}$：Lagrangian乗数
非負の定数

→制約条件のないLagrangian関数について，
$L(\boldsymbol{x}, \boldsymbol{\lambda})$の停留点$(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$：鞍点を求める問題として再定義できる（嬉しい！）

2.1 UEの定式化

- **UE/FD-PrimalへのLagrangian関数の導入**

【UE/FD-Primal】

$$\min Z(\boldsymbol{x}) = \sum_a \int_0^{x_a} t_a(\boldsymbol{\omega})\, d\omega$$

$$s.t \quad \sum_k f_k^{rs} - q_{rs} = 0 \qquad \forall rs \in \Omega$$

$$x_a = \sum_{rs\in\Omega} \sum_{k\in K_{rs}} f_k^{rs} \delta_{a,k}^{rs} \qquad \forall a \in A,\ \forall rs \in \Omega$$

$$f_k^{rs} \geq 0 \qquad \forall rs \in \Omega$$

$$x_a \geq 0 \qquad \forall a \in A$$

※$\boldsymbol{x}$を$\boldsymbol{f}$の関数として立式

【Lagrangian関数】

$$L(\boldsymbol{f}, \boldsymbol{\lambda}) = Z(x(\boldsymbol{f})) - \sum_r \sum_s \lambda_{rs} \left\{ \sum_k f_k^{rs} - q_{rs} \right\} f_k^{rs}$$

$$s.t \quad f_k^{rs} \geq 0$$

2.1 UEの定式化

- **Karush-Kuhn-Tucker条件（KKT条件）：一次の最適性条件**

【Lagrangian関数】

$$L(\boldsymbol{f},\boldsymbol{\lambda}) = Z(x(\boldsymbol{f})) - \sum_r \sum_s \lambda_{rs} \left\{ \sum_k f_k^{rs} - q_{rs} \right\} f_k^{rs}$$

$$s.t \qquad f_k^{rs} \geq 0$$

Lagrangian関数を$\boldsymbol{f}$について最小化する鞍点：KKT条件を求める

$$\frac{\partial L(\boldsymbol{f}^*,\boldsymbol{\lambda}^*)}{\partial f_k^{rs}} = 0$$

$L(\boldsymbol{f}^*,\boldsymbol{\lambda}^*)$は，　$\boldsymbol{f}$について$L(\boldsymbol{f},\boldsymbol{\lambda})$を最小化

$$\lambda_{rs} \frac{\partial L(\boldsymbol{f}^*,\boldsymbol{\lambda}^*)}{\partial \lambda_{rs}} = 0 \; and \; \frac{\partial L(\boldsymbol{f}^*,\boldsymbol{\lambda}^*)}{\partial \lambda_{rs}} \leq 0$$

$L(\boldsymbol{f}^*,\boldsymbol{\lambda}^*)$は，　$\boldsymbol{\lambda}$について$L(\boldsymbol{f},\boldsymbol{\lambda})$を最大化

鞍点における必要条件として，

$\frac{\partial L(\boldsymbol{x}^*,\boldsymbol{\lambda}^*)}{\partial \lambda_{rs}} = 0$ または，　$\lambda_{rs} = 0$の

$$f_k^{rs} \geq 0$$

2.1 UEの定式化

- **Lagrangianの偏微分値の計算**

$$\frac{\partial L(\boldsymbol{f}^*, \boldsymbol{\lambda}^*)}{\partial f_k^{rs}} = \frac{\partial}{\partial f_k^{rs}}\left[Z\left(x(\boldsymbol{f}^*)\right) - \sum_r \sum_s \lambda_{rs}\left\{\sum_k f_k^{rs} - q_{rs}\right\} f_k^{rs}\right]$$

$$= \frac{\partial}{\partial f_k^{rs}}\left\{\sum_{a\in A}\int_0^{x_a} t_a(\boldsymbol{\omega})\, d\omega\right\} - \frac{\partial}{\partial f_k^{rs}}\sum_r \sum_s \lambda_{rs}\left\{\sum_k f_k^{rs} - q_{rs}\right\} f_k^{rs}$$

$$= \sum_{a\in A} \underbrace{\frac{d}{dx_a}\left\{\int_0^{x_a} t_a(\boldsymbol{\omega})\, d\omega\right\}}_{t_a(x_a)} \underbrace{\frac{\partial x_a}{\partial f_k^{rs}}}_{\delta_{a,k}^{rs}} - \lambda_{rs}$$

（リンク-経路間の関係式から）

$$\because \frac{\partial x_a}{\partial f_k^{rs}} = \frac{\partial}{\partial f_k^{rs}}\left\{\sum_{rs\in\Omega}\sum_{k\in K_{rs}} f_k^{rs}\delta_{ak}^{rs}\right\} = \delta_{a,k}^{rs}$$

$$= \sum_{a\in A} t_a(x_a)\delta_{a,k}^{rs} - \lambda_{rs}$$

$$\because c_k^{rs} = \sum_{a\in A} t_a \delta_{a,k}^{rs}$$

$$= c_k^{rs} - \lambda_{rs}$$

$$\frac{\partial L(\boldsymbol{f}^*, \boldsymbol{\lambda}^*)}{\partial \lambda_{rs}} = \frac{\partial}{\partial \lambda_{rs}}\left[Z\left(x(\boldsymbol{f}^*)\right) - \sum_r \sum_s \lambda_{rs}\left\{\sum_k f_k^{rs} - q_{rs}\right\} f_k^{rs}\right] = \underbrace{\left(q_{rs} - \sum_k f_k^{rs}\right)}_{\text{フロー保存則}} f_k^{rs}$$

2.1 UEの定式化

- **KKT条件の再掲**

$$c_k^{rs} - \lambda_{rs} = 0$$

$$\left(q_{rs} - \sum_k f_k^{rs}\right) f_k^{rs} = 0 \; and \left(q_{rs} - \sum_k f_k^{rs}\right) f_k^{rs} \leq 0$$

$$f_k^{rs} \geq 0$$

Wardropの第一原則の定義と等価な結果が得られる

$f_k^{rs} > 0$ のとき, $c_k^{rs} = \lambda_{rs} \equiv c_{min}^{rs}$, $q_{rs} - \sum_k f_k^{rs} = 0$

…利用される経路の旅行時間は皆等しく、

$f_k^{rs} = 0$ のとき, $c_k^{rs} \geq \lambda_{rs} \equiv c_{min}^{rs}$, $q_{rs} = 0$

…利用されない経路の旅行時間よりも小さいか、せいぜい等しい

2.2 all-or-nothing配分のアルゴリズム

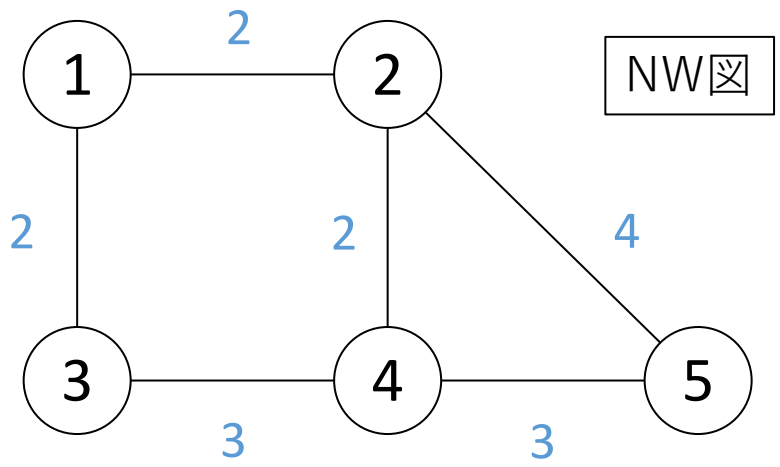
- **all-or-nothing配分のアルゴリズム(UE解法の準備として)**

リンクコストが流量に依存しない(**flow-independent**な)場合の解法
最短経路探索 → **最小費用経路へ全需要配分**を全ての起点に対して実行.

Step 1	$n=0$ とする．始点ノード i，終点ノード j を持つすべてのリンク交通量 x_{ij} について $x_{ij}=0$ とする
Step 2	n番目の起点(セントロイド)を o とする．起点 o から他のすべてのノードへ最短経路探索を行い，起点 o から他のすべてのノードへの最短経路 $C_{min}[o \to i]$ と各ノード i に対する先行ポインタ F_i を求める
Step 3	$C_{min}[o \to j]$ の降順(o から遠い順)にノード j を考える．ノード j に流入し，かつ最短経路ツリーに含まれるリンク $i(=F_j) \to j$ の交通量 x_{ij} を次式で改訂する．$$x_{ij} = x_{ij} + \left(q_{oj} + \sum_{m \in O_j} x_{jm} \right)$$ O_j:ノードjから流出するリンクの終点集合 q_{oj}:起点oから終点jまでの分布交通量
Step 4	$n=N$ なら終了.そうでなければ $n=n+1$ としStep 2へ

2.2 all-or-nothing配分のアルゴリズム

- **all-or-nothing配分のアルゴリズム(UE解法の準備として)**



NW図

値はリンクコストに対応

OD表

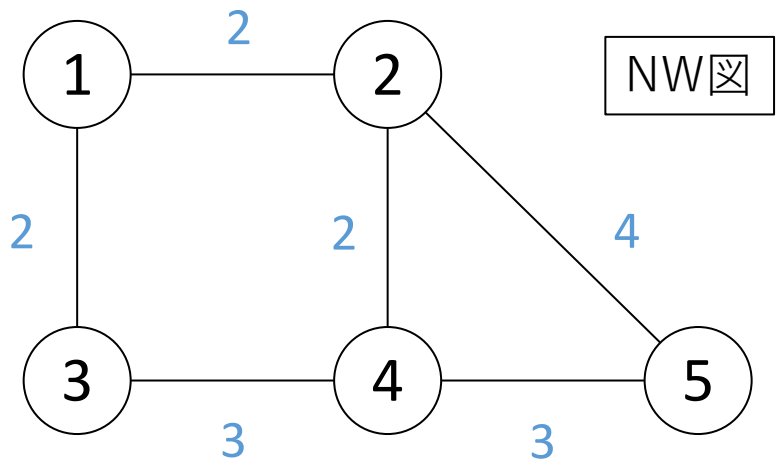
O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

