第9回　基礎ゼミ
ネットワークの表現と経路探索

福田研 修士1年　小川晃平

本日の内容

第2章　ネットワークの表現とリンクコスト関数

第3章　ネットワークフローが満足すべき条件

第8章　利用者均衡モデルの解法
　8.1　最短経路探索アルゴリズムとデータ構造

1節　交通ネットワークの構成要素と交通網のネットワーク表現

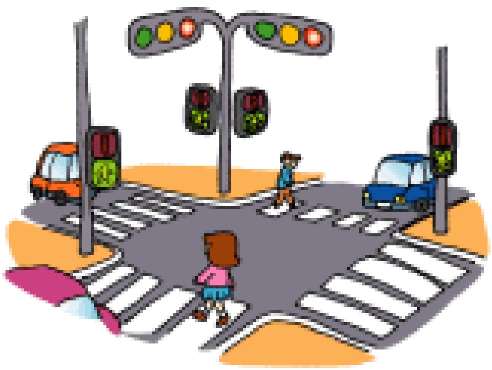
- 交通ネットワークの基本的な構成要素と交通網のネットワーク表現法
- 少し複雑な交通網のネットワーク表現

2節　リンクパフォーマンス関数

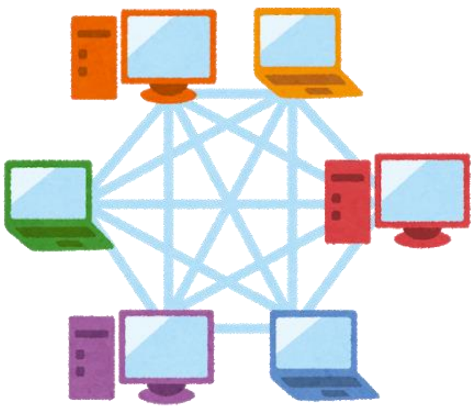
- 代表的なリンクパフォーマンス関数
- 均衡交通量配分計算方法との対応関係

1節　交通ネットワークの構成要素と交通網のネットワーク表現

物理的ネットワーク



概念的ネットワーク



いずれにせよ，ネットワークは“**点の集合**”と“**線の集合**”から成り立っている

- 点→交差点，接点
- 線→道路，情報の流れ

数学的表現

点の集合･･･ノード集合N
線の集合･･･リンク集合A

有向グラフ$G[N,A]$で表すことができる

1節　交通ネットワークの構成要素と交通網のネットワーク表現

交通ネットワークにおいては・・・

ノード集合N

- 交通需要の発生ノード集合R
- 交通需要の集中ノード集合S
- それ以外のノード集合

＊各ノードnは正の整数で対応する

リンク集合A

- 実際の道路区間
- 状況に応じた仮想的な道路区間

＊リンクの要素aは正の整数で対応する

通常，交通需要分析は分析対象地域をゾーンに分割して行われる
⇒ゾーンの中心にあるノードを“セントロイド”と呼ぶ

1節　交通ネットワークの構成要素と交通網のネットワーク表現

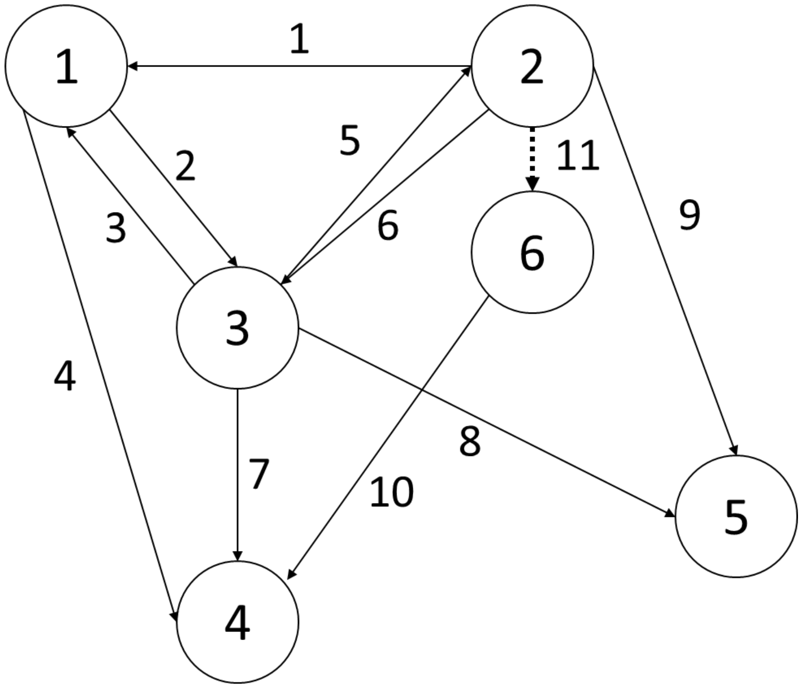
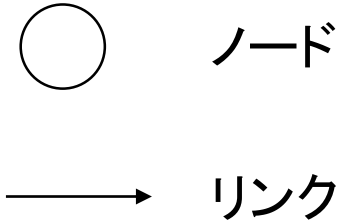


