Dijkstra's algorithm & Dial's algorithm

- 土木学会(1998)
  交通ネットワークの均衡分析　第8章 pp.133-140、第9章 pp. 167-171
- Sheffi, Y. (1985)
  Urban transportation networks, Part 4, Chapter 11, pp.288-297

2016年5月23日
スタートアップゼミ#6
B4　森田智美

本日の話題

- Dijkstra's algorithm
  - ダイクストラ法(ラベル確定法)
  - ヒープ構造
  - ラベル修正法
- Dial's algorithm
  - アルゴリズム
  - ロジット型確率配分モデルとの等価性
  - ロジット型の限界

Dijkstra's algorithm

Dijkstra法の導入

- 均衡配分を解く過程には、最小経路(shortest path)探索が含まれる

- そのうちのひとつの解としてDijkstra's algorithm (ダイクストラ法orラベル確定法)を扱う

- ノードjまでの最短経路は
  $d(j) = \min \{ t_{ij} + d(i) \}$ ただしiはjに向かうリンクが発するノード
  という再帰的な式が書ける

※ベルマンの最適性原理(動的計画法の基本)

Dijkstra法の特徴

- 起点ノードに近いノードから順に、全方向に向かって、最短経路を列挙していく
- ひとつのOriginに対して、他の全てのノードへの最短経路と最小費用が一度に求まる
- 最短経路とならないと確認した経路については計算しない

  →効率のよいアルゴリズム

定式化の前に

- 全てのノードは集合Kまたは$\overline{K}$ に含まれる
  K：Oからの最短経路が確定したノードの集合
  $\overline{K}$：Oからの最短経路が未確定のノードの集合

- 部分的最小費用 c_j　　←これが「ラベル」
  そのノードまでの今のところの最小費用(旅行時間)

- リンクコスト t_{ij} (≧0)
  ※負も含む場合はベルマンフォード法

- 先行ポインタF_j
  ノードjに到達する最短経路(暫定)上で、jの直前となるノード

algorithm

Step 1

- 全てのノードjについて　$c_j = \infty$,　$F_j = 0$,　$j \in \overline{K}$
- $c_o \equiv 0$, $i \equiv o$とし、ノードoを集合Kに移す

Step 2

- ノードiから出る全てのリンクの終点ノード{m}について
  $c_m > c_i + t_{im}$ ならば $c_m = c_i + t_{im}$ と更新、$F_m = i$とする