(Step 1) 全 x_{ij} について $x_{ij} = 0$ とする
(Step 2) Dijkstra法での最短経路探索($o = 1$)

(Step 3) $C_{min}[o \rightarrow j]$ の降順で x_{ij} を更新

2.2 all-or-nothing配分のアルゴリズム

- **all-or-nothing配分のアルゴリズム(UE解法の準備として)**



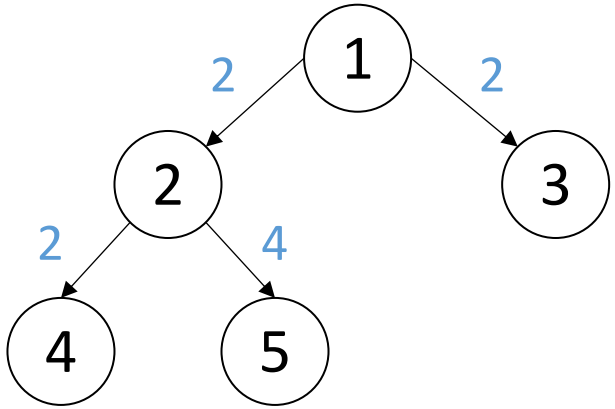
NW図

値はリンクコストに対応

OD表

O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

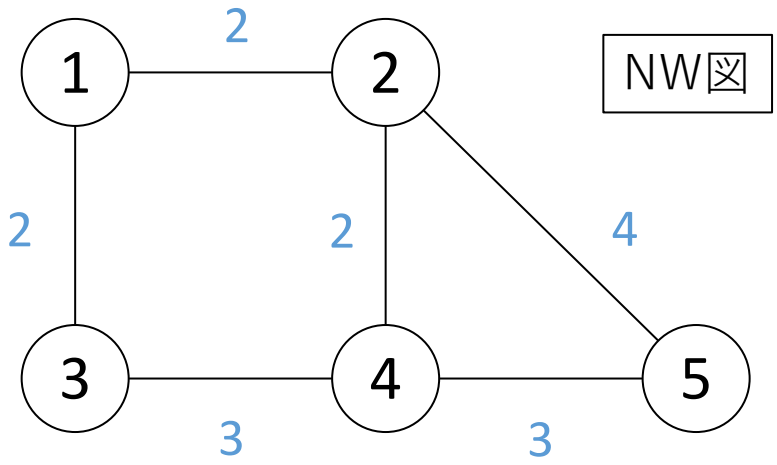
(Step 1) 全 x_{ij} について $x_{ij} = 0$ とする
(Step 2) Dijkstra法での最短経路探索($o = 1$)



(Step 3) $C_{min}[o \rightarrow j]$ の降順で x_{ij} を更新

2.2 all-or-nothing配分のアルゴリズム

- **all-or-nothing配分のアルゴリズム(UE解法の準備として)**

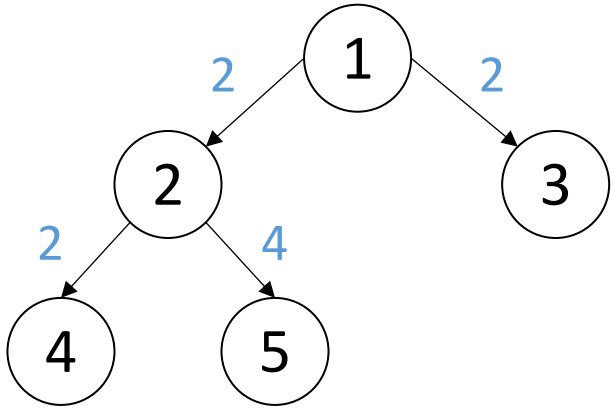


値はリンクコストに対応

OD表

O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

(Step 1) 全 x_{ij} について $x_{ij} = 0$ とする
(Step 2) Dijkstra法での最短経路探索($o = 1$)



(Step 3) $C_{min}[o \rightarrow j]$ の降順で x_{ij} を更新

$x_{25} = 0 + (3 + 0) = 3$

$x_{24} = 0 + (2 + 0) = 2$

$x_{12} = 0 + (4 + 3 + 2) = 9$

$x_{13} = 0 + (5 + 0) = 5$

$$x_{ij} = x_{ij} + \left(q_{oj} + \sum_{m \in O_j} x_{jm} \right)$$

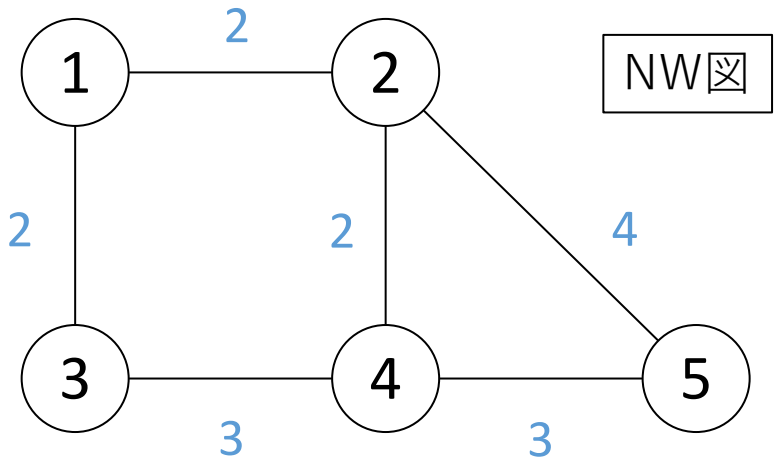
O_j:ノードjから流出するリンクの終点集合

q_{oj}:起点oから終点jまでの分布交通量

(Step 4) $n = n + 1$としてstep2へ($o = 2$)

2.2 all-or-nothing配分のアルゴリズム

■ **演習問題：**起点 o がノード 2の場合についてall-or-nothing配分を実行してください



値はリンクコストに対応

OD表

O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

(Step 2) Dijkstra法での最短経路探索($o = 2$)

(Step 3) $C_{min}[o \to j]$ の降順で x_{ij} を更新

(参考)

o = ノード1の結果

$x_{12} = 9$　$x_{24} = 2$

$x_{13} = 5$　$x_{25} = 3$

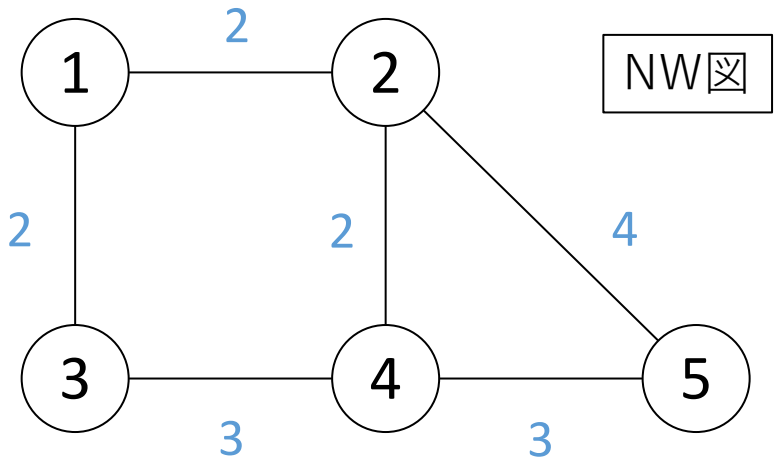
$$x_{ij} = x_{ij} + \left(q_{oj} + \sum_{m \in O_j} x_{jm}\right)$$

O_j:ノードjから流出する
リンクの終点集合

q_{oj}:起点oから終点jまでの
分布交通量

2.2 all-or-nothing配分のアルゴリズム

■ **演習問題（解答）**：起点 o がノード 2の場合についてのall-or-nothing配分



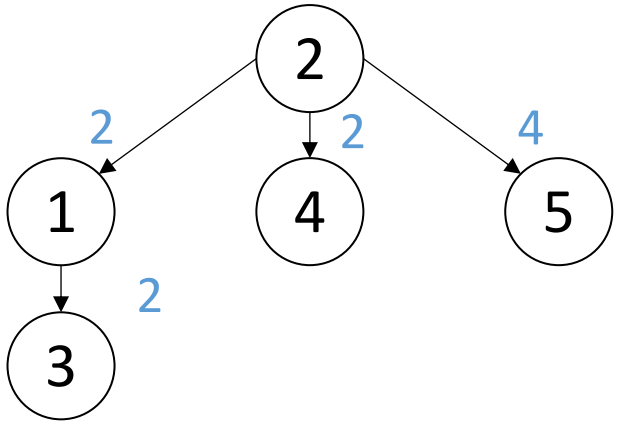
NW図

値はリンクコストに対応

OD表

O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

(Step 2) Dijkstra法による最短経路探索



(Step 3) $C_{min}[o \rightarrow j]$ の降順で x_{ij} を更新

(参考)

o = ノード1の結果

$x_{12} = 9$　$x_{24} = 2$

$x_{13} = 5$　$x_{25} = 3$

$$x_{ij} = x_{ij} + \left(q_{oj} + \sum_{m \in O_j} x_{jm} \right)$$

O_j:ノードjから流出する
リンクの終点集合

q_{oj}:起点oから終点jまでの
分布交通量

$x_{13} = 5 + (4 + 0) = 9$　$x_{25} = 3 + (6 + 0) = 9$

$x_{24} = 2 + (1 + 0) = 3$　$x_{21} = 0 + (3 + 9) = 12$

(Dialアルゴリズム：SUEにおける配分)

