システム最適化特論

前期前半 月 5, 6限 14:00-16:10 5号館 第16講義室

担当:平田 健太郎

7/1 内点法（続）

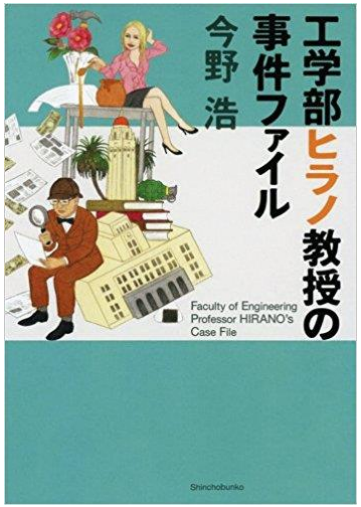
最短経路問題と動的計画法（DP）

講義日程（予定）

6/17　第１回　最適化問題と線形計画法（LP）
6/24　第２回　内点法
7/1　第３回　最短経路問題と動的計画法（DP）
7/8　第４回　最適制御
7/18* 第５回　二次計画法(QP)とモデル予測制御(MPC)
7/22　第６回　凸解析と線形行列不等式
7/29　第７回　線形行列不等式(LMI)による制御系解析・設計
8/5　第８回　非線形最適化

* irregular

カーマーカー特許とソフトウェア
　―数学は特許になるか
中公新書　今野 浩 (著)



1984年11月
米国OR学会(ダラス)にて発表.「自分の解法の出現によって新しい時代の幕が開いた」と述べるが, 専門家からの技術的質問には一切答えないというルール破りの行動に出た.

COMBINATORICA 4 (4) (1984) 373—395

A NEW POLYNOMIAL-TIME ALGORITHM FOR LINEAR PROGRAMMING

N. KARMARKAR

Received 20 August 1984
Revised 9 November 1984

We present a new polynomial-time algorithm for linear programming. In the worst case, the algorithm requires $O(n^{3.5}L)$ arithmetic operations on $O(L)$ bit numbers, where n is the number of variables and L is the number of bits in the input. The running-time of this algorithm is better than the ellipsoid algorithm by a factor of $O(n^{2.5})$. We prove that given a polytope P and a strictly interior point $\mathbf{a} \in P$, there is a projective transformation of the space that maps P, $\mathbf{a}$ to P', $\mathbf{a}'$ having the following property. The ratio of the radius of the smallest sphere with center $\mathbf{a}'$, containing P' to the radius of the largest sphere with center $\mathbf{a}'$ contained in P' is $O(n)$. The algorithm consists of repeated application of such projective transformations each followed by optimization over an inscribed sphere to create a sequence of points which converges to the optimal solution in polynomial time.

1. Informal outline

In this Section we give an informal discussion of the main ideas involved in the algorithm. A more rigorous description will be given in the next Section.

1.1. Problem definition

サー・チャンドラシェーカル・ヴェンカタ・ラーマン
Sir **Chandrasekhara Venkata Raman** (1888- 1970)
インドの物理学者. 1930年ノーベル物理学賞受賞. ラマン効果（ラマンスペクトル）の発見者. インド本国で研究したインド人研究者としては初めてのノーベル賞受賞者.

Dr Narendra K Karmarkar

スブラマニアン・チャンドラセカール
Subramanyan Chandrasekhar (1910- 1995)
インド生まれのアメリカの天体物理学者. 恒星の終焉に関する「チャンドラセカール限界」を提唱したことで知られる. 1983年,「星の構造と進化にとって重要な物理的過程の理論的研究」でノーベル物理学賞受賞.

1984年アメリカ, ベル研究所の研究者ナレンドラ・カーマーカー（Narendra Karmarkar）によって世界初のアルゴリズム特許が出願され, 88年に特許申請が認められた. 数式（算法）は、特許にならないという常識をくつがえした.

日本にもAT&T社から「最適資源割当て方法」として出願され成立.（特許第2033073号）

1989年 AT&T LPソルバKORBXと専用コンピュータ(8.9M$)の抱き合わせ販売を開始.

【KORBX購入先】
デルタ航空: 前述
米空軍: 70,000制約式, 500,000変数の問題が解けるようになった.