Step 3

- 集合 $\overline{K}$　に属する全てのノードについて、

$$c_j \equiv \min_{p \in \overline{K}} \{c_p\}$$

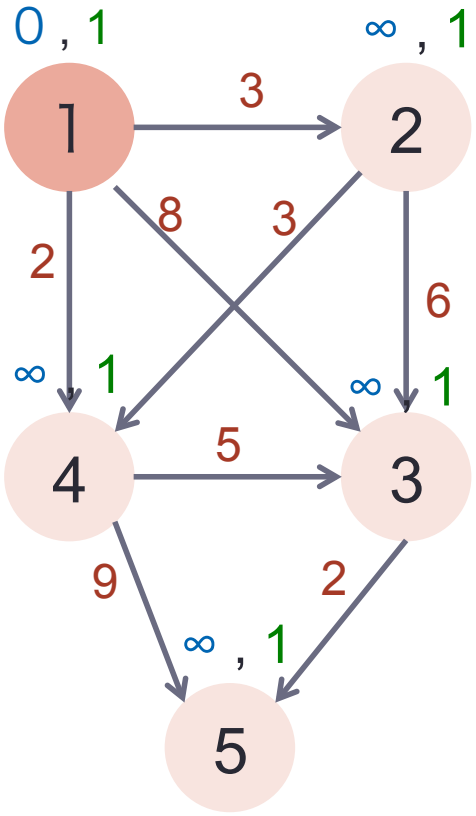
で決定されたノードjを集合Kに移す

Step 4

- $c_p = \infty$ 以外の全てのノードが集合Kに移されれば終了
- そうでなければ $i = j$として、Step 2から繰り返し

簡単な例

ノード1をO、ノード5をDとする最短経路を考える

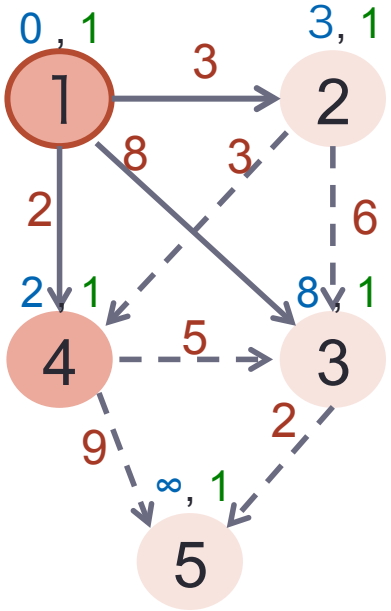


赤… リンクコスト
青… 部分最小費用
緑… 先行ポインタ

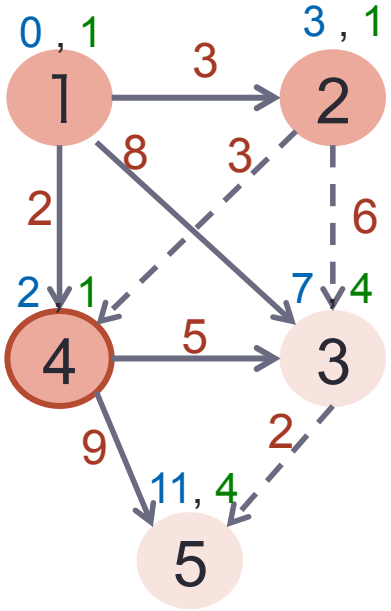
色が濃いノード ∈ K
色が薄いノード ∈ $\overline{K}$

簡単な例

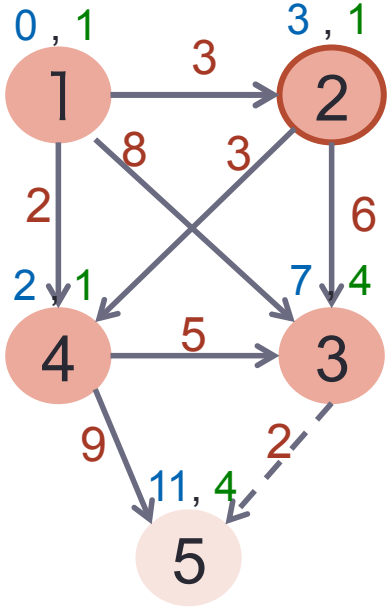
t = 1



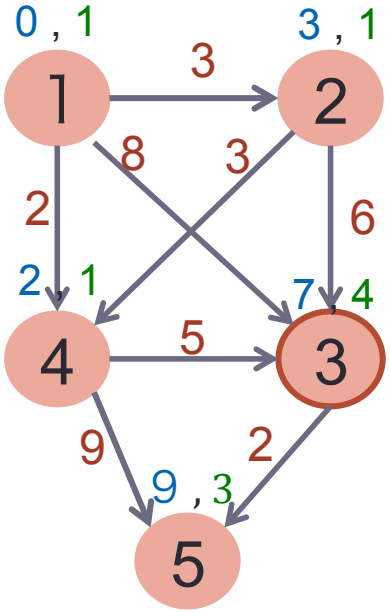
t = 2



t = 3



t = 4



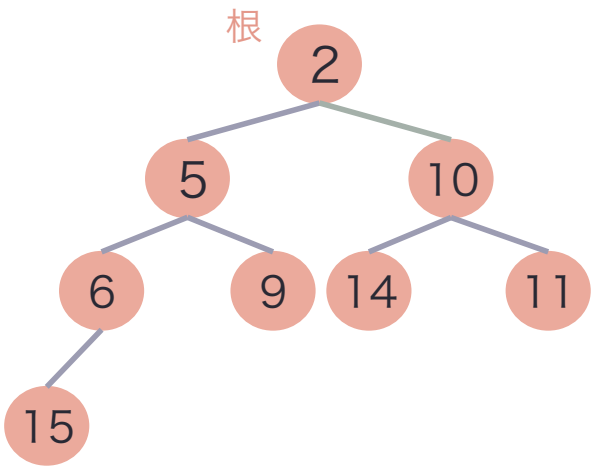
ノード5 → (F_5 =)ノード3 → (F_3 =)ノード4 → (F_4 =)ノード1
と遡ることで、最短経路がわかる