- **Dialのアルゴリズム**

妥当な経路にロジット型で確率的に配分する
SUE解法における配分理論

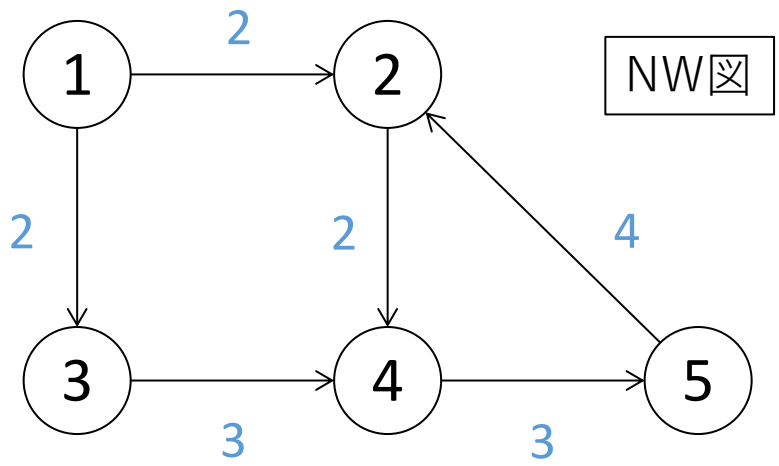
↔

all-or-nothing 配分は，
最小費用経路へ全需要配分を
全ての起点に対して実行する

Step 1	**最短経路探索** 起点 r から他のすべてのノードへの最小交通費用 $c(i)$ を計算
Step 2	**全リンクについてリンク尤度 $L[i \to j]$ を計算** $$L[i \to j] = \begin{cases} \exp[\theta\{c(j) - c(i) - t_{ij}\}] & for\ c(i) < c(j) \\ 0 & otherwise \end{cases}$$ t_{ij}:分布交通量
Step 3	**前進処理** 起点 r から $c(i)$ の値の昇順(r から近い順)にノードを考え， 各ノード i から流出するリンクのリンクウェイト $W[i \to j]$ を計算 $$W[i \to j] = \begin{cases} L[i \to j] & for\ i = r \\ L[i \to j] \sum_{m \in I_i} W[m \to i] & otherwise \end{cases}$$ I_i:ノード i に流入するリンクの始点集合
Step 4	**後進処理** $c(i)$ の値の降順にノードを考え，各ノード j に流入するリンクの交通量 x_{ij} を計算 $$x_{ij} = \left(q_{rj} + \sum_{m \in O_j} x_{jm} \right) \frac{W[i \to j]}{\sum_{m \in I_i} W[m \to j]}$$

(Dialアルゴリズム：SUEにおける配分)

- **Dialのアルゴリズム**$(\theta = 1)$



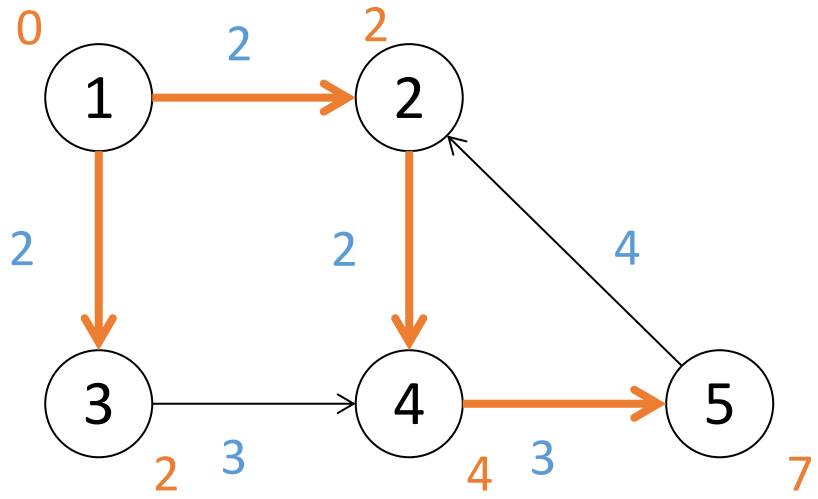
NW図

値はリンクコストに対応

OD表

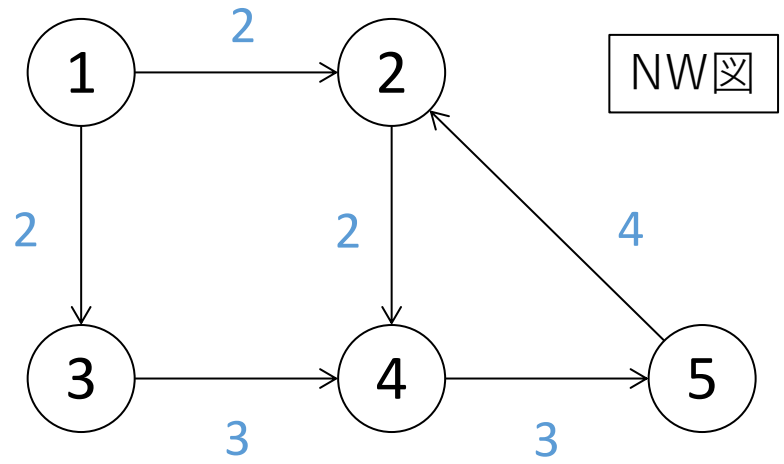
O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

Step 1	**最短経路探索** 起点 $r = 1$ から他のすべてのノードへの最小交通費用 $c(i)$ を計算



(Dialアルゴリズム：SUEにおける配分)

- **Dialのアルゴリズム**$(\theta = 1)$



値はリンクコストに対応

OD表

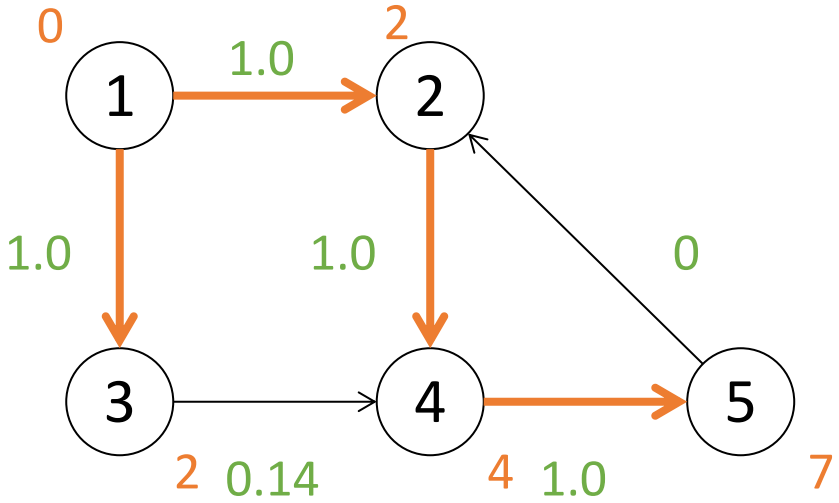
O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

Step 2

全リンクについて
リンク尤度 $L[i \to j]$ を計算

$$L[i \to j] = \begin{cases} \exp[\theta\{c(j) - c(i) - t_{ij}\}] & c(i) < c(j) \\ 0 & otherwise \end{cases}$$

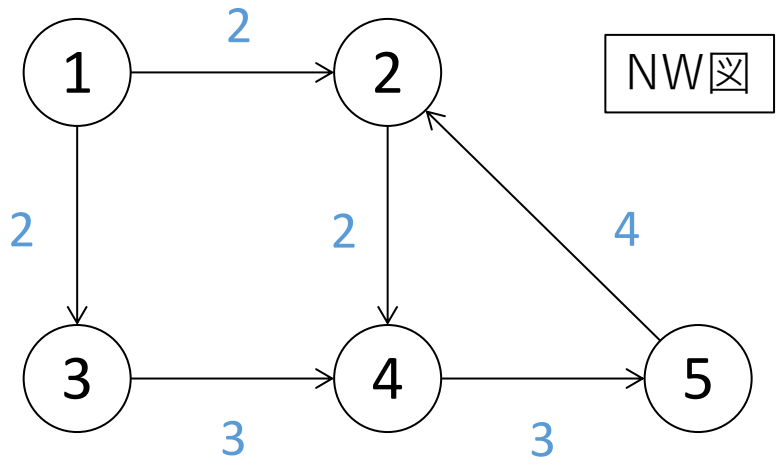
最短経路上は尤度1，後戻りする経路は0



$$L[3 \to 4] = \exp[1 * \{c(4) - c(3) - t_{34}\}]$$
$$= \exp[4 - 2 - 4]$$
$$= \exp[-2] = 0.14$$

(Dialアルゴリズム：SUEにおける配分)

- **Dialのアルゴリズム**$(\theta = 1)$



値はリンクコストに対応

OD表

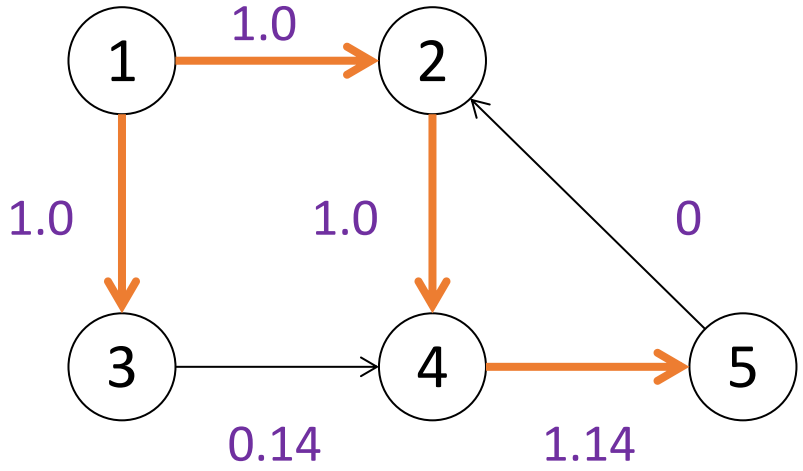
O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