Karmarkar法は, 対数障壁関数を用いた射影ニュートン障壁法と同値であることが示された (Gill *et. al.*, 1986)

$$\min 2x$$

$$s.t.\ -1 \le x \le 1$$

$$\min f(x)$$

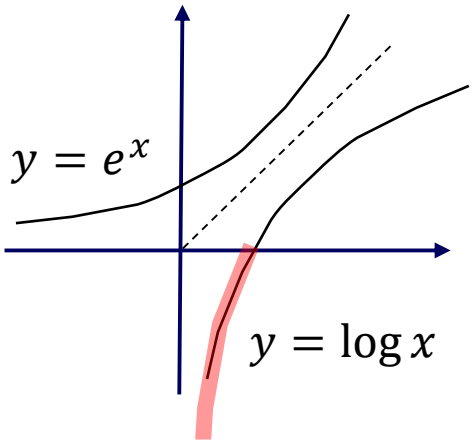
$$f(x) = 2x - \left(\frac{1}{t}\right)\log(1-x^2)$$

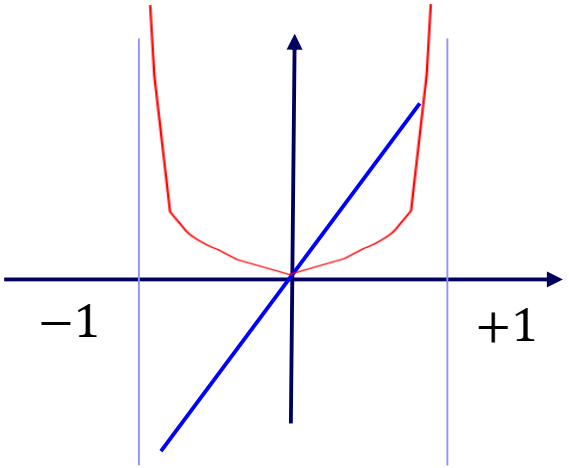
$$x \in \mathcal{R}$$

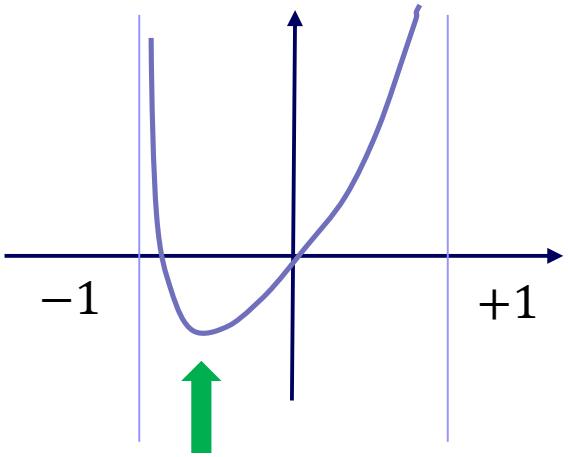
最適解： $x = -1$

$$-\left(\frac{1}{t}\right)\log(1-x^2)$$

$f(x)$







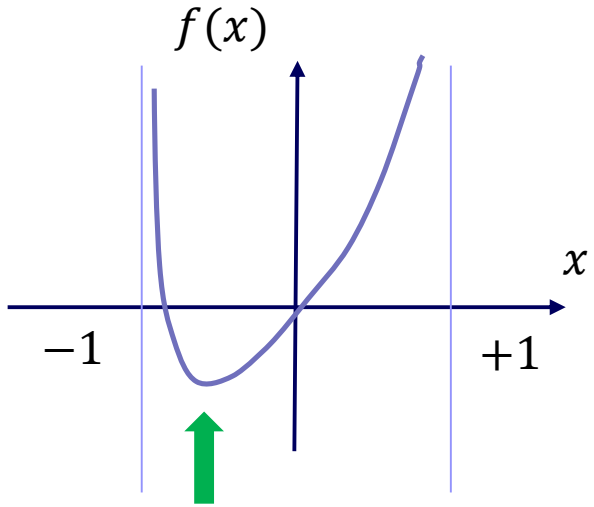
t が小さいほど± 1付近の赤の傾きは急激に

$f(x)$ を最小にする解は原問題の最適解に近づく

$$\min f(x)$$

$$f(x) = 2x - \left(\frac{1}{t}\right)\log(1-x^2)$$

$$x \in \mathcal{R}$$



$f(x)$ を最小にする解は原問題の最適解に近づく

$$f'(x) = 2 - \frac{1}{t}\frac{(-2x)}{1-x^2} = 0$$

$$\Rightarrow \quad 2t(1-x^2) + 2x = 0$$

$$\Rightarrow \quad x^2 - \left(\frac{1}{t}\right)x - 1 = 0 \quad \Rightarrow \quad x = \frac{\frac{1}{t} \pm \sqrt{\frac{1}{t^2}+4}}{2}$$