図1　道路網のネットワーク表現



O＼D	3	5	発生量
1	90	130	220
2	50	70	120
集中量	140	200	340

1 → 3の分布交通量　　総生成交通量

図1のネットワークは1～6のノードと
1～11のリンクで構成されている

リンク1～10は固有のリンク抵抗値t_aを持つ
リンク11は仮想的リンクで抵抗値は定数
ノード6は料金ゲートを表現した仮想ノード

1節　交通ネットワークの構成要素と交通網のネットワーク表現

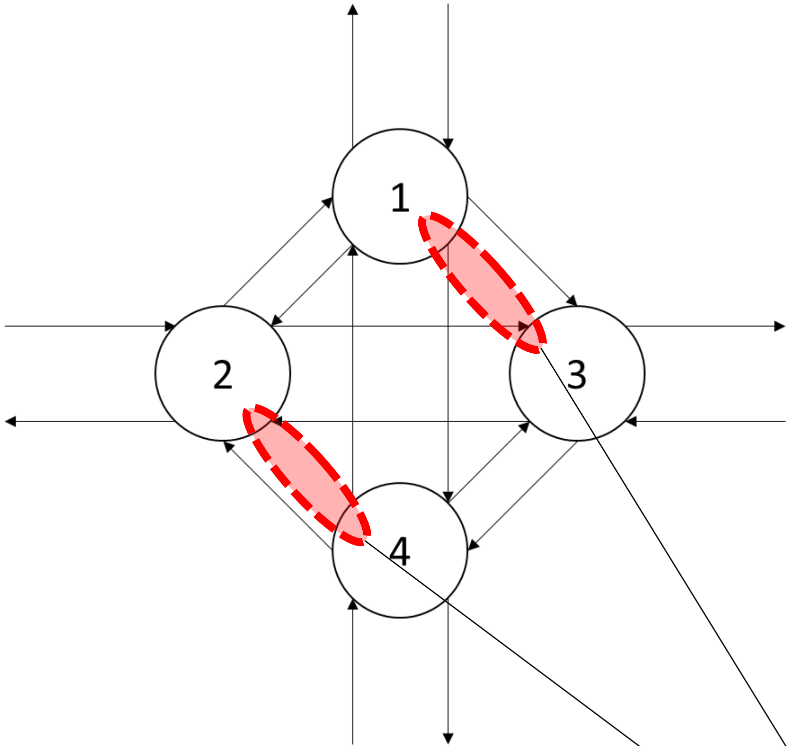


図2　平面交差点のネットワーク

交差点内に方向別リンクを設定

右・左折，直進別の
交差点遅れが表現可能
（1つのノードだけでは表現できない）

また，東西の道路から南北への右折が
規制されている状況をモデル

＊右折するリンクが存在しない

1節　交通ネットワークの構成要素と交通網のネットワーク表現

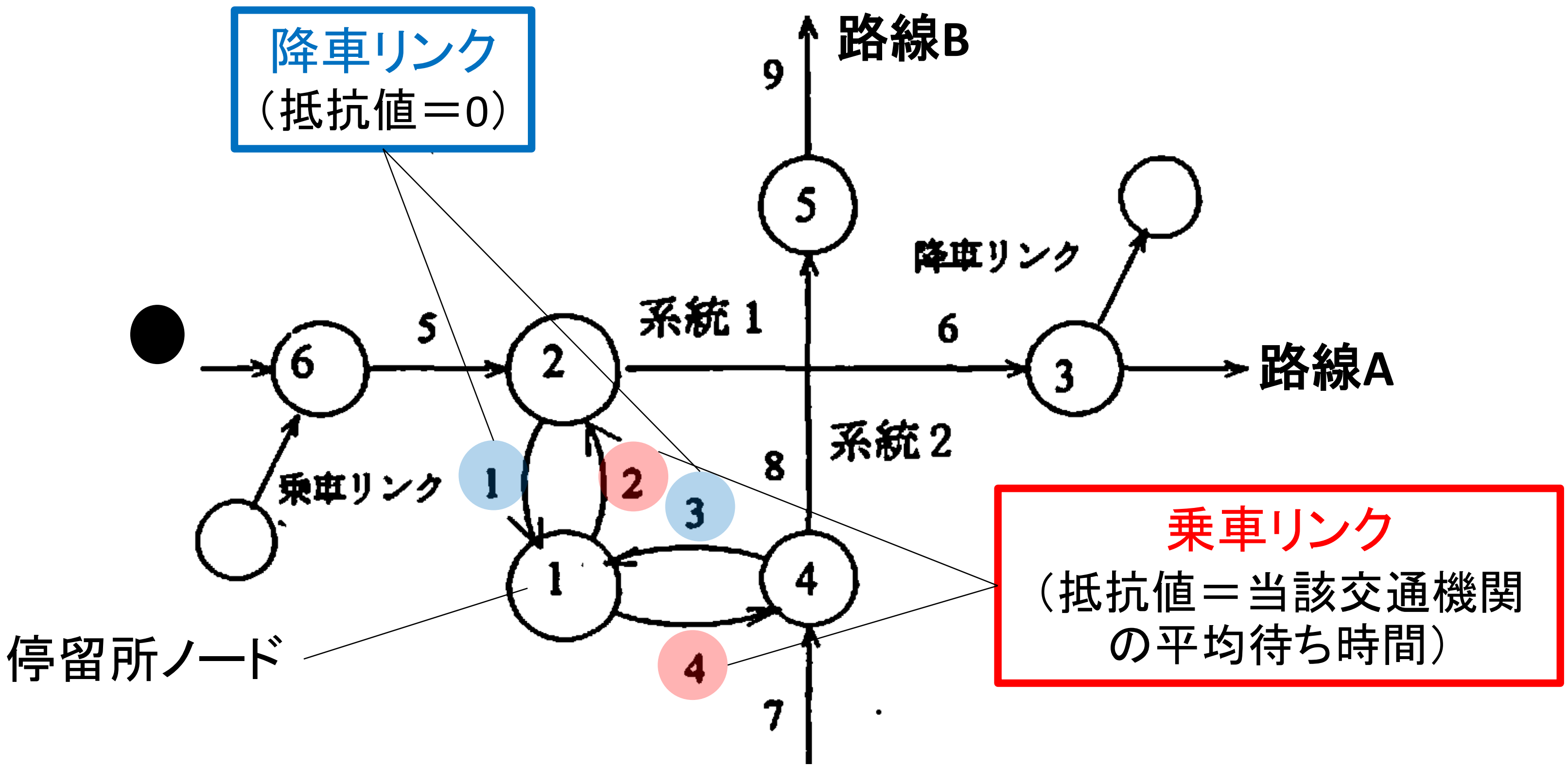


図3　公共交通輸送網のネットワーク

分析の対象となる実際の交通網は，分析の目的・求められる精度・予算などに応じて適切に表現されなければならない

1節　交通ネットワークの構成要素と交通網のネットワーク表現

詳細(ミクロ)な都市交通需要分析

⇒街路網ネットワーク(city street network)が最適

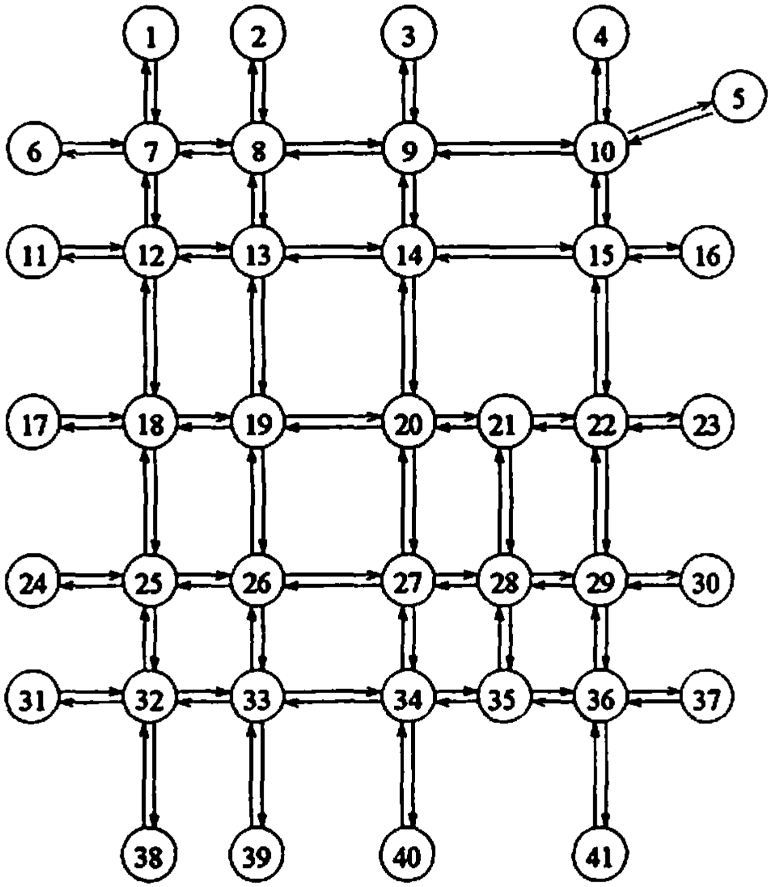