Step 3

前進処理

起点 1 から $c(i)$ の値の昇順にノードを考え，各ノード i から流出するリンクのリンクウェイト $W[i \to j]$ を計算

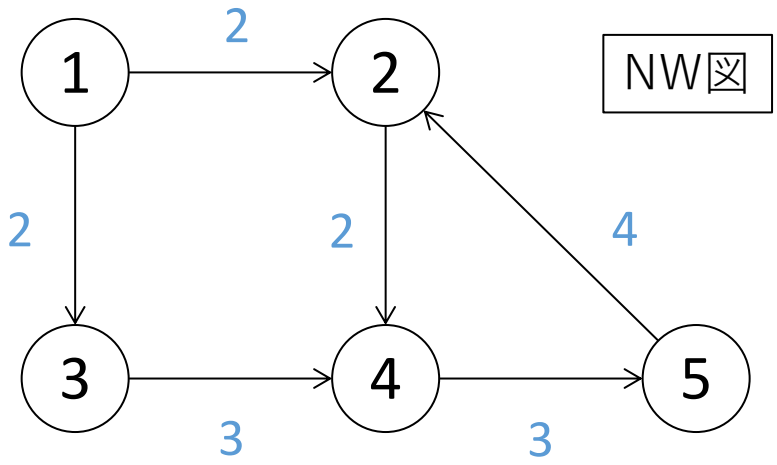
$$W[i \to j] = \begin{cases} L[i \to j] & for\ i = r \\ L[i \to j] \sum_{m \in I_i} W[m \to i] & otherwise \end{cases}$$



$$W[4 \to 5] = L[4 \to 5](W[3 \to 4] + W[2 \to 4])$$
$$= 1.0 * (1 + 0.14) = 1.14$$

(Dialアルゴリズム：SUEにおける配分)

- **Dialのアルゴリズム**$(\theta = 1)$



値はリンクコストに対応

OD表

O/D	n1	n2	n3	n4	n5
n1	0	4	5	2	3
n2	3	0	4	1	6
n3	3	1	0	4	2
n4	2	4	2	0	2
n5	1	4	3	5	0

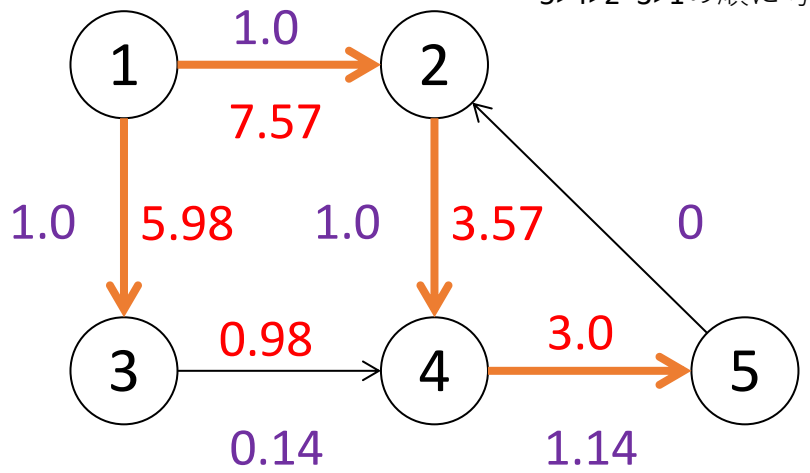
Step 4

後進処理

$c(i)$ の値の降順にノードを考え，各ノード j に流入するリンクの交通量 x_{ij} を計算

$$x_{ij} = \left(q_{rj} + \sum_{m \in O_j} x_{jm}\right) \frac{W[i \to j]}{\sum_{m \in I_i} W[m \to j]}$$

5>4>2=3>1の順に考える



計算例

$$x_{45} = \left(q_{15} + \sum_{m \in O_j} x_{5m}\right) \frac{W[4 \to 5]}{\sum_{m \in I_i} W[4 \to 5]}$$

$$= (q_{15} + 0) \frac{W[4 \to 5]}{W[4 \to 5]} = 3.0$$

$$x_{24} = (q_{14} + x_{45}) \frac{W[2 \to 4]}{W[2 \to 4] + W[3 \to 4]} = 3.57$$

(Dialアルゴリズムとロジットモデル)

- **等価性の証明**

一つのODペアに注目し、 k 番目のパスが $r \to A \to B \to \cdots \to Y \to Z \to s$ のとき、Dialのアルゴリズムでこのパスが選択される確率は

$$P_k = \frac{W[Z \to s]}{\sum_m W[m \to s]} \frac{W[Y \to Z]}{\sum_m W[Y \to Z]} \cdots \frac{W[A \to B]}{\sum_m W[A \to B]} \frac{W[r \to A]}{\sum_m W[r \to A]}$$

リンクウェイトはリンク尤度によって以下のように定義されていた

$$W[i \to j] = \begin{cases} L[i \to j] & for\ i = r \\ L[i \to j] \sum_{m \in I_i} W[m \to j] & otherwise \end{cases}$$

よって $P_k = L[Z \to s]L[Y \to Z] \ldots L[A \to B]L[r \to A] / \sum_m W[m \to s]$

(Dialアルゴリズムとロジットモデル)

- **等価性の証明**

$$P_k = L[Z \to s]L[Y \to Z] \dots L[A \to B]L[r \to A] / \sum_m W[m \to s]$$

また，リンク尤度の定義を分子に代入すると

$$L[i \to j] = \begin{cases} \exp[\theta\{c(j) - c(i) - t_{ij}\}] & for\ c(i) < c(j) \\ 0 & otherwise \end{cases}$$

$$\begin{aligned} & L[Z \to s]L[Y \to Z] \dots L[A \to B]L[r \to A] \\ & = \exp[\theta\{c(s) - c(Z) - t_{Zs}\}]\exp[\theta\{c(Z) - c(Y) - t_{YZ}\}] \\ & \dots \exp[\theta\{c(B) - c(A) - t_{AB}\}] \exp[\theta\{c(A) - c(r) - t_{rA}\}] \\ & = \exp[\theta\{c(s) - (t_{rA} + t_{AB} + \dots + t_{YZ} + t_{Zs})] \\ & = \exp[\theta\{c(s) - c_k\}] \end{aligned} \quad \cdots ①$$

フロー保存則は，

$$\sum_k P_k = \sum_k \frac{\exp[\theta\{c(s) - c_k\}]}{\sum_m W[m \to s]} = 1 \text{ より，} \quad \frac{\exp[\theta c(s)]}{\sum_m W[m \to s]} = \frac{1}{\sum_k \exp[-\theta c_k]} \quad \cdots ②$$

(Dialアルゴリズムとロジットモデル)

- **等価性の証明**

$$L[Z \to s]L[Y \to Z] \dots L[A \to B]L[r \to A] = \exp[\theta\{c(s) - c_k\}] \quad \cdots ①$$

$$\frac{\exp[\theta c(s)]}{\sum_m W[m \to s]} = \frac{1}{\sum_k \exp[-\theta c_k]} \quad \cdots ②$$

①と②より，

$$P_k = L[Z \to s]L[Y \to Z] \dots L[A \to B]L[r \to A] / \sum_m W[m \to s]$$

$$= \frac{\exp[c(s)]\exp[-\theta c_k]}{\sum_m W[m \to s]} = \frac{\exp[-\theta c_k]}{\sum_k \exp[-\theta c_k]}$$

：ロジット型の選択確率の形

2.3 Frank-Wolfe法

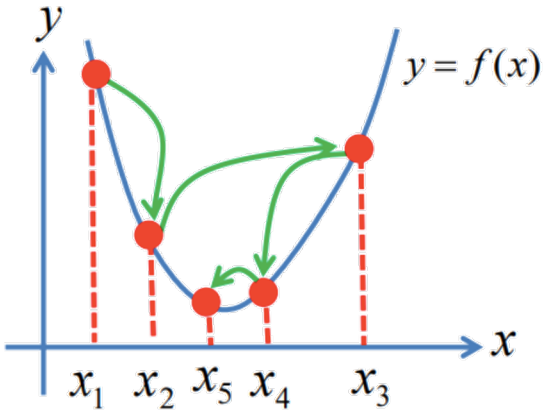
- **凸集合が保証された非線形最適化問題の解法**

凸計画問題のアルゴリズムの概略

【ステップA】降下方向の探索
：どの方向に向かえば目的関数が降下するか？

【ステップB】ステップサイズの一次元探索
：どこまで進めるか？

(イメージ)



【ステップA】と【ステップB】を繰り返すことで最小値を見つける

2.3 Frank-Wolfe法

- **凸集合が保証された非線形最適化問題の解法**

【ステップA】降下方向の探索
：どの方向に向かえば目的関数が降下するか？

n 回目の探索の結果，点 $\boldsymbol{x}^{(n)}$ が得られたとする．
降下方向ベクトルを $\boldsymbol{d} = \boldsymbol{y} - \boldsymbol{x}^{(n)}$ によって与える．
方向ベクトル $\boldsymbol{y}$ は，主問題の目的関数の点 $\boldsymbol{x}^{(n)}$ における一次または二次のTaylor展開によって得られた近似問題を最適化問題として再定義することによって求める．

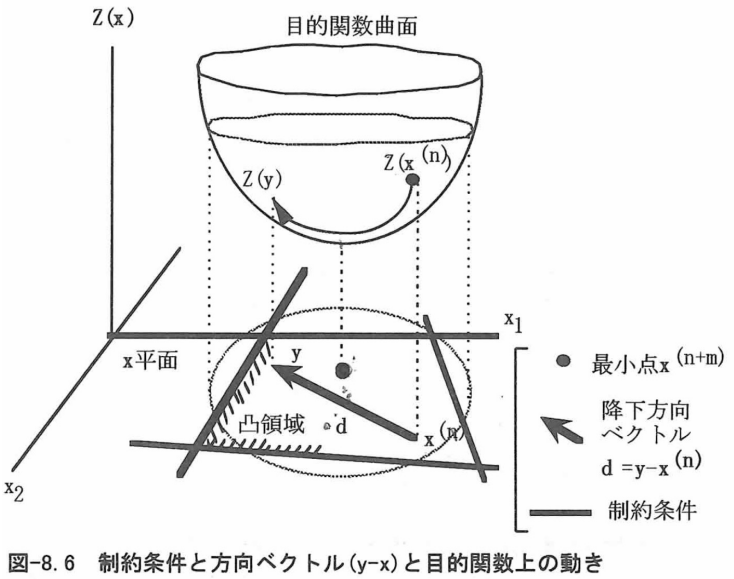


図-8.6　制約条件と方向ベクトル(y-x)と目的関数上の動き

【ステップB】ステップサイズの一次元探索
：どこまで進めるか？

$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}$ で与えられる $\boldsymbol{x}^{(n+1)}$ をもともとの目的関数に代入し，目的関数 $Z(\boldsymbol{x}^{(n+1)})$ を最小化する $\boldsymbol{x}^{(n+1)}$ を得る．このとき，$\boldsymbol{x}^{(n)}$ と $\boldsymbol{y}$ が既に定数であるため，$Z(\alpha)$ としてパラメータ α の一つの変数を持つだけの一次元最適化となる．
＝黄金分割法等の多くの解法が用意されている形として変換される．