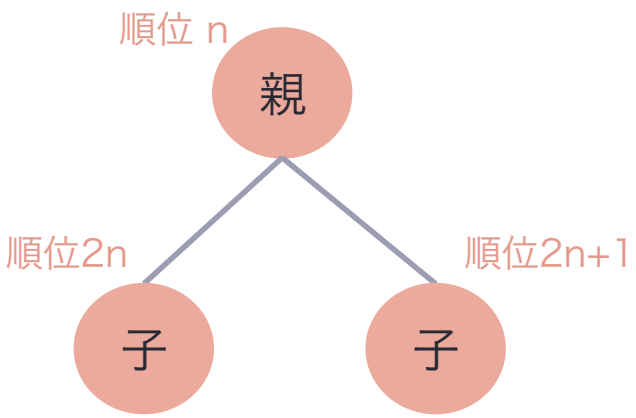
ヒープ(heap: 根付き木)構造

- Step 3で最小ラベルを求めるために毎回ラベルをソートするのは効率が悪い
- 全データをソートすることなく最小値を見つけられる

- 2分木の例

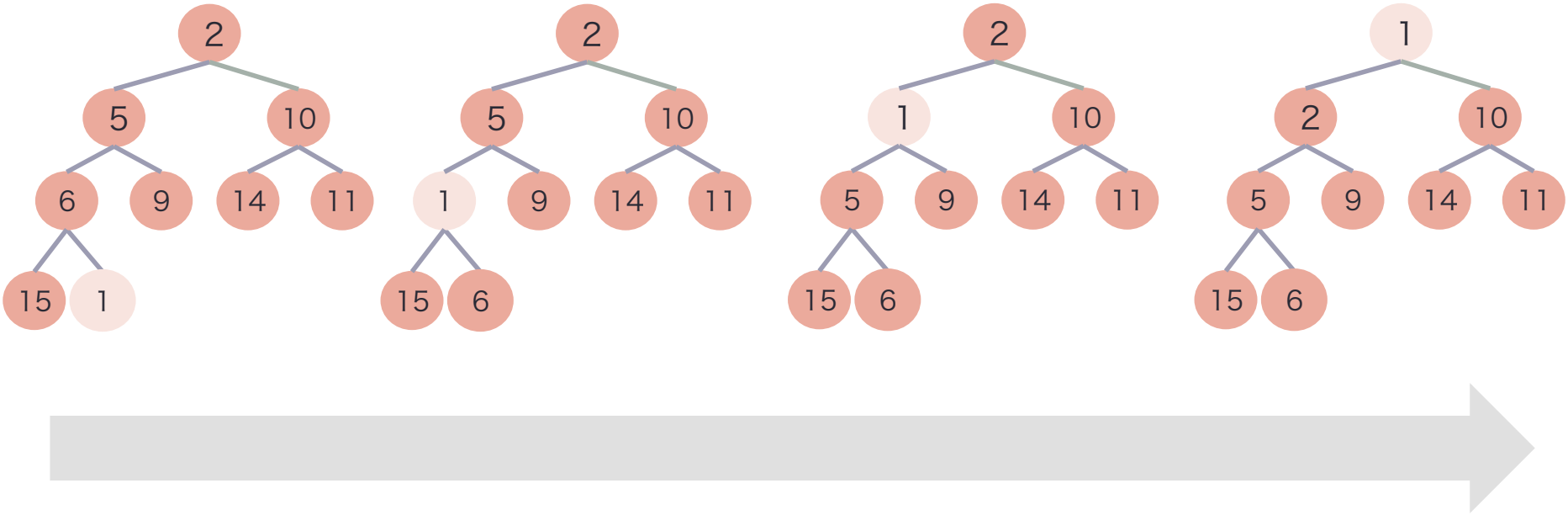
配列順位	1	2	3	4	5	6	7	8	9
ラベル	2	5	10	6	9	14	11	20	15





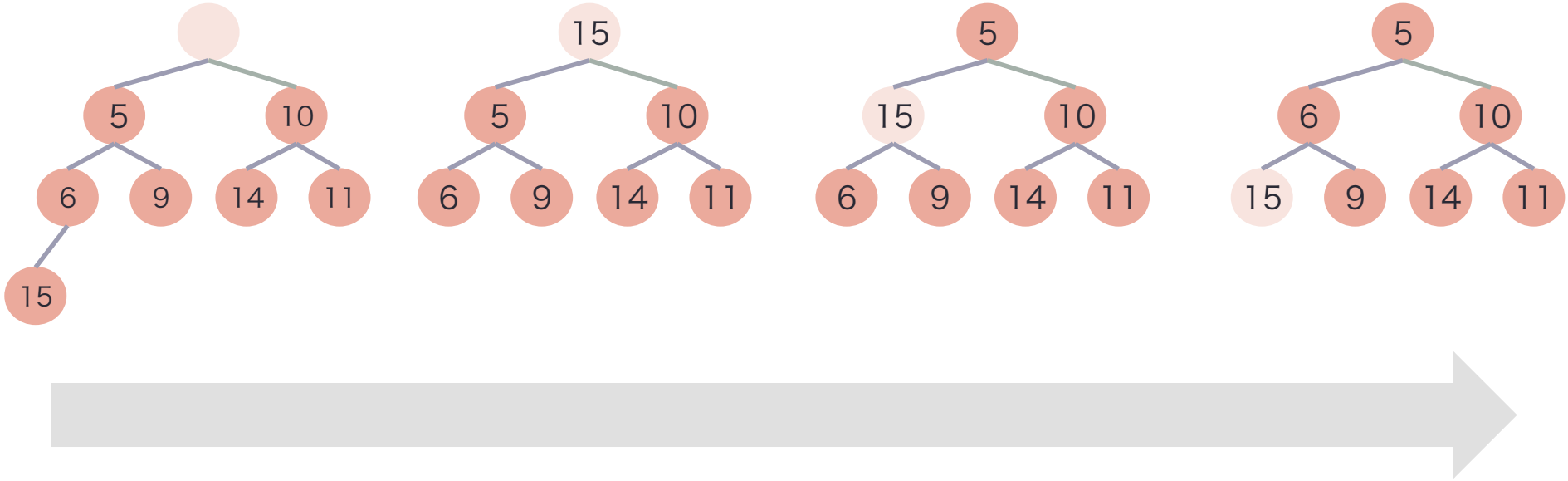
データの挿入

- 最下位のノードとして新たなデータgをおく
- 親ノードとの比較
  →ヒープ条件を満たさなかったら親と子を入れ替え



最小データの削除

- 最下位のデータを根ノードに移す
- 子ノードのうち小さい方と比較
  →ヒープ条件を満たさなかったら入れ替え



補足

- 計算量はノード数Nとして、
  ヒープなし：$O(N^2)$
  ヒープあり：$O(N * \log N)$

- アルゴリズムの中の∞は、実際には十分大きな値Bで代用

- c_jが∞のままプログラムが終了→c_jはoとつながっていない

ラベル修正法

- ダイクストラ法のStep 2はLabelingという操作

ノードiから出る全てのリンクの終点ノード{m}について
$c_m > c_i + t_{im}$ ならば $c_m = c_i + t_{im}$ と更新、$F_m = i$とする