図4　都市街路網ネットワーク

- リンク→すべての道路区間
- ノード→すべての交差点・交通施設

1節　交通ネットワークの構成要素と交通網のネットワーク表現

大都市圏（マクロ）での交通需要分析

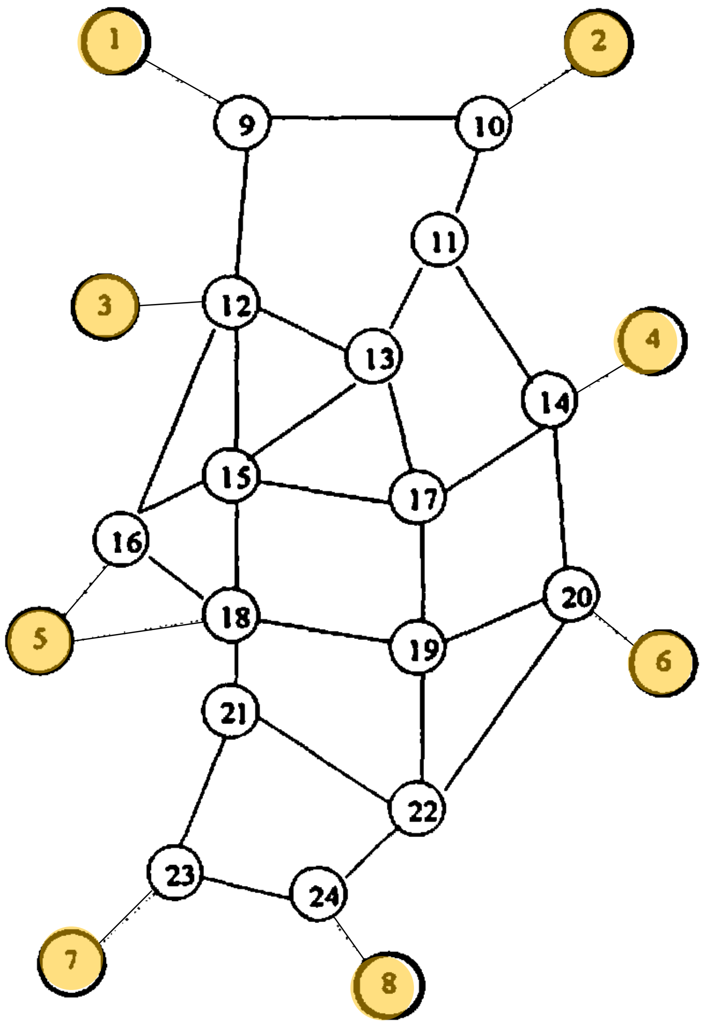


図5　主要道路網ネットワーク

分析対象地域をゾーン分割
（ゾーン中心にセントロイドを想定）

↓

主要道路で構成されるネットワークを設定

↓

ゾーン間で交通需要分析を行う

1節　交通ネットワークの構成要素と交通網のネットワーク表現

時間・空間を同時に表現するネットワーク
⇒タイムテーブルにしたがって運行される列車ダイヤと輸送網を表現する

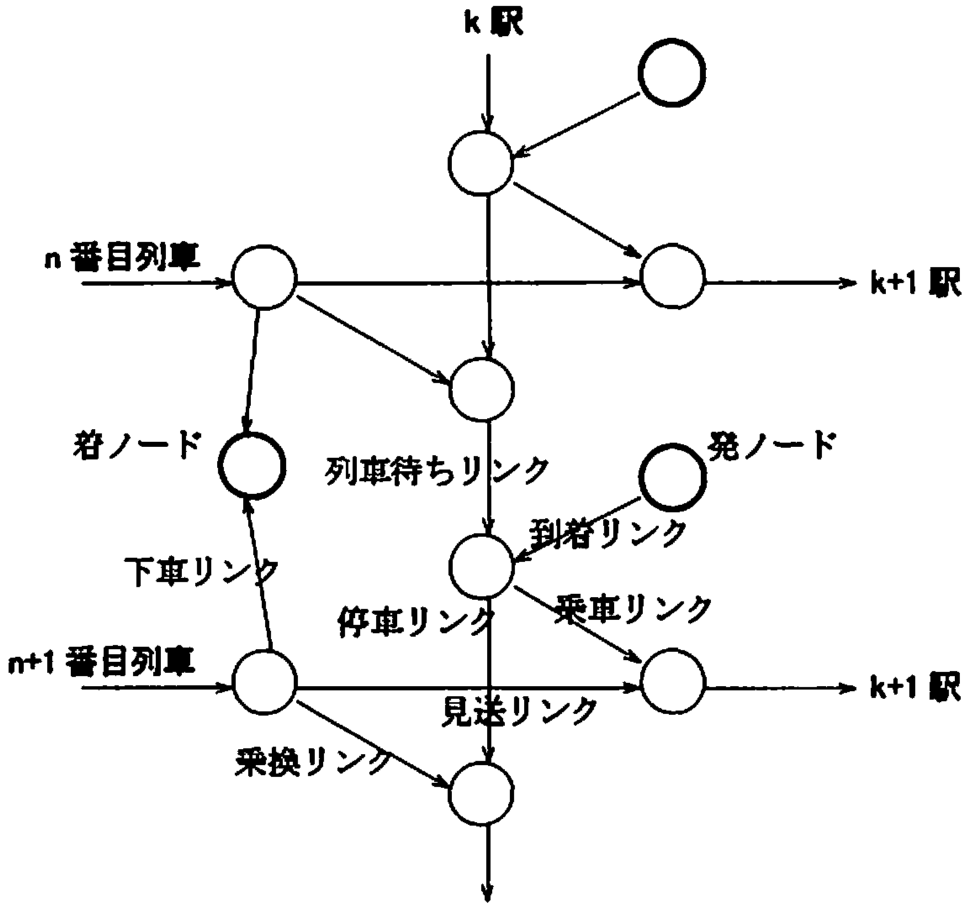


図6　列車ダイヤの時間・空間的ネットワーク表現

横軸・・・時間（時刻）
縦軸・・・空間（駅・駅間）

1節　交通ネットワークの構成要素と交通網のネットワーク表現

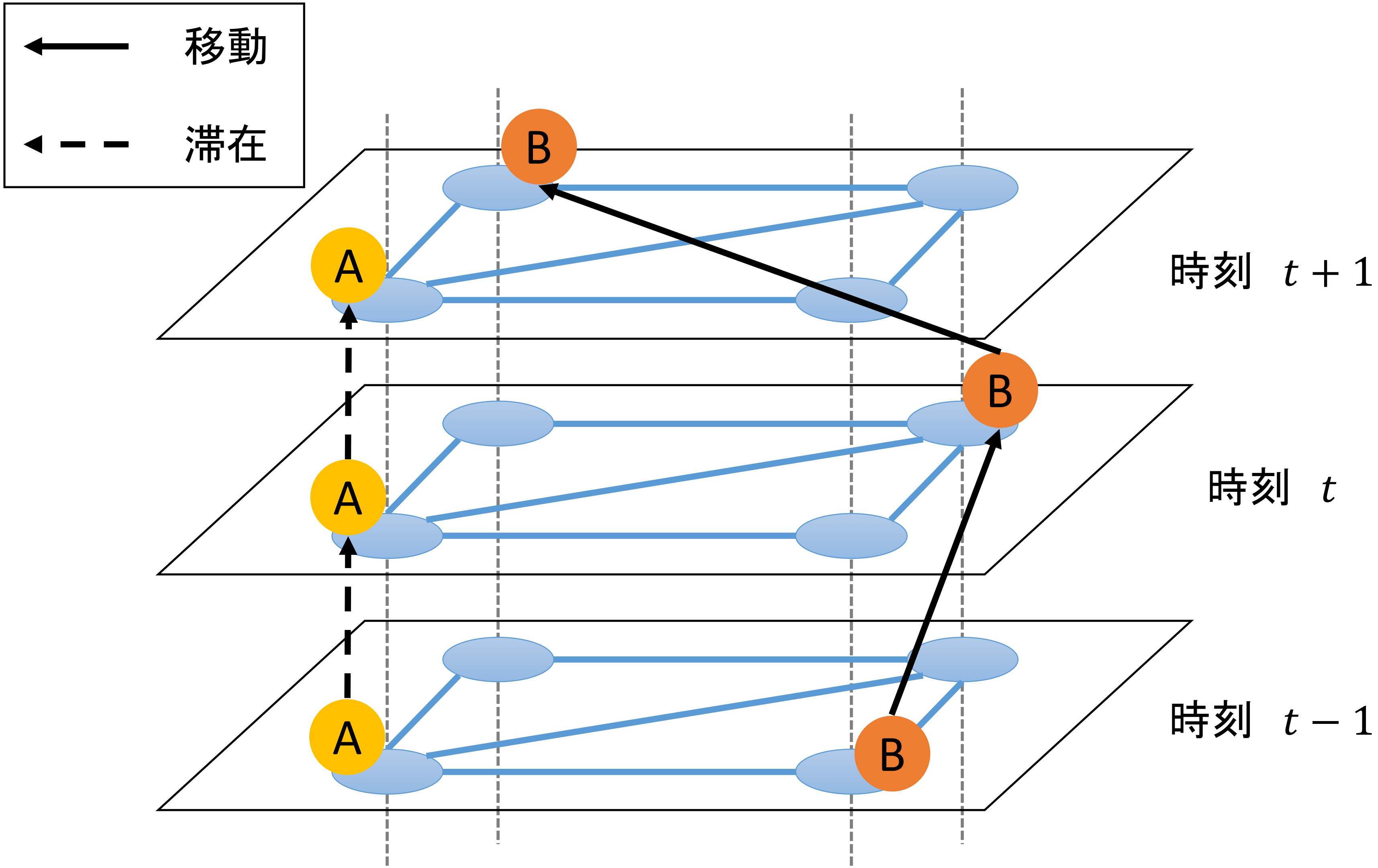


図7　時間・空間的ネットワーク表現（3次元）

2節　リンクパフォーマンス関数

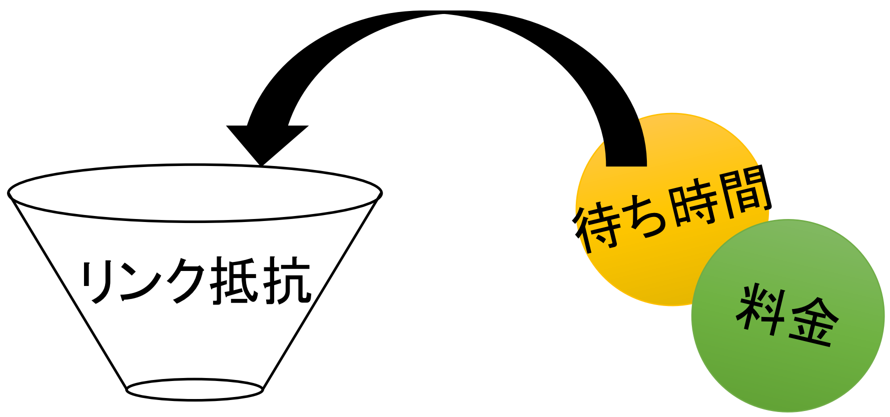
交通ネットワーク上の各リンクは特有の抵抗を持つ

抵抗を構成する要因
- 旅行時間
- 走行費用
- 安全性
- 快適性…etc

人の経路選択において
最も主要な要因は“**<u>目的地までの所要時間</u>**”