2.3 Frank-Wolfe法

- **Frank-Wolfe法の概要**

【UE/FD-Primal】

$$\min Z(\boldsymbol{x}) = \sum_a \int_0^{x_a} t_a(\boldsymbol{\omega})\, d\omega$$

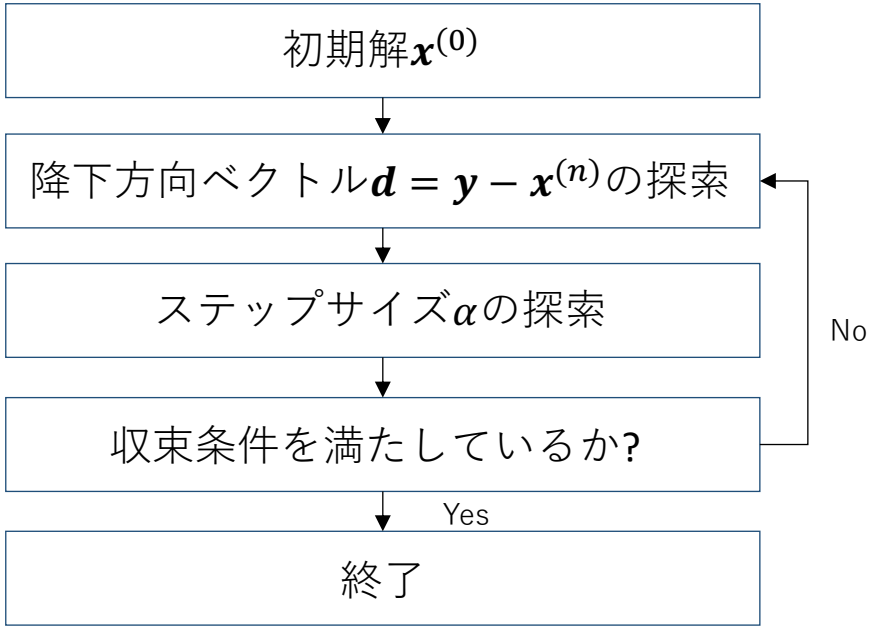
$s.t$

$$\sum_k f_k^{rs} - q_{rs} = 0 \qquad \forall rs \in \Omega$$

$$x_a = \sum_{rs\in\Omega} \sum_{k\in K_{rs}} f_k^{rs} \delta_{a,k}^{rs} \qquad \forall a \in A,\ \forall rs \in \Omega$$

$$f_k^{rs} \geq 0 \qquad \forall rs \in \Omega$$

$$x_a \geq 0 \qquad \forall a \in A$$



特徴
◎必要な記憶容量が少ない
◎手順が簡単
×収束が進むと解の近傍で”ジグザグ運動” → **収束スピードが緩慢**

→改良手法として
- **打ち切り二次計画法**
- **Simplicial Decomposition法**

がある(発展学習)

2.3 Frank-Wolfe法

- **降下方向ベクトル $\boldsymbol{d} = \boldsymbol{y} - \boldsymbol{x}^{(n)}$ の探索**

n 回目の探索の結果与えられた点 $\boldsymbol{x}^{(n)}$ において，目的関数を線形近似する．

$$
\begin{aligned}
Z_p(\boldsymbol{y}) \cong Z'(\boldsymbol{y}) \quad &= Z_p\left(\boldsymbol{x}^{(n)}\right) + \nabla Z_p\left(\boldsymbol{x}^{(n)}\right)^T \left(\boldsymbol{y} - \boldsymbol{x}^{(n)}\right) \\
&= Z_p\left(\boldsymbol{x}^{(n)}\right) + \sum_{a \in A} \left(y_a - x_a^{(n)}\right) \frac{\partial Z_p\left(\boldsymbol{x}^{(n)}\right)}{\partial x_a^{(n)}} \\
&= Z_p\left(\boldsymbol{x}^{(n)}\right) + \sum_{a \in A} \left(y_a - x_a^{(n)}\right) t_a\left(x_a^{(n)}\right) \\
&= Z_p\left(\boldsymbol{x}^{(n)}\right) - \sum_{a \in A} x_a^{(n)} t_a\left(x_a^{(n)}\right) + \sum_{a \in A} y_a t_a\left(x_a^{(n)}\right)
\end{aligned}
$$

$Z_p\left(\boldsymbol{x}^{(n)}\right)$ と $\sum_{a \in A} x_a^{(n)} t_a\left(x_a^{(n)}\right)$ は定数

よって以下の補助問題を得る

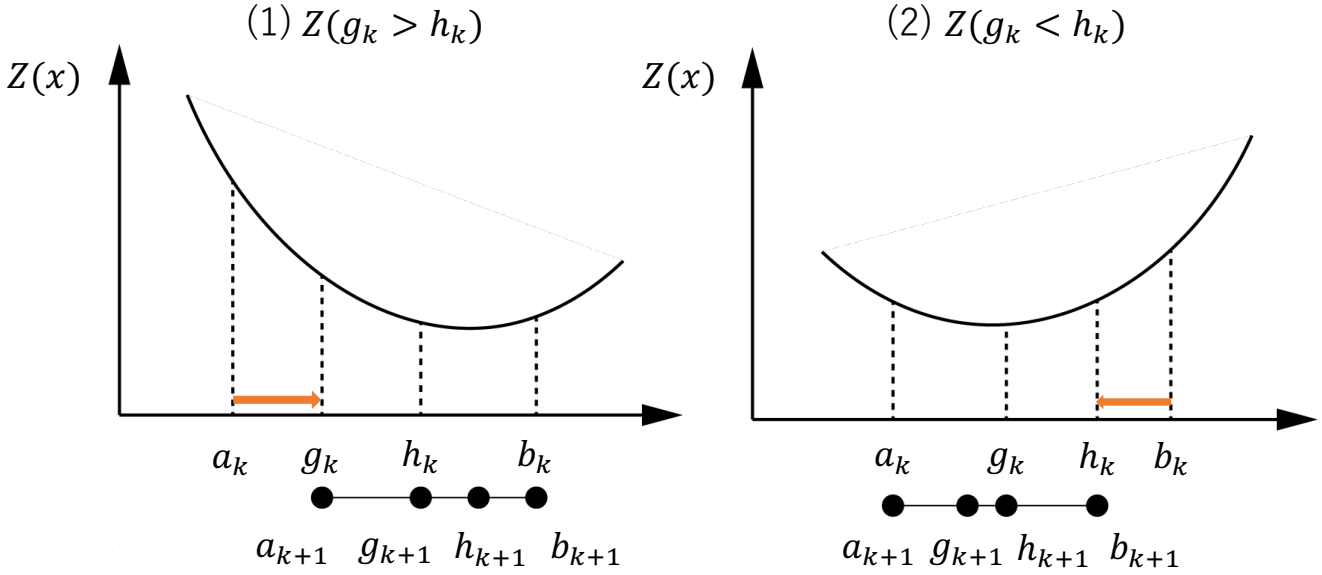
$$
\min Z'(\boldsymbol{y}) = \sum_{a \in A} y_a t_a\left(x_a^{(n)}\right) \qquad s.t. \sum_k f_k^{rs} - q_{rs} = 0, y_a = \sum_k \sum_{rs \in \Omega} \delta_{a,k}^{rs} f_k^{rs}
$$

2.3 Frank-Wolfe法

- **一次元探索の手法：黄金分割法**

モチベーション：
一変数の凸関数 $Z(\boldsymbol{x})$ の区間 $[a, b]$ における最小値を求めたい.

概要：
最小点を含む k 回目の閉区間 $[a_k, b_k]$ から，縮小率 s だけ小さく，かつ最小値が存在する次の閉区間 $[a_{k+1}, b_{k+1}]$ を順次列挙して最小点を探索する.

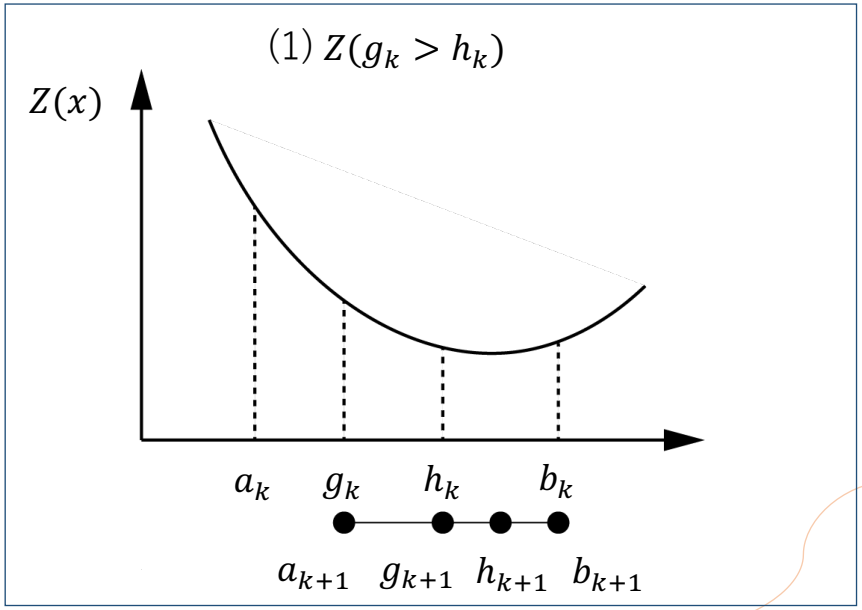


内点 g_k, h_k を
取りながら探索する

最小値は
(1)のとき $[g_k, b_k]$
(2)のとき $[h_k, a_k]$
にあることが分かる.