→各ノードについてラベルを順に確定させていき、全ノードで
$c_m > c_i + t_{im}$ となるラベルがなくなった時点で終了

- ラベル修正法では、ある規則で作成されるリンクリストの順にラベリングを行うことで、最小経路を求める

- リストアップを工夫して効率化

ラベル修正法「幅優先探索」

Step 1

- 全てのノードjについて　$c_j = \infty$, $F_j = 0$, $j \in \overline{K}$
- $c_o \equiv 0$, $i \equiv o$とし、{ノードリスト}にノードoを登録

Step 2

- {ノードリスト}の先頭ノードiを取り出し{ノードリスト}から削除

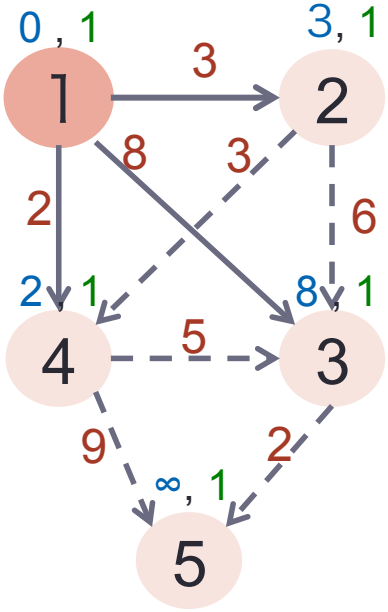
Step 3

- ノードiから出る全てのリンクの終点ノード{m}について
  $c_m > c_i + t_{im}$ ならば $c_m = c_i + t_{im}$ と更新、$F_m = i$ とし、ノードmが{ノードリスト}に含まれていなければ最後尾に登録