ゆえに所要時間がリンク抵抗の単一の測度として用いられる場合が多い

しかし，公共交通輸送ネットワークへのパーソントリップの配分問題では…



2節　リンクパフォーマンス関数

$t_a = t_a(x_a)$ ・・・リンクパフォーマンス関数（t_aを表す関数）

リンク抵抗t_aは定数ではなく，当該リンクの交通量x_aの関数と考えられる
⇒ x_aが増加すると，所要時間は増加し安全性は低下する

リンクパフォーマンス関数≒リンクコスト関数，リンク抵抗関数，リンク旅行時間関数

交通流理論におけるリンクパフォーマンス関数

→ リンクパフォーマンス関数①
　交通混雑による交通流の停止はあり・信号停止なし

→ リンクパフォーマンス関数②
　交通混雑による交通流の停止はあり・信号停止あり

2節　リンクパフォーマンス関数

リンクパフォーマンス関数①
交通混雑による交通流の停止はあり・信号停止なし

リンクa上での速度V_a(km/h)と交通量x_a(pcu/h)，密度D_a(pcu/km)の関係

$$x_a = D_a * V_a$$

＊passenger car unit（車両一台を自動車交通量に換算した自動車換算交通量）

速度V_aと密度D_aは近似的に線形関係にある

$$V_a = V_{af} - \left(\frac{V_{af}}{D_{aj}}\right) * D_a$$

V_{af}：ゼロフロー時の速度（自由速度）
D_{aj}：速度がゼロのときの密度（飽和密度）

上記2式より，D_aを削除すると…

$$V_a^2 - V_{af} * V_a + \left(\frac{V_{af}}{D_{aj}}\right) * x_a = 0 \cdots 式X$$

2節　リンクパフォーマンス関数

式Xは速度に関する2次関数
この式からリンクaを通過可能な最大交通量C_a（＝可能交通量）を求める

$$\frac{dX}{dV_a} = 2V_a - V_{af} = 0$$

$$V_a = \frac{1}{2}V_{af}$$

$V_a = \frac{1}{2}V_{af}$を式Xに代入すると

$$C_a = D_{aj} * \left(\frac{V_{af}}{4}\right)$$

リンクaの単位距離当たりの所要時間t_aは速度V_aの逆数であるから，
式Xは以下のようになる

$$\left(\frac{1}{t_a}\right)^2 - \left(\frac{1}{t_{a0}}\right) * \left(\frac{1}{t_a}\right) + \left\{\frac{\left(\frac{1}{t_{a0}}\right)}{D_{aj}}\right\} x_a = 0$$

t_{a0}：ゼロフロー時の所要時間

2節　リンクパフォーマンス関数

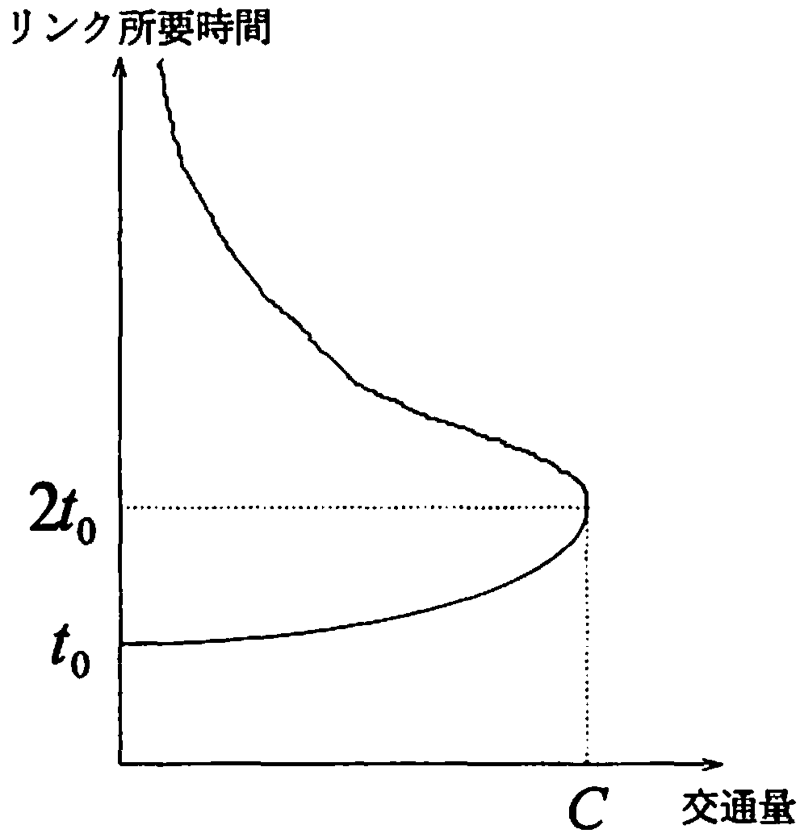


図　停止のない場合の交通量と旅行時間

- ひとつの交通量の値に対して2つの所要時間の値を持つ二値関数
- 交通量Cが最大のとき，$V_a = \frac{1}{2}V_{af}$ ゆえに$t = 2t_{a0}$