2.3 Frank-Wolfe法

- **一次元探索の手法：黄金分割法**



内点 g_k, h_k を，
$b_k - g_k = h_k - a_k = s(b_k - a_k)$
を満たすようにとると，

① $g_k = b_k - s(b_k - a_k)$
② $h_k = a_k + s(b_k - a_k)$

(1) $Z(g_k > h_k)$ の場合，$k+1$ 回目の点を左図のようにとれる．

③ $a_{k+1} = g_k,\ g_{k+1} = h_k,\ b_{k+1} = b_k$

※ k 回目の3点を $k+1$ 回目で利用可能

$g_{k+1} = b_{k+1} - s(b_{k+1} - a_{k+1})$ に③を代入すると，　④ $h_k = b_k - s(b_k - g_k)$

④に①，②を代入すると，
$$a_k + s(b_k - a_k) = b_k - s * s(b_k - a_k)$$
$$s^2(b_k - a_k) + s(b_k - a_k) - (b_k - a_k) = 0$$
⑤ $s^2 + s - 1 = 0\ (\because a_k \neq b_k)$

⑤を解くと，$s = \frac{\sqrt{5}-1}{2} = 0.618\ldots$　　c.f. 黄金数 $\frac{\sqrt{5}+1}{2} = \frac{1}{s}$

2.3 Frank-Wolfe法

- **補助問題を解いて y を得る**

【補助問題】

$$\min Z'(\boldsymbol{y}) = \sum_{a \in A} y_a t_a\left(x_a^{(n)}\right) \qquad s.t. \sum_k f_k^{rs} - q_{rs} = 0, y_a = \sum_k \sum_{rs \in \Omega} \delta_{a,k}^{rs} f_k^{rs}$$

・・・ $t_a\left(x_a^{(n)}\right)$ のもとで走行時間を最小化する y を求める問題と解釈可能.

$t_a\left(x_a^{(n)}\right)$ は定数のため， $t_a\left(x_a^{(n)}\right)$ のリンク所要時間で求められる最短経路に
全てのOD交通量を流す**all-or-nothing配分**をすることで，
$Z'(\boldsymbol{y})$ を最小化する y を得られる.

⇩

降下方向ベクトル $\boldsymbol{d} = \boldsymbol{y} - \boldsymbol{x}^{(n)}$ を求める

$$Z_p(\boldsymbol{y}) \cong Z'(\boldsymbol{y}) = Z_p\left(\boldsymbol{x}^{(n)}\right) + \nabla Z_p\left(\boldsymbol{x}^{(n)}\right)^T \left(\boldsymbol{y} - \boldsymbol{x}^{(n)}\right)$$

目的関数 $Z_p(\boldsymbol{y})$ の点 $\boldsymbol{x}^{(n)}$ における最急勾配に沿った降下方向ベクトル

2.3 Frank-Wolfe法

- **Frank-Wolfe法のアルゴリズム**

Step 1	**初期実行可能解の設定** 交通量 0 における all-or-nothing配分によって，収束回数 $n=1$ として初期実行可能解となるリンク交通量$\left\{x_a^{(n)}\right\}$を与える.
Step 2	**リンクコストの更新** $\left\{x_a^{(n)}\right\}$に対する所要時間$\left\{t_a\left(x_a^{(n)}\right)\right\}$を計算
Step 3	**降下方向の探索** 最短経路探索によって各OD間の最短経路を求め，その最短経路にall-or-nothing配分によって全交通需要を負荷し$\{y_a\}$を求める.
Step 4	**ステップサイズの一次元探索** 交通量の更新：$x_a^{(n+1)}=x_a^{(n)}+\alpha^{(n)}d_a^{(n)}=x_a^{(n)}+\alpha^{(n)}(y_a-x_a^{(n)})$ $\min_{0\le\alpha\le1}\sum_a\int_0^{x_a^{(n+1)}}t_a(w)\,dw$　　α についての一次元探索 により，ステップサイズ$\alpha^{(n)}$とリンク交通量$x_a^{(n+1)}$を求める.

2.3 Frank-Wolfe法

- **Frank-Wolfe法のアルゴリズム（続き）**

Step 5

収束判定

予め設定した ε_1, ε_2, K に対して

(a) $\left(x_a^{(n+1)} - x_a^{(n)}\right) t_a\left(x_a^{(n)}\right) \leq \varepsilon_1$　　降下方向ベクトルの傾き

(b) $\max_a \left|\left(x_a^{(n+1)} - x_a^{(n)}\right) / x_a^{(n)}\right| \leq \varepsilon_2$　　リンク交通量変化

(c) $n > K$　　収束計算回数

のいずれかを満たすならば計算終了.
そうでない場合 $n = n + 1$ としてStep 2へ戻る.

※リンクに容量制約 $x_a < C_a$ がある問題では，ステップサイズ探索時に α に関する以下の制約条件を追加する.

（BPR関数等の$x_a^{(n)}$に対して”緩やかな変化”をする関数ではなく，Davidson関数のようにリンク交通量が容量に近づくとリンクコストが無限大になるような場合）

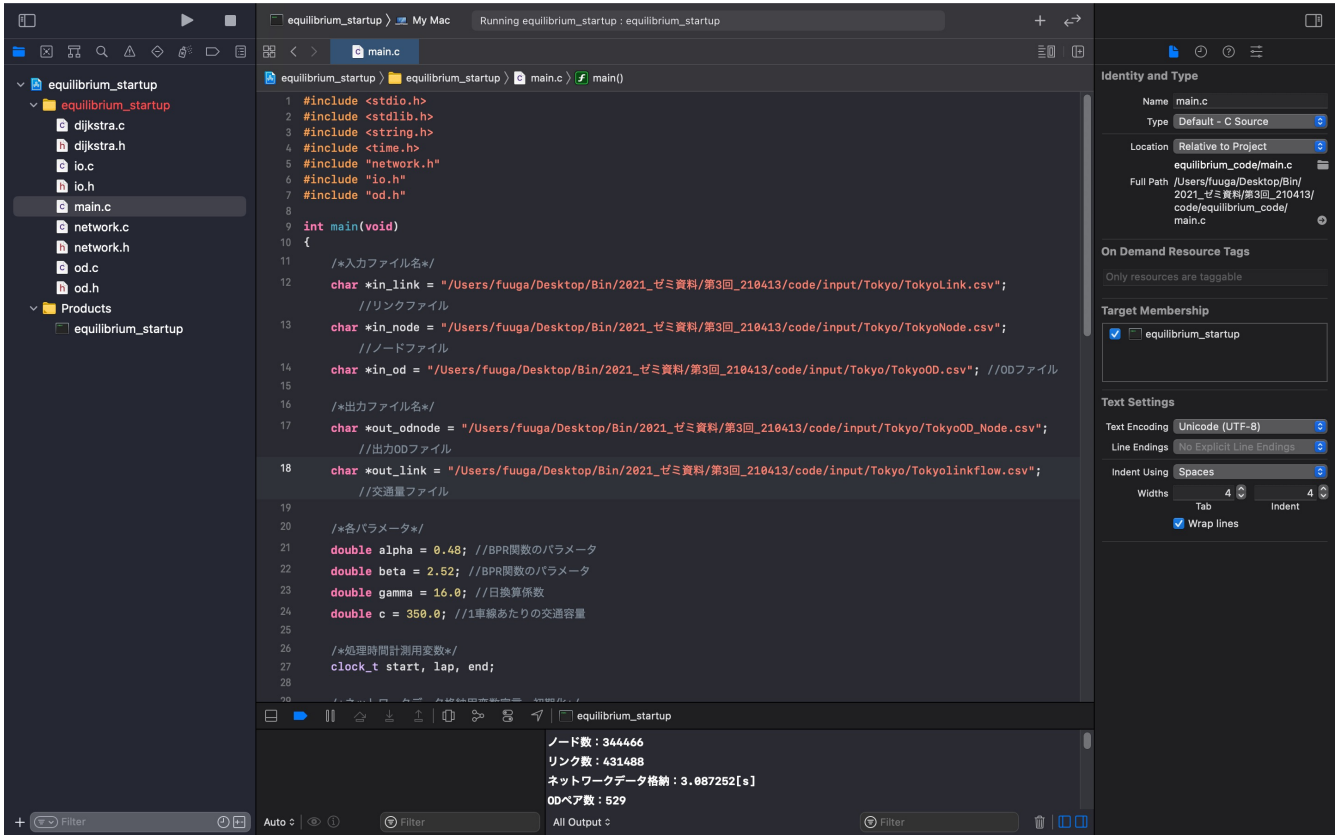
$$\alpha < \alpha_{\max} = \min_{x_a < y_a} \left\{\left(C_a - x_a^{(n)}\right) / \left(y_a - x_a^{(n)}\right)\right\}$$

3. 均衡配分コードの実装に向けて

3.1 Cの導入

- **プログラム**

均衡配分のコードはC言語で書かれています.
(みなさんMacのようなので) XcodeやCLionがおすすめです.