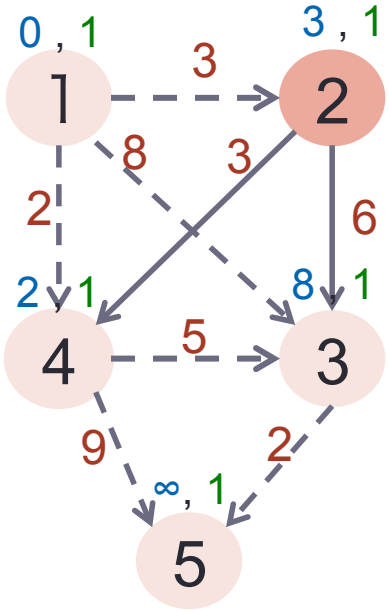
Step 4

- {ノードリスト}が空集合なら終了
- そうでなければStep 2へ

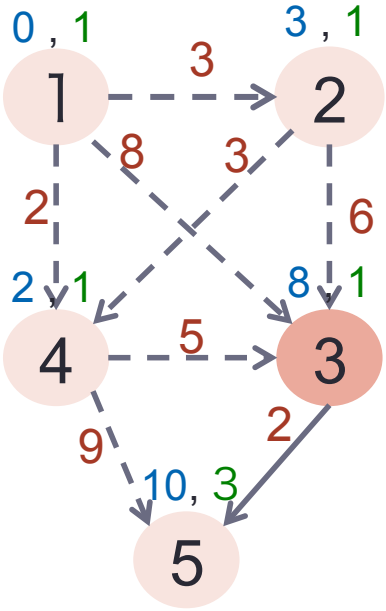
簡単な例



{2,3,4}

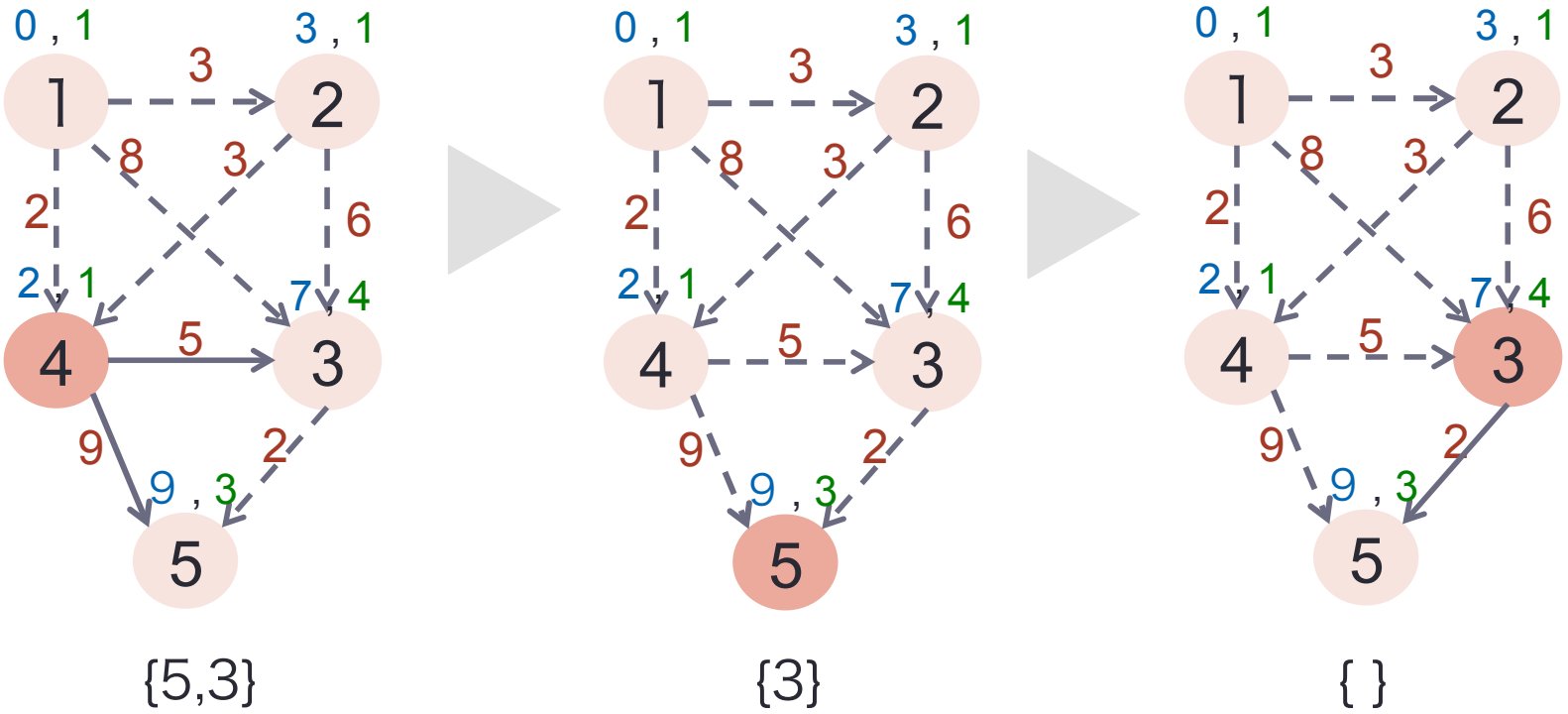


{3,4}



{4,5}

簡単な例



Dial’s algorithm

Dialのアルゴリズムの導入

- 確率的利用者均衡配分(SUE)の解法の一部として使われる
- 経路列挙は避けたい…
- ある基準を満たす「妥当な」経路のみを考慮することで、効率的にロジット型確率配分と整合するリンク交通量パターンを計算できる

Dialのアルゴリズム

Step 0(準備)

(a) 起点r から他の全ノードへの最小交通費用c(i)を計算

$$c(i) \leftarrow \min[r \rightarrow i]$$

(b) 全リンクについてリンク尤度L[i → j]を計算 (θ > 0)

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

Dialのアルゴリズム

Step 1 前進処理(forward pass)

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・ウェイトを計算

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

※I_iはノードiへ流入するリンクの起点集合

Dialのアルゴリズム

Step 2 後退処理(backward pass)

c(i)の値の降順(rから遠い順)にノードを考え、
各ノードjに流入するリンク交通量x_{ij}を次式により計算

$$x_{ij} = (q_{rj} + \sum_{m \in O_j} x_{jm}) \frac{W[i \rightarrow j]}{\sum_{m \in I_j} W[m \rightarrow j]}$$

※ O_jはノードjから流出するリンクの終点集合

リンク尤度の特色

- リンクi→jが起点からノードjまでの最短経路上にある場合、
  そのリンク尤度は最大値1

  ※ $c(j) = c(i) + t_{ij}$となる

- リンクiを通ってノードjに到達する経路がより遠回りであるほどリンク尤度は小さくなる

  ※ $c(j)$ と $c(i) + t_{ij}$の差が大きくなる

- ノードiからjへ進むと起点からの最短距離で考えて後戻りする場合、リンクi→jのリンク尤度は0

  ※ $c(i) \geqq c(i)$

リンク尤度の特色

つまりこの規則に従う配分は、

- 全経路の中で最短経路の選ばれる確率が最大
- コストの大きい経路ほど選択率が小さくなる
- 起点から「遠ざかる」順にノードをつなげたパスのみが候補となる

という特徴を持つ

Sheffi本でのDial's algorithm

- Step 1が異なる double-pass algorithmを利用

(a) 起点r からと終点sまでの最小交通費用r(i), s(i)を計算

(b) 全リンクについてリンク尤度L[i → j]を計算 (θ > 0)

$$L[i \rightarrow j] \begin{cases} \exp[\theta\{r(j) - r(i) - t_{ij}\}] & \text{if } r(i) < c(j) \text{ and } s(i) > s(j) \\ 0 & \text{otherwise} \end{cases}$$

- single-pass algorithmの方が効率性は高い
- double-pass algorithmはネットワークの対称性をフローに反映できる

- double-pass algorithm が single-pass algorithmよりも優れている保証はない (by Sheffi)