2節　リンクパフォーマンス関数

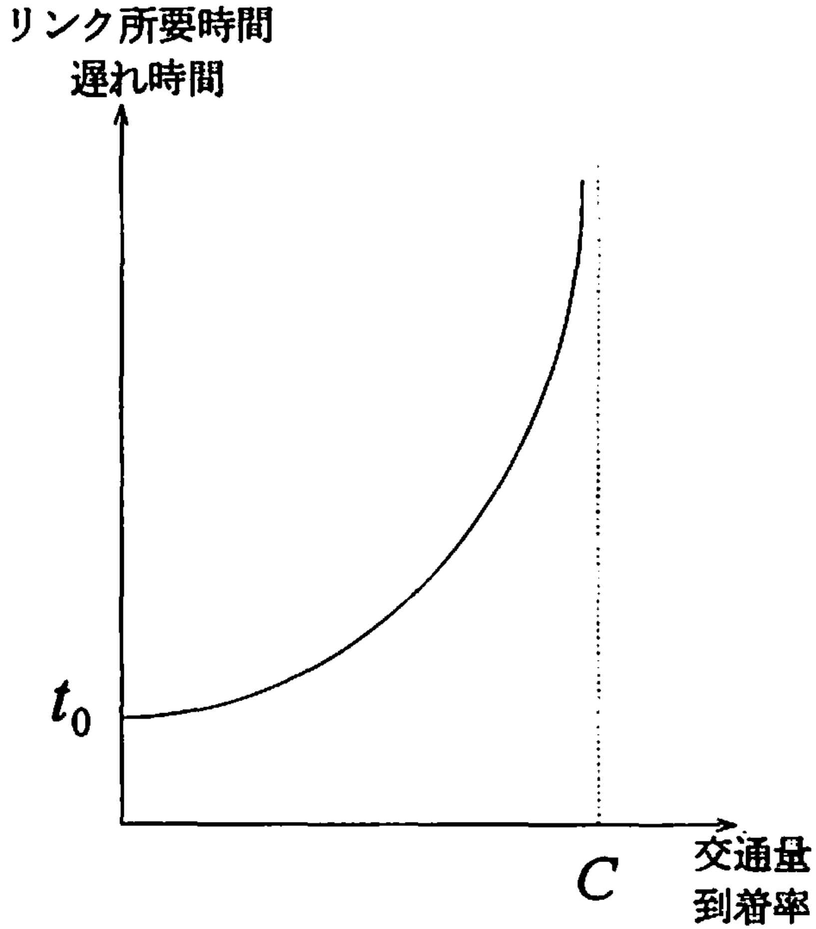


図　停止のある場合の交通量と旅行時間

一方，当該リンクの下端で信号停止を余儀なくされる場合

所要時間＝走行時間＋信号遅れ時間

交通量が信号交差点容量に近づくにつれ信号遅れ時間が急速に増加

交通流がストップ

ゆえに交通容量に漸近するような関数

2節　リンクパフォーマンス関数

Davidson関数･･･待ち行列理論により導出された関数

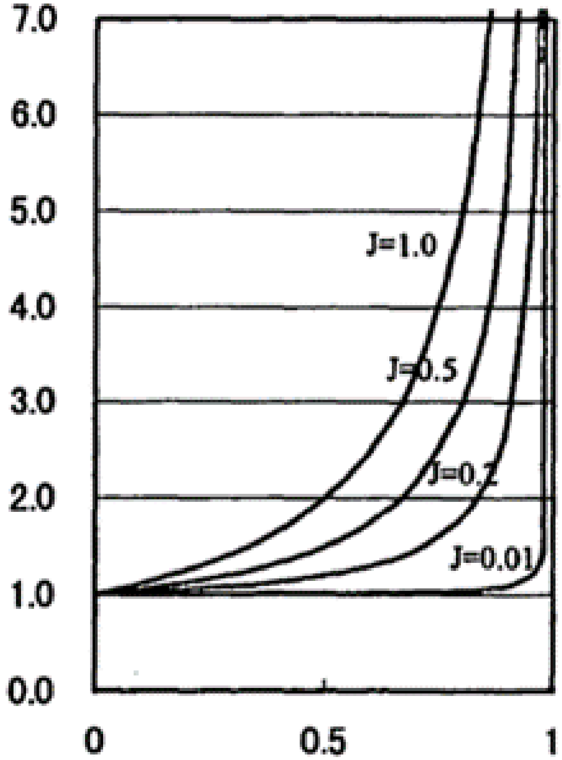


図　Davidson関数

$$t_a(x_a) = t_{a0}\left\{1 + \frac{Jx_a}{(C_a - x_a)}\right\}$$

x_a : リンク交通量

t_{a0} : 交通量ゼロのときの所要時間

C_a : 交通容量

J : 遅れパラメータ（リンク特性に依存）

未知数

2節　リンクパフォーマンス関数

BPR関数・・・米国道路局が考案したモデル

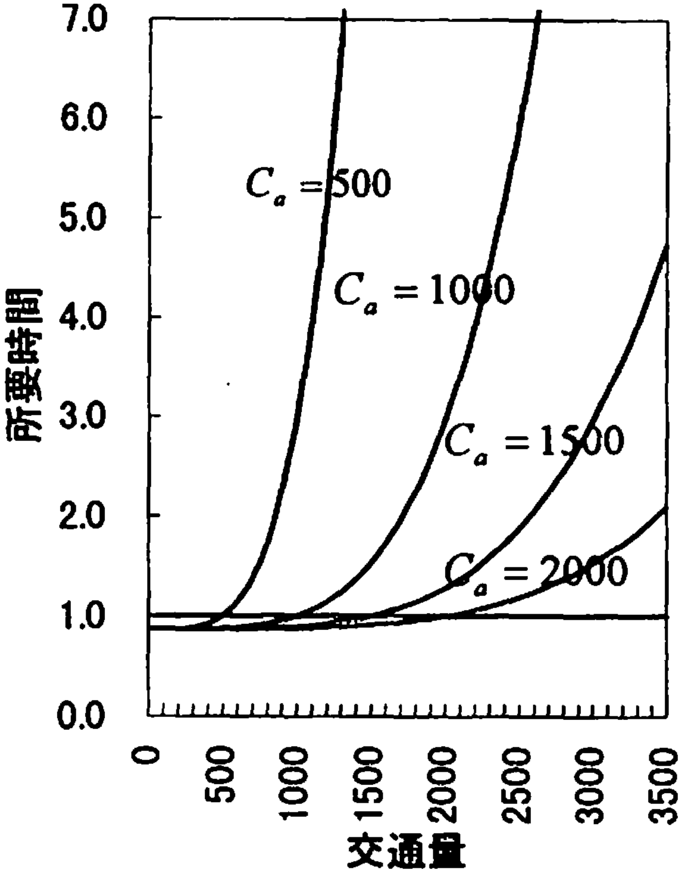


図　BPR型リンクパフォーマンス関数

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

α, β:パラメータ
米国（$\alpha = 0.15, \beta = 4$）

↓（日本の道路規格に適用）

日本（$\alpha = 2.62, \beta = 5$ ）

第3章　ネットワークフローが満足すべき条件

1節　パス—リンクインシデンスマトリックス

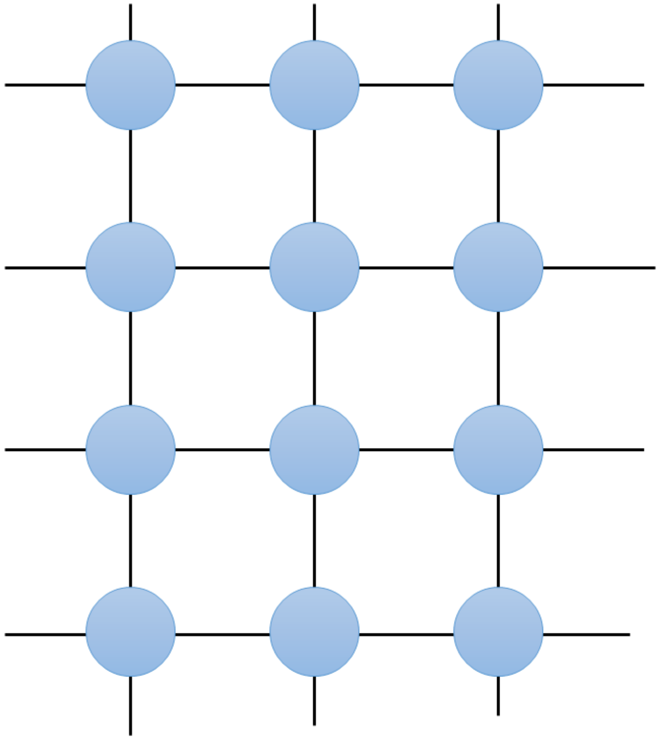
- 交通ネットワークを数学的に記述する術

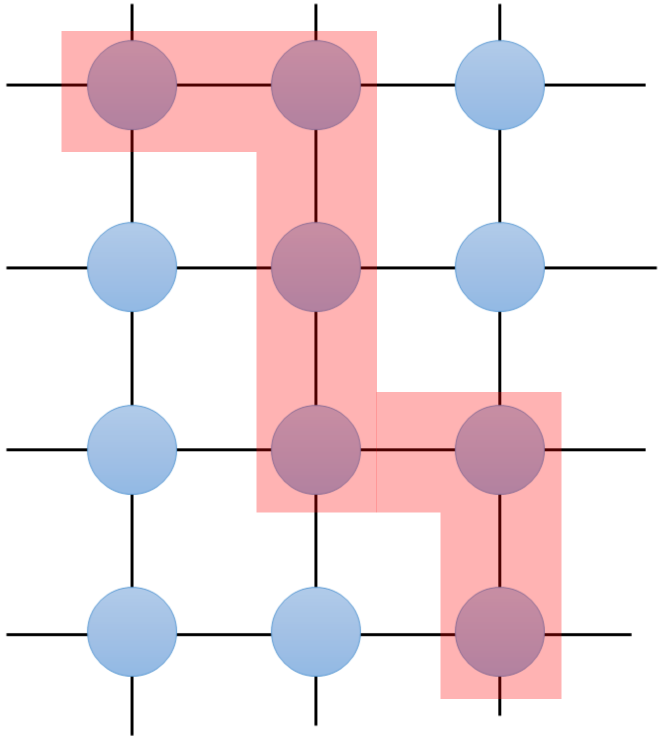
2節　ネットワークフローの保存条件

- 均衡フローを求める際にネットワークが満たすべき条件

1節　パス－リンクインシデンスマトリックス

- 任意のODペアrs間には利用可能な経路は沢山存在する
  しかし，全ての経路が利用されているわけではない

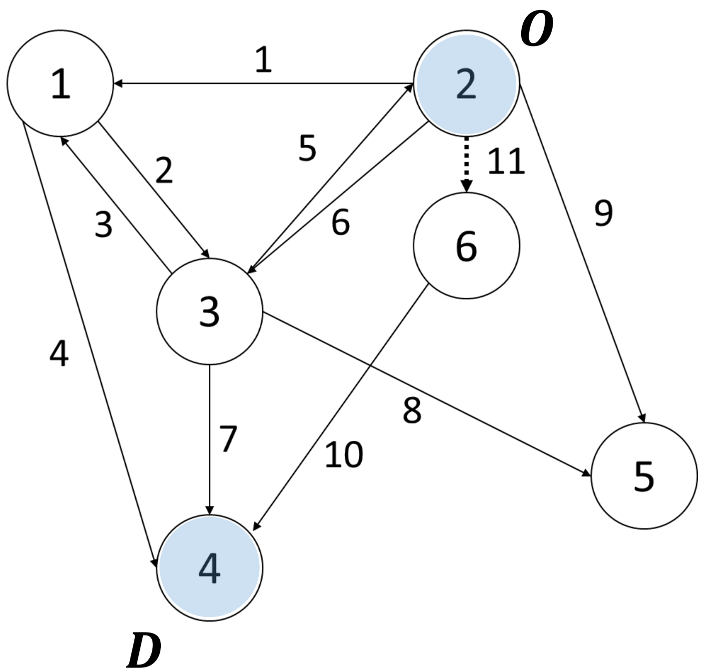




1節　パスーリンクインシデンスマトリックス

- 有効経路集合K_{rs}を数学的に記述する

ネットワーク図



パスーリンクインシデンスマトリックスΔ^{24}

$a\backslash k$　1　2　3　4　5　← 経路番号

$$
\Delta^{24} = \left\{ \delta^{24}_{a,k} \right\} =
\begin{array}{c}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11
\end{array}
\begin{pmatrix}
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