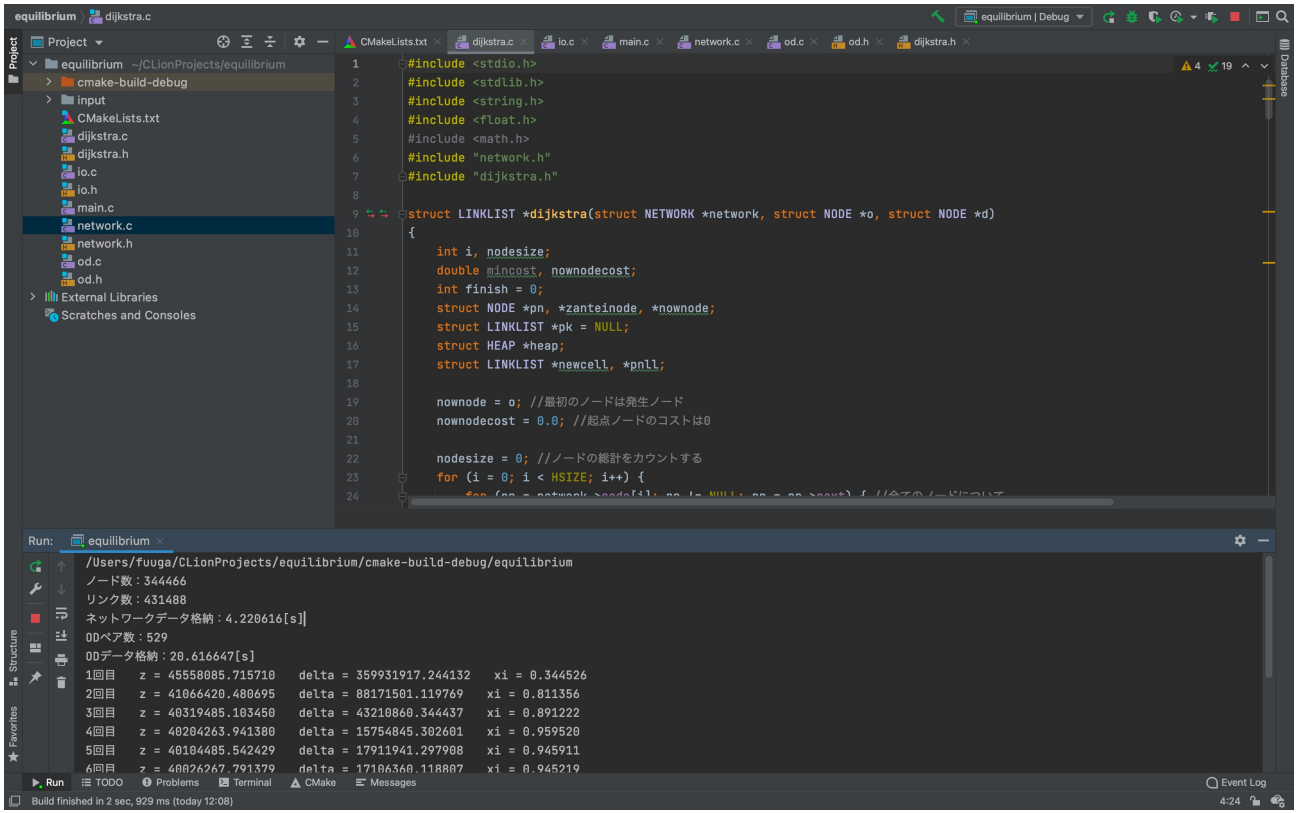


Xcode

3.1 Cの導入

- **プログラム**

均衡配分のコードはC言語で書かれています.
(みなさんMacのようなので) XcodeやCLionがおすすめです.



CLion

3.1 Cの導入

- **実際のネットワークデータをつかってみよう**

東京都市圏のネットワークデータ

TokyoNode.csv / TokyoLink.csv

ノード数：344466
リンク数：431488

	A	B	C
1	nodeID	緯度	経度
2	100	35.5032413	139.500246
3	101	35.5033163	139.500246
4	200	35.5032413	139.504371
5	201	35.5033163	139.500246
6	300	35.5032415	139.514832
7	301	35.5035663	139.504571
8	302	35.503733	139.504608
9	303	35.5038746	139.504583
10	401	35.5038746	139.504583
11	402	35.503733	139.504608

Node

	A	B	C	D	E
1	LinkID	起点ノードID	終点ノードID	車線数	最高速度
2	1	100	101	2	30
3	2	101	117500	2	30
4	3	117500	201	2	30
5	4	201	100	2	30
6	5	200	301	2	30
7	6	301	302	2	30
8	7	302	303	2	30
9	8	303	120500	2	30
10	9	120500	401	2	30
11	10	401	402	2	30

Link

3.1 Cの導入

- **実際のネットワークデータをつかってみよう**

東京都市圏の自動車OD表（TokyoOD.csv）

	A	B	C	D	E
1	Olat	Olon	Dlat	Dlon	number
2	35.69189	139.75963	35.69189	139.75963	77421
3	35.69189	139.75963	35.676075	139.77673	28681
4	35.69189	139.75963	35.65529	139.73944	31224
5	35.69189	139.75963	35.700584	139.71861	15954
6	35.69189	139.75963	35.717148	139.74731	10119
7	35.69189	139.75963	35.71435	139.78577	8497
8	35.69189	139.75963	35.706894	139.81201	3673
9	35.69189	139.75963	35.671326	139.81288	8375
10	35.69189	139.75963	35.611454	139.72455	1907
11	35.69189	139.75963	35.629833	139.68925	1733

Olat (latitude)：出発地の経度

Olon (longitude)：出発地の緯度

トリップ数

3.1 Cの導入

- **主要なパラメータ**

BPR関数の各種パラメータの設定が必要.

【BPR関数】

$$t_a(x_a) = t_{a0}\left\{1 + \alpha\left(\frac{x_a}{C_a}\right)^{\beta}\right\}$$

t_a …リンクaの旅行時間

t_{a0} …リンクaの自由旅行時間

x_a …リンクaの時間交通量(台/時)

C_a …リンクaの時間交通容量(台/時)

α, β …パラメータ

```
/*各パラメータ*/
    double alpha = 0.48; //BPR関数のパラメータ
    double beta = 2.52; //BPR関数のパラメータ
    double gamma = 16.0; //日換算係数
    double c = 350.0; //1車線あたりの交通容量
```

Link

	A	B	C	D	E
1	LinkID	起点ノードID	終点ノードI	車線数	最高速度
2	1	100	10	2	30
3	2	101	11750	2	30
4	3	117500	20	2	30
5	4	201	10	2	30
6	5	200	30	2	30
7	6	301	30	2	30
8	7	302	30	2	30
9	8	303	12050	2	30
10	9	120500	40	2	30
11	10	401	40	2	30