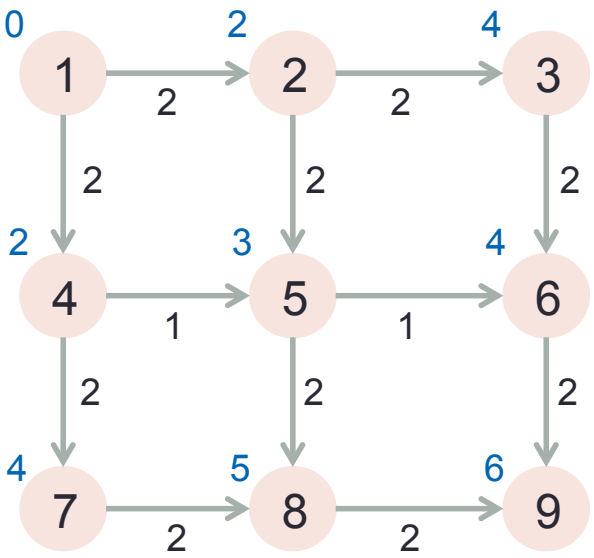
具体例

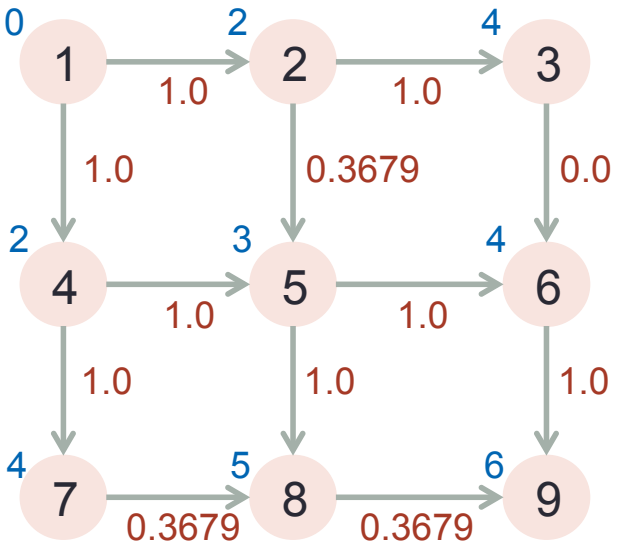
ノード1→ノード9へのパスを考える (θ=1)

Step 0:
(a) 始点から各ノードへの最小費用を計算

Step 0:
(b) 各リンク尤度を計算

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





具体例

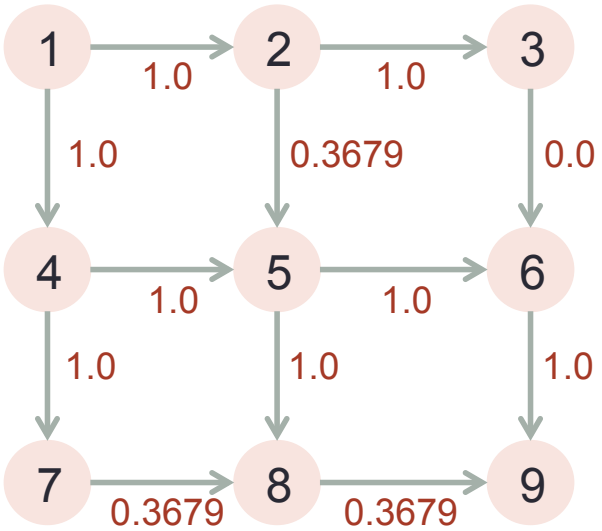
Step1 (forward pass) :

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・ウェイトを計算

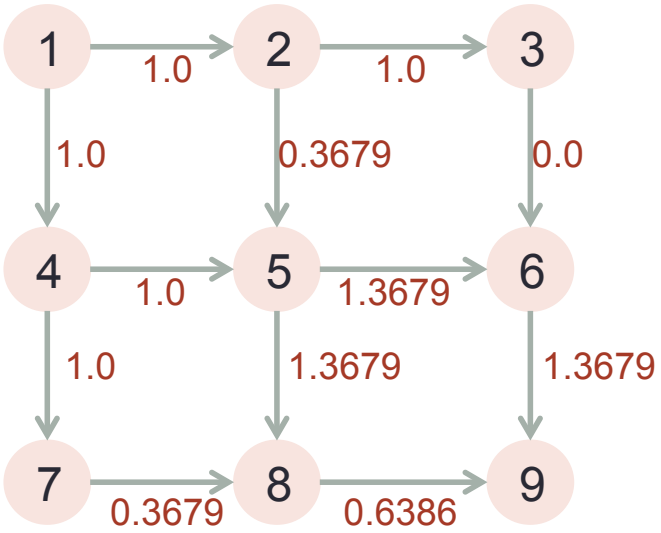
$$W[i \to j] = \begin{cases} L[i \to j] & \text{for i = r} \\ L[i \to j] \sum_{m \in I_i} W[m \to i] & \text{otherwise} \end{cases}$$

前から後ろへ計算

数値はリンク尤度



数値はリンク・ウェイト



具体例

Step2 (backward pass) :

c(i)の値の降順(rから遠い順)にノードを考え、

各ノードjに流入するリンク交通量x_{ij}を計算

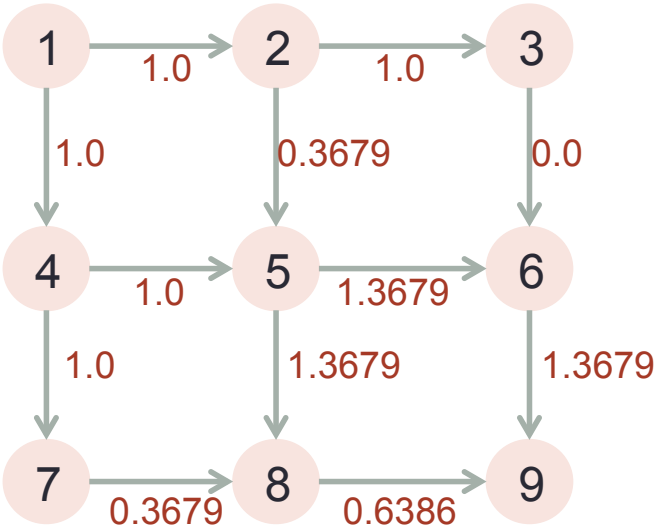
$$x_{ij} = (q_{rj} + \sum_{m \in O_j} x_{jm}) \frac{W[i \to j]}{\sum_{m \in I_j} W[m \to j]}$$