↑

リンク番号

$$
\delta^{24}_{a,k} = \begin{cases} 1（経路k中にリンクaが存在する） \\ 0（経路k中にリンクaが存在しない） \end{cases}
$$

2節　ネットワークフローの保存条件

ネットワーク上のフローを表す基本的な変数

- OD交通量（OD フロー）　$Q_{rs}(rs \in \Omega)$
- 経路交通量（経路フロー）　$f_k^{rs}(k \in K_{rs}, rs \in \Omega)$
- リンク交通量（リンクフロー）　$x_a(a \in A)$

◆ フローの保存条件

1. $x_a = \sum_{rs} \sum_k \delta_{a,k}^{rs} f_k^{rs} \quad a \in A$ ➡ $rsOD$間のパス—リンクインシデンスマトリックスの(a, k)要素
2. $\sum_k f_k^{rs} = Q_{rs} \quad rs \in \Omega$ ➡ OD交通量の保存式
3. $f_k^{rs} \geq 0 \quad k \in K_{rs}, rs \in \Omega$ ➡ 経路交通量についての非負条件

2節　ネットワークフローの保存条件

1. $x_a = \sum_{rs} \sum_k \delta_{a,k}^{rs} f_k^{rs} \qquad a \in A$

2. $\sum_k f_k^{rs} = Q_{rs} \qquad rs \in \Omega$

3. $f_k^{rs} \geq 0 \qquad k \in K_{rs}, rs \in \Omega$

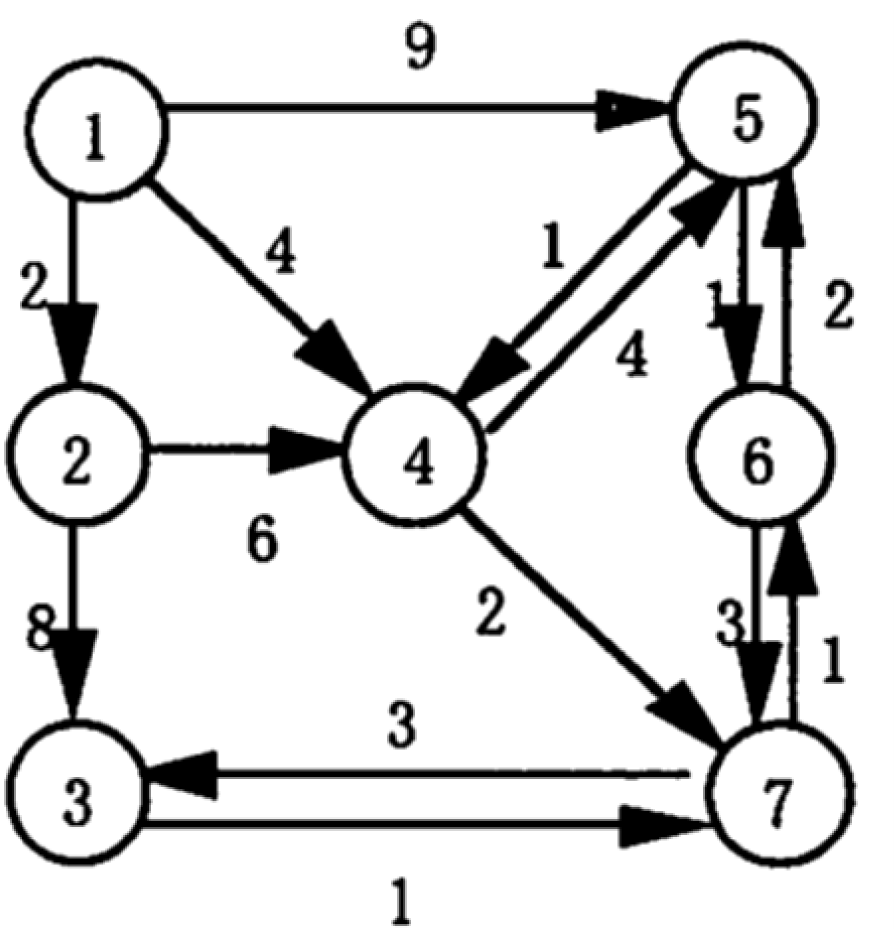
フローの保存条件は経路交通量f_k^{rs}を変数としている
⇒f_k^{rs}が交通需要者の経路選択を直接的に表現することができるため

しかし実際の経路交通量は道路区間上では観測できない

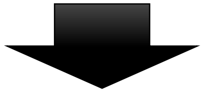
モデルによる推定結果の適合性

- １式により交通量x_aを算出し，観測値と比較する

R を活用したネットワークの作成



左にあるネットワークを
R を活用して表現する



パッケージ“igraph”を使用

ネットワークの作成について

■ コード

```
#ネットワークの表現（可視化）

#初回だけ，以下の行を実行してパッケージをインストールしてください
install.packages("igraph")

#パッケージ
library(igraph)

p<-as.matrix(read.csv("matrix.csv",header=F)) #隣接行列（matrix.csvの中身）の読み込み
g<-graph.adjacency(p,weighted=TRUE) #グラフオブジェクトの作成
V(g)$name<-c("1","2","3","4","5","6","7") #ノード番号
lay<-rbind(c(1,3),c(1,2),c(1,1),c(2,2),c(3,3),c(3,2),c(3,1)) #グラフレイアウト(座標の指定)

#データ，レイアウト，ノードのラベル，点の大きさ，リンクのラベル
plot(g, layout=lay,vertex.label=V(g)$name,vertex.size=20,edge.label=E(g)$weight)
```

$$
\begin{array}{cc}
 & \text{目的地ノード} \\
\text{出発地ノード} & \begin{pmatrix} 0 & 2 & 0 & 4 & 9 & 0 & 0 \\ 0 & 0 & 8 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 3 \\ 0 & 0 & 3 & 0 & 0 & 1 & 0 \end{pmatrix}
\end{array}
$$

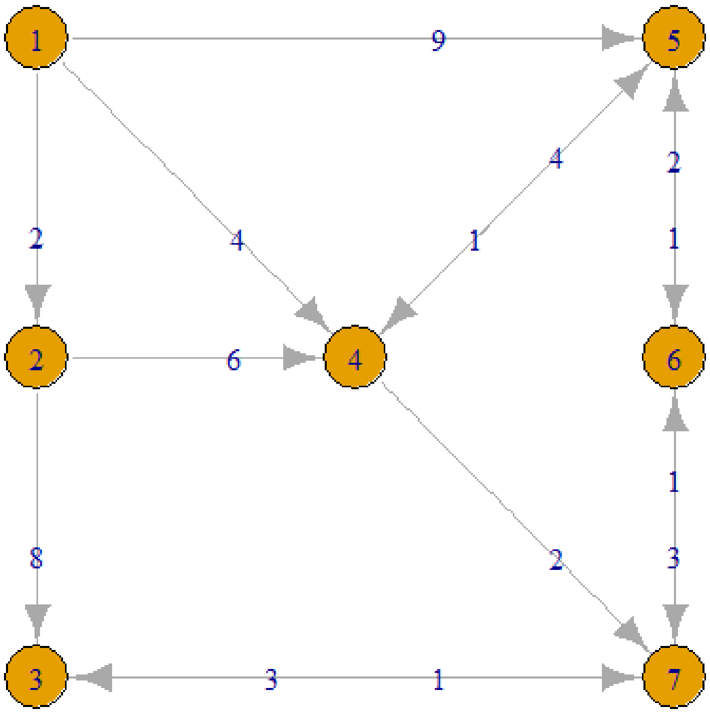
ノード2からノード3へのリンクコスト

ネットワークの作成について

■ 結果

```
#ネットワークの表現（可視化）

#初回だけ，以下の行を実行してパッケージをインストールしてください
install.packages("igraph")

#パッケージ
library(igraph)

p<-as.matrix(read.csv("matrix.csv",header=F)) #隣接行列（matrix.csvの中身）の読み込み
g<-graph.adjacency(p,weighted=TRUE) #グラフオブジェクトの作成
V(g)$name<-c("1","2","3","4","5","6","7") #ノード番号
lay<-rbind(c(1,3),c(1,2),c(1,1),c(2,2),c(3,3),c(3,2),c(3,1)) #グラフレイアウト(座標の指定)

#データ，レイアウト，ノードのラベル，点の大きさ，リンクのラベル
plot(g, layout=lay,vertex.label=V(g)$name,vertex.size=20,edge.label=E(g)$weight)
```