$C_a = l_a * \text{gamma} * \text{c}$

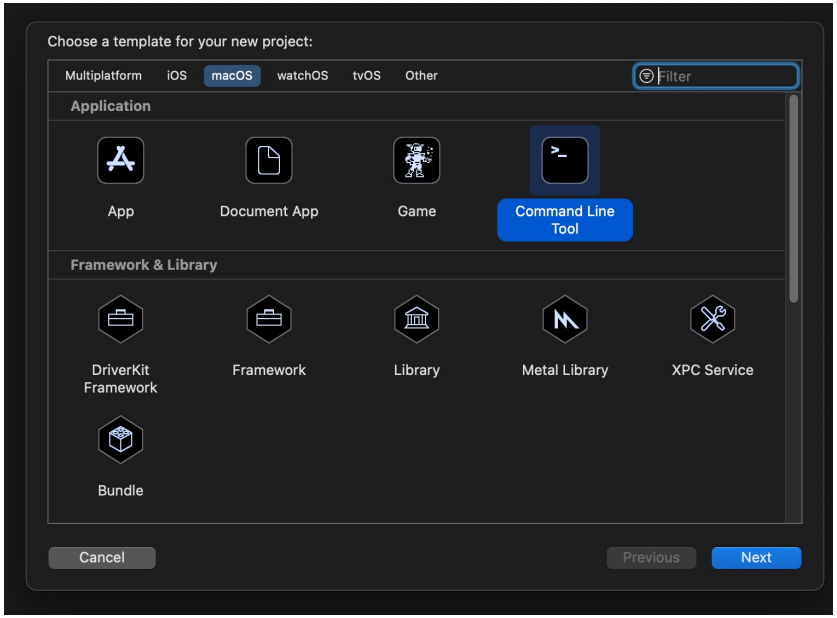
リンク容量＝車線数×日換算係数×1車線あたりの交通容量

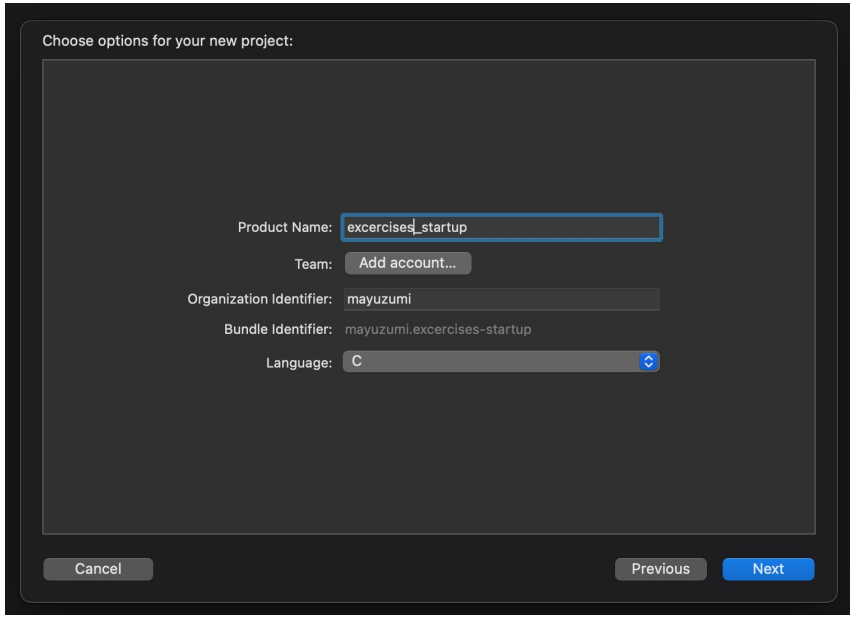
3.2 C言語の基本

- **Cのプログラム（基礎知識）**

IDE(ここではXcodeについて説明)にてプロジェクトを立ち上げ

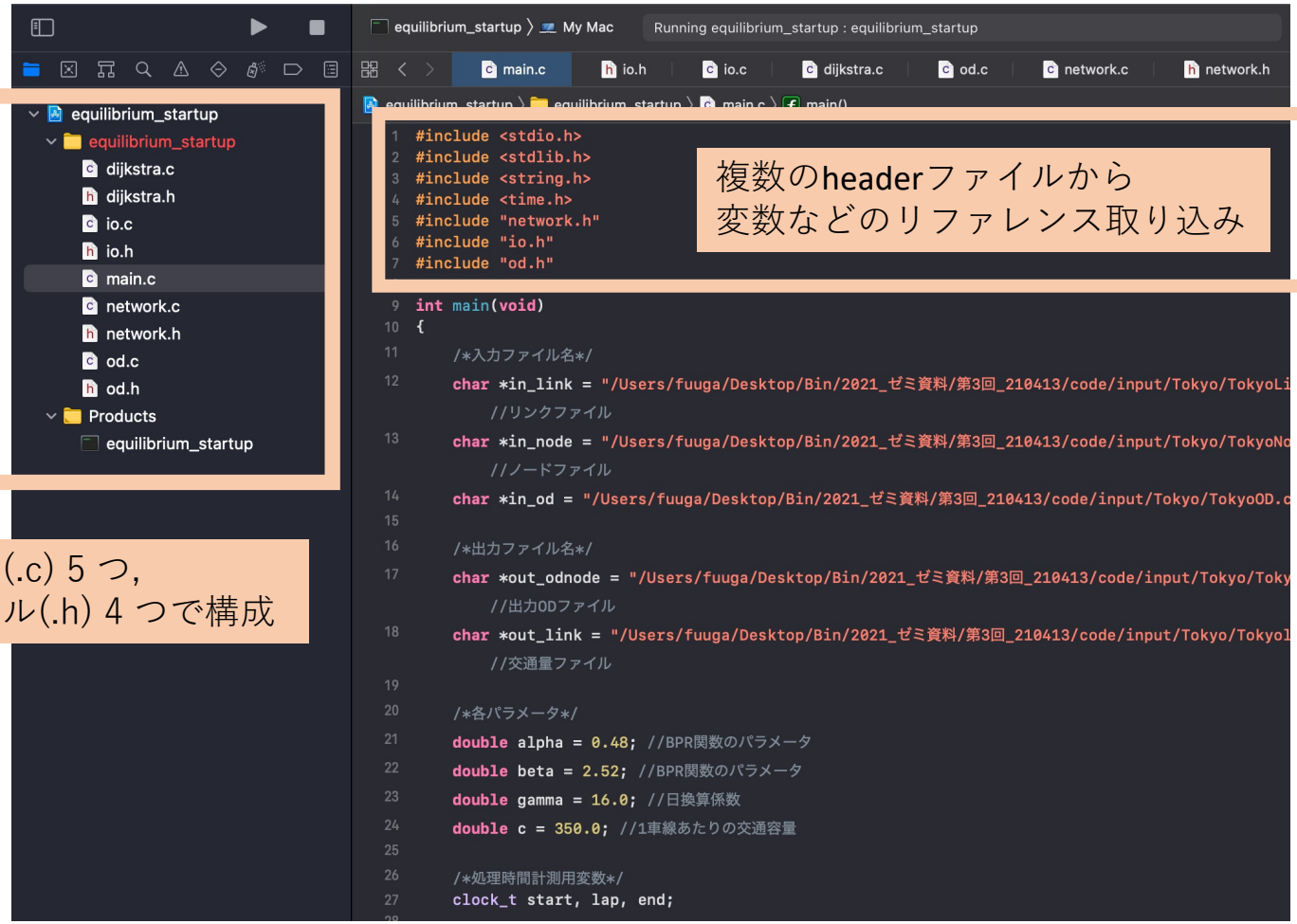
メニューバーから，File>New>Project でCommand Line ToolからC言語を選択





3.2 C言語の基本

- **Cのプログラム（基礎知識）**



複数のheaderファイルから
変数などのリファレンス取り込み

ソースファイル(.c) 5 つ,
ヘッダーファイル(.h) 4 つで構成

3.2 C言語の基本

- **Cのプログラム（基礎知識）**

```
/*各パラメータ*/
    double alpha = 0.48; //BPR関数のパラメータ
    double beta = 2.52; //BPR関数のパラメータ
    double gamma = 16.0; //日換算係数
    double c = 350.0; //1車線あたりの交通容量
```

変数の設定の仕方：　**型名　識別子;**

主な３つの data type

int num; … 整数型（short int）
char c; … 文字型
double db; … 浮動小数点型(8byte)

3.2 C言語の基本

- **Cのプログラム（基礎知識）**

関数の設定の仕方：

```
戻り値の型 関数名(引数リスト)
{
   文;
    ・・・
   return 式;
}
```

```
/*BPR関数*/
double bpr(double flow, double freecost, double capacity, double
alpha, double beta)
{
    return freecost * (1.0 + alpha * pow((flow / capacity), beta));
}
```

関数の呼び出し：

```
関数名(引数リスト);
```

```
//BPR関数でリンクコストを計算
pl->cost = bpr(pl->flow, pl->freecost, pl->capacity, alpha, beta);
```

3.2 C言語の基本

- **Cのプログラム（基礎知識）**

配列：　**型名 配列名[要素数];**

```
//example
int centershiken[7];
char test[5] = {“J”,”E”,”M”,”S”,”H”};
```

アドレス：　**&変数名**

変数がコンピュータのメモリのどこに記憶されているか：アドレスを知ることができる.
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, f の文字を用いた16進数で通常表される.

ポインタ：　**型名 *ポインタ名;**

アドレスを格納する特殊な変数．型名は格納できるアドレスの持ち主の変数に依存する.

```
//example
int main(void)
{
        int a = 5;
        int *pA;
        pA = &a;          <- これによって *pA は &a にある変数を示す
}
```

3.2 C言語の基本

- **Cのプログラム（基礎知識）**

構造体型：

```
struct 構造体名 {
   型名 識別子;
   型名 識別子;
    ・・・
};
```

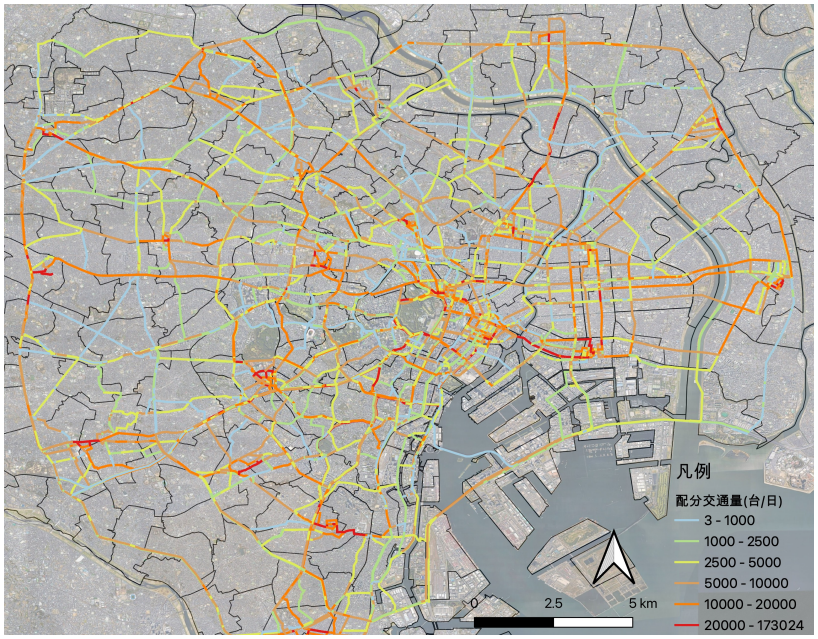
複数の変数をまとめるもの

```
/*ノードの情報を格納するNODE構造体型*/
struct NODE {
    char ID[40]; //ノードID
    double lat; //緯度
    double lon; //経度
    int state; //dijkstra法で使う状態変数
    double mincost; //dijkstra法で使う最小コスト
    struct LINK *beforelink; //dijkstra法で使う前リンク
    struct LINKLIST *nextlinklist; //このノードを起点とするリンク
    struct NODE *next;
};
```

4. 課題

4 課題

[1] 東京都NWで均衡配分を実行し，その結果をQGISに表現する．



初期条件を変えずに計算すると
左図のように配分結果が出ます．

⓪配分計算の実行

①配布したJavaのコードを実行し，
ノードとリンクを結合させた
csvファイルを作成する．

②①をGIS上でXY座標に落として
線データ化する

③均衡配分結果である各リンクの交通
量を，②のリンクIDと紐づけて可視化

可視化の詳細については 共有ファイル内の「均衡配分のマニュアル」を参考にしてください．

[2] 均衡配分におけるパラメータや対象ネットワーク，分布交通量を興味のあるシナリオに沿って変化させて，均衡配分結果を考察してみる．

参考資料

・参考書

土木学会．(1998)．交通ネットワークの均衡分析ー最新の理論と解法

Yosef Sheffi．(1985)． *Urban Transportation Networks: Equilibrium Analysis With Mathematical Programming Methods*.

交通計画システム研究会．(2006)．都市の交通計画ー総合交通体系調査と交通需要の分析・予測.

・羽藤研究室スタートアップゼミ・理論談話会資料　多数

http://bin.t.u-tokyo.ac.jp/startup13/test/0626_shibahara.pdf

http://bin.t.u-tokyo.ac.jp/startup15/file/4-3.pdf

http://bin.t.u-tokyo.ac.jp/startup15/file/5-1.pdf

http://bin.t.u-tokyo.ac.jp/startup16/file/4-2.pdf

http://bin.t.u-tokyo.ac.jp/startup16/file/6-1.pdf

http://bin.t.u-tokyo.ac.jp/startup16/file/6-2.pdf

http://bin.t.u-tokyo.ac.jp/startup17/file/2-2.pdf

http://bin.t.u-tokyo.ac.jp/startup19/file/slide6.pdf

http://bin.t.u-tokyo.ac.jp/startup20/file/slide4.pdf

http://bin.t.u-tokyo.ac.jp/startup13/test/%E3%82%B9%E3%82%BF%E3%83%BC%E3%83%88%E4%BB%8A%E6%B3%892.pdf

http://bin.t.u-tokyo.ac.jp/kaken/pdf/traffic_assign%20tutrial%20for%20C.pdf

・東工大　朝倉・福田研究室
http://www.plan.cv.titech.ac.jp/fukudalab/e7ceb1ff3218f0c026d3d26a65b69a6f336b64aa.pdf

http://www.plan.cv.titech.ac.jp/fukudalab/2017basicseminar/14_01.pdf