$$x_{89} = 1000 \times \frac{0.6386}{0.13679 + 0.6386} = 318$$

$$x_{78} = 318 \times \frac{0.3679}{0.3679 + 1.3679} = 67$$

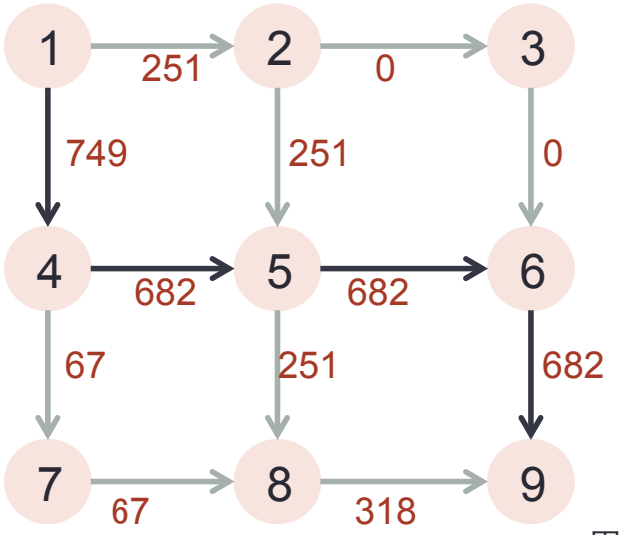
のように後ろから前へ計算

数値はリンク・ウェイト



q = 1000

数値はリンクフロー



黒線が最短経路

ロジットモデルとの等価性

- 一つのODペアに注目し、k番目のパスがr→A→B→…→Y→Z→sであるとき、Dialのアルゴリズムでこのパスが選択される確率は

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

- ここでリンク・ウェイトの定義より$i \neq r$のとき

$$W[i \to j] = L[i \to j] \sum_{m \in I_i} W[m \to i]$$

- よってP_kは

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

ロジットモデルとの等価性

- さらにこの式の分子はリンク尤度の定義より

$$L[Z \to s]L[Y \to Z]\cdots L[A \to B]L[r \to A]$$

$$= \exp[\theta\{c(s) - c(Z) - t_{Zs}\}]\exp[\theta\{c(Z) - c(Y) - t_{YZ}\}]$$
$$\cdots\exp[\theta\{c(B) - c(A) - t_{AB}\}]\exp[\theta\{c(A) - c(r) - t_{rA}\}]$$

$$= \exp[\theta\{c(s) - (t_{rA} + t_{AB} + \cdots + t_{YZ} + t_{Zs})\}]$$

$$= \exp[\theta\{c(s) - c_k\}]$$

- フロー保存則は

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

- つまり

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

ロジットモデルとの等価性

- よって $P_k = \exp[\theta\{c(s) - c_k\}] \Big/ \sum_m W[m \to s]$

$$= \exp[\theta c(s)]\exp[-\theta c_k] \Big/ \sum_m W[m \to s]$$

$$= \frac{\exp[-\theta c_k]}{\sum_k \exp[-\theta c_k]}$$

となり、ロジット型の選択確率の形になった

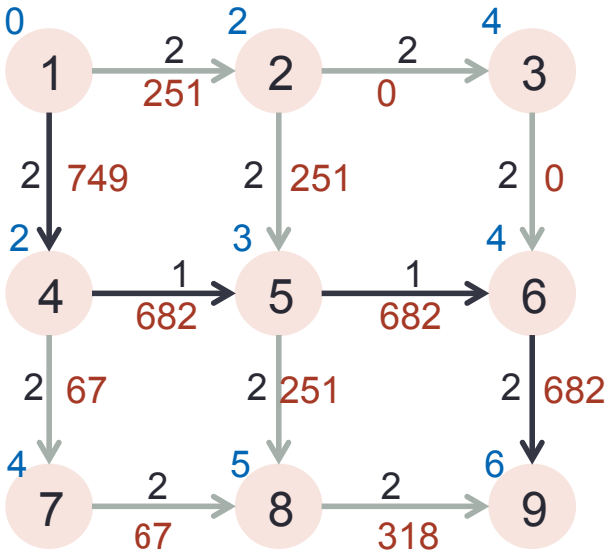
- なお、複数のODペアに拡張する際は、以上のアルゴリズムを各起点ごとに繰り返せばよい

Dial's algorithmの問題点

- 経路を暗黙裏に限定したために不自然なフローパターンを生成する場合がある

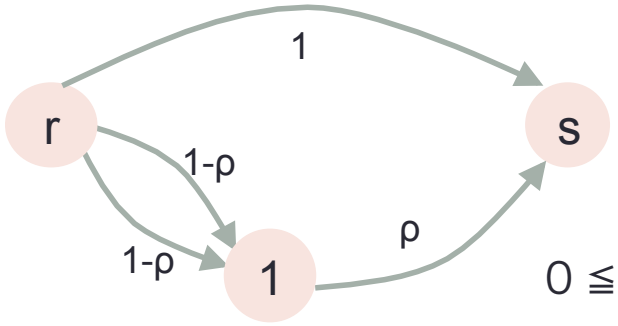
- 先ほどの例：

  1→2→3→6もありえそう…



ロジット型配分の問題 その1 (by Sheffi)