今回作成したネットワーク



pythonにおけるネットワークの表現

- pythonにはとても便利なNetworkXというパッケージが存在する。
- これを使うと様々なネットワークの表現が可能となる。
- まず、importする。(pyplotも)

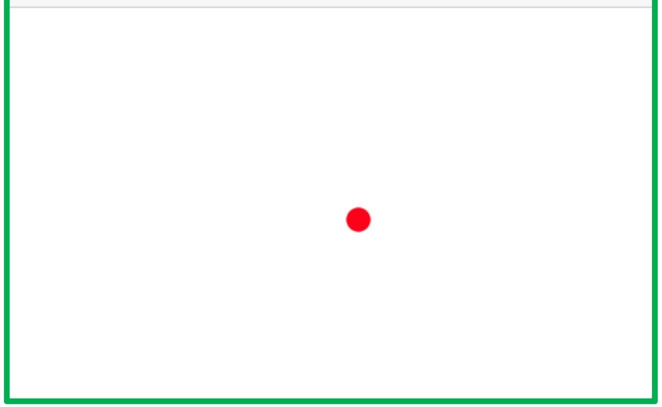
```
import networkx as nx
import matplotlib.pyplot as plt
```

- とりあえず空のグラフを作る。

```
G = nx.Graph()
```

- ノードを追加する、そして描画する

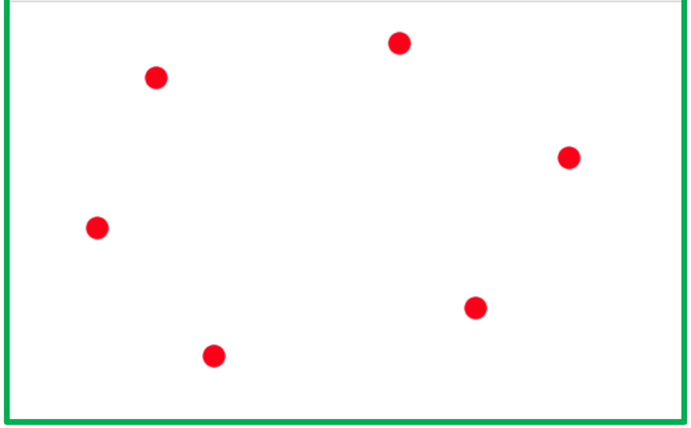
```
G.add_node(0) # add_nodeでノードを追加
nx.draw(G) # nx.draw()でグラフを描画
```



pythonにおけるネットワークの表現

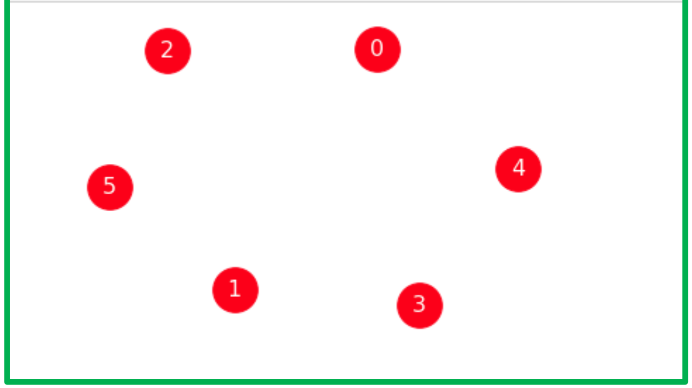
- 一度にたくさん追加もできる。

```
G.add_nodes_from([1,2,3,4,5])
nx.draw(G)
```



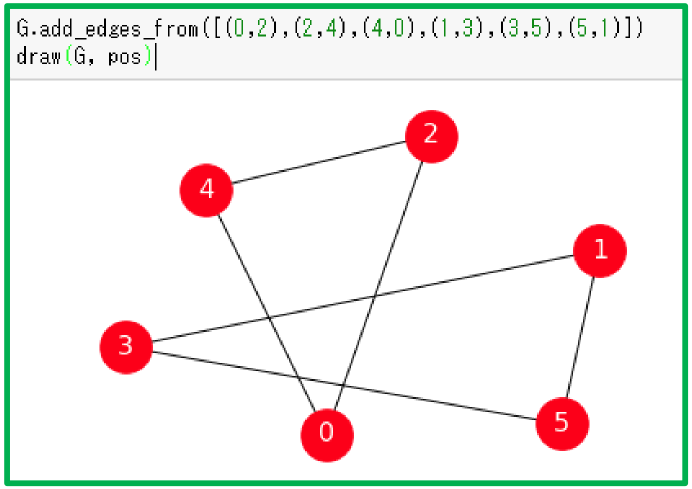
- ノードの名前を振ったりもできる
  - posは位置関係などをいじる項
  - plt.axis(‘off’)をしないと軸が表示される
  - 色は頭文字で表せられるやつもある
  - ここからは魔法を使ってこれらをdrawで表す

```
pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, node_size=1000, node_color="r")
nx.draw_networkx_edges(G, pos, width=1)
nx.draw_networkx_labels(G, pos, font_size=16, font_color="w")
plt.axis('off')
plt.show()
```



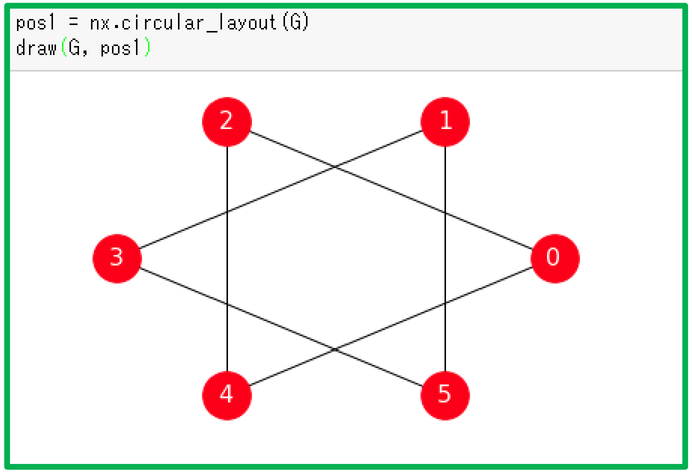
pythonにおけるネットワークの表現

- ノードの追加は.add_edge
- 同様に一度に追加もできる



- 思った通りの配置にならないためposをいじる

- posを円形のように配置するように設定する



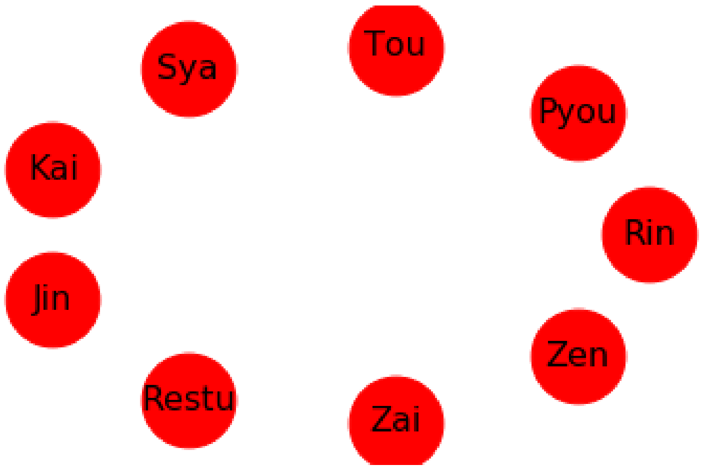
- いい感じになる

pythonにおけるネットワークの表現

- ノードの名前は文字列にもできる
- こんなこともできる

```
def KUJI():
    import networkx as nx
    import matplotlib.pyplot as plt

    kuji = nx.Graph()
    kuji.add_nodes_from(['Rin','Pyou','Tou','Sya','Kai','Jin','Restu','Zai','Zen'])
    kujipos = nx.circular_layout(kuji)
    nx.draw_networkx_nodes(kuji, kujipos, node_size=2000, node_color="r")
    nx.draw_networkx_labels(kuji, kujipos, font_size=16, font_color="black")
    plt.axis('off')
    plt.show()

KUJI()
```

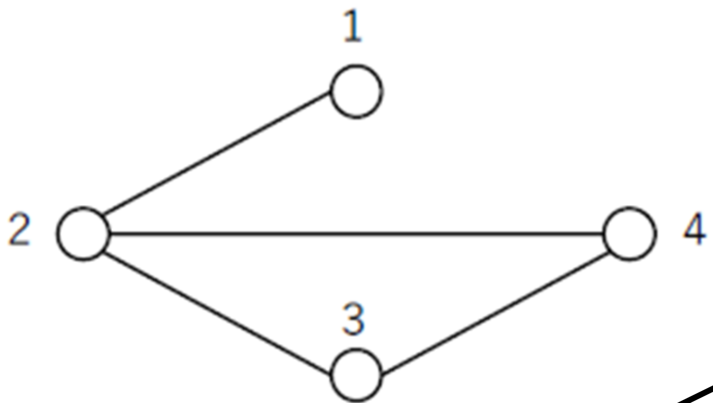
- 切羽詰まっているときにこれに20分かけた



- 他にも隣接行列やcsvファイルからそのまま読み取る方法があるが、それは追って紹介する

発展的内容　ラプラシアン行列とリンクの脆弱性

◆ 通常のネットワーク



ノード2に繋がるリンクの本数

ノード1とノード2を結ぶリンクが存在する

$$L = D - A$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

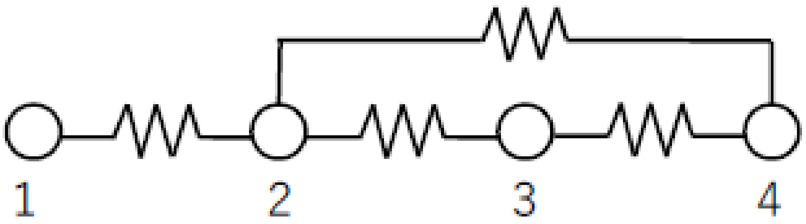
L : ラプラシアン行列

D : 次数行列

A : 隣接行列

発展的内容　ラプラシアン行列とリンクの脆弱性

◆ ネットワークに対応する一次元バネ・質点系



• バネ・質点系の運動方程式

$$m\begin{Bmatrix}\ddot{u}_1\\ \ddot{u}_2\\ \ddot{u}_3\\ \ddot{u}_4\end{Bmatrix}+k\begin{bmatrix}1 & -1 & 0 & 0\\ -1 & 3 & -1 & -1\\ 0 & -1 & 2 & -1\\ 0 & -1 & -1 & 2\end{bmatrix}\begin{Bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{Bmatrix}=\begin{Bmatrix}0\\ 0\\ 0\\ 0\end{Bmatrix}$$

k : バネ定数

m : 質量

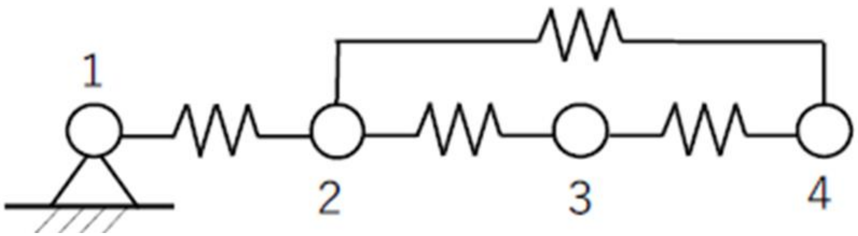
u_i : 質点の変位

ω : 角速度

$$\because \ddot{u}_i = -\omega^2 u_i$$

$$\begin{bmatrix}1 & -1 & 0 & 0\\ -1 & 3 & -1 & -1\\ 0 & -1 & 2 & -1\\ 0 & -1 & -1 & 2\end{bmatrix}\begin{Bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{Bmatrix}=\frac{m\omega^2}{k}\begin{Bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{Bmatrix}$$

◆ ノード固定条件付き一次元バネ・質点系



＊ノード1が固定されているため$u_1 = 0$

• バネ・質点系の運動方程式

$$m\begin{Bmatrix}\ddot{u}_1\\ \ddot{u}_2\\ \ddot{u}_3\\ \ddot{u}_4\end{Bmatrix} + k\begin{bmatrix}1+M & -1 & 0 & 0\\ -1 & 3 & -1 & -1\\ 0 & -1 & 2 & -1\\ 0 & -1 & -1 & 2\end{bmatrix}\begin{Bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{Bmatrix} = \begin{Bmatrix}0\\ 0\\ 0\\ 0\end{Bmatrix}$$

M:ペナルティー定数
↓
十分に大きな正の数

発展的内容　ラプラシアン行列とリンクの脆弱性

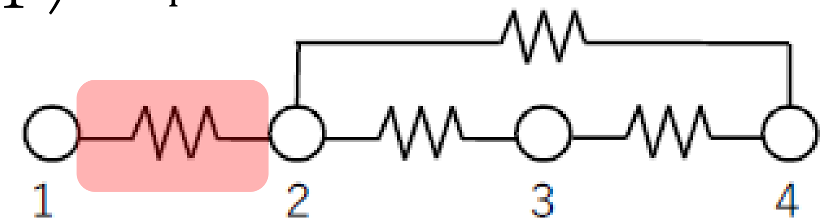
固有値・固有ベクトルの算出

◆ ネットワークに対応する一次元バネ・質点系

$$\lambda^3 + \lambda^2 - 2\lambda = 0 \qquad \therefore \lambda_1 = -2, \quad \lambda_2 = 0, \quad \lambda_3 = 1$$

＊ラプラシアン行列の固有値は必ず正となる → 第2最小固有値は1となる

$$\begin{pmatrix} 0 & -1 & 0 & 0 \\ -1 & 2 & -1 & -1 \\ 0 & -1 & 1 & -1 \\ 0 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = 0 \qquad \therefore (0.82\,, 0.2E^{-15}\,, -0.41\,, -0.41)$$



固有ベクトルで最大の成分は“0.82”
⇒バネ張力の最大値はリンク（1，２）に生じる

リンク（1，２）が脆弱性が高い

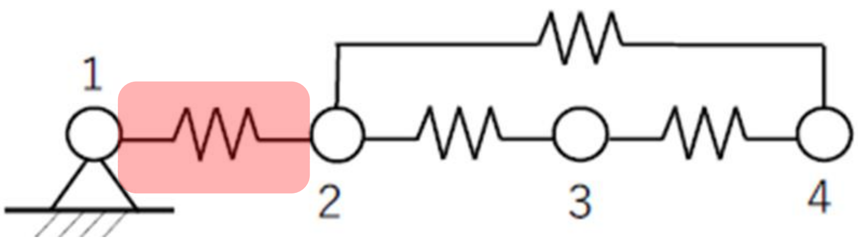
発展的内容　ラプラシアン行列とリンクの脆弱性

◆ ノード固定条件付き一次元バネ・質点系

＊中央集権的に物資や情報が伝達される状況

固有値　$\lambda = 0.27$

固有ベクトル　$(0\,, -0.46\,, -0.63\,, -0.63)$



固有ベクトルで最大の成分は“0”
⇒バネ張力の最大値はリンク（1，2）に生じる

リンク（1，2）が脆弱性が高い

参考文献

交通ネットワークの均衡分析　–最新の理論と解法–

積荷の均衡配分による海上貿易の港の混雑分析
http://www.ise.chuo-u.ac.jp/ise-labs/taguchi-lab/pdf/r01d8101025.pdf

小林俊一，中山昌一郎，松井千里，若林桂汰：道路ネットワークのラプラシアン行列による脆弱性解析，第55回土木計画学研究発表会・講演集