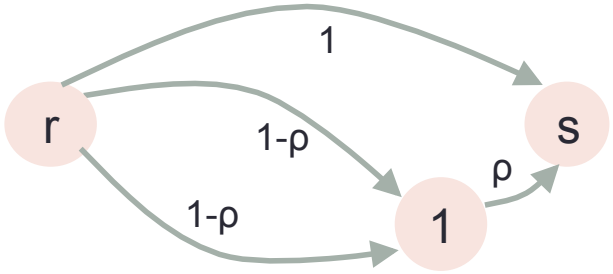
例として用いるネットワーク



$0 \leqq \rho \leqq 1$

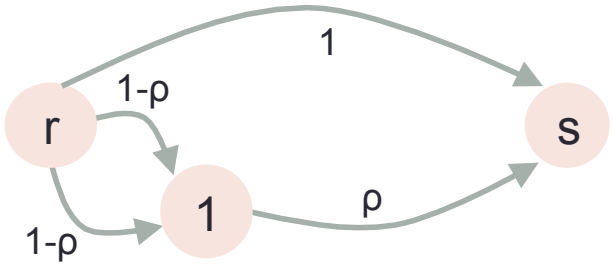
3つのパスに流れるフローは？

- $0 < \rho \ll 1$のケース



旅行者は3つの別々のパスとして認識するだろう

- $\rho \fallingdotseq 1$のケース



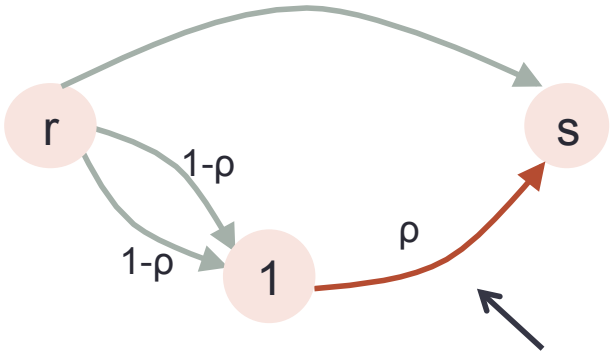
旅行者は下の2つのパスを単一の候補として認識するだろう

ロジット型配分の問題 その1 (by Sheffi)

- つまり図のネットワークでは、$0 \leqq \rho \leqq 1$のとき
  上のパスフローは1/3 ～ 1/2
  下の2つのパスフローは1/3 ～ 1/4
  で推移すると考えられる

- しかし、ロジットベースで配分を行うとρの値によらず
  各パスに1/3を配分してしまう

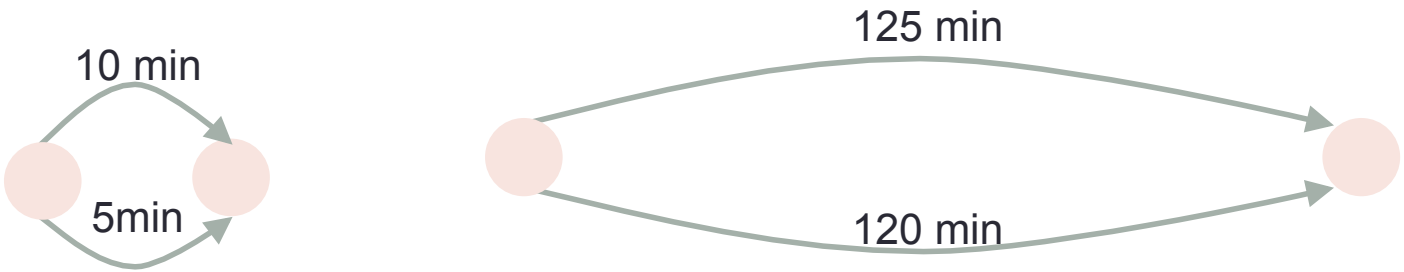
- 交通機関選択モデルにおける
  赤バス・青バス問題



このリンクの相対的な長さが
2つのパスの相関の大きさに対応

ロジット型配分の問題 その2 (by Sheffi)

例として用いるネットワーク



ほぼ下の経路が
選ばれるだろう

上の経路も相当数
選ばれるだろう

- ロジット型のモデルでは2つの例で同一のフローパターンを生成してしまう

※リンク尤度の計算式

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

ロジット型配分の問題 その2 (by Sheffi)

- 旅行時間の「差」による評価から生まれる問題

- ロジット型：
  誤差項に同一の分布を仮定

- 実際：
  パスが長いと認識旅行時間の分散が大きい

まとめとSheffiの主張

- Sheffiが提示した問題は2つ：
  ①誤差項の分布に相関がある場合
  ②誤差項の分散が同一でない場合
  これらのケースをロジット型配分は上手く扱えない…

- しかし
  ロジット型配分　→　計算の効率性が高く優秀

- プロビット型の配分　→　ロジット型の欠点を緩和