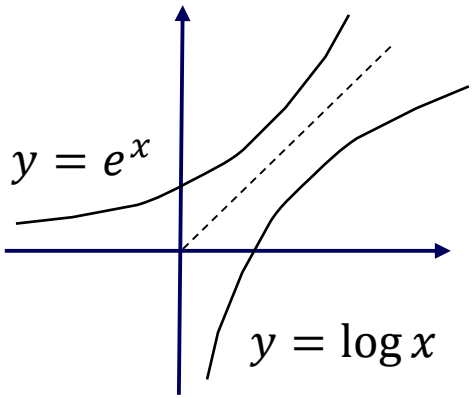
$x > 0$ の解は $x \notin [-1,1]$

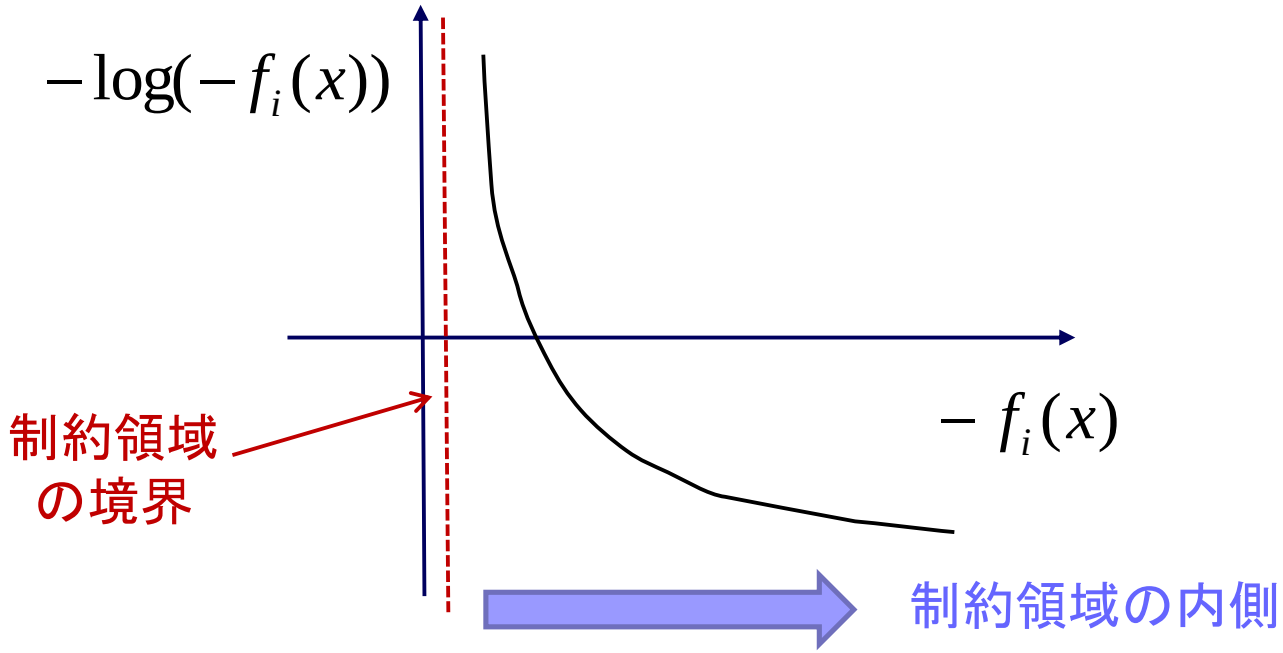
$x \to -1 + 0 \ (t \to 0+)$

バリア関数（ペナルティ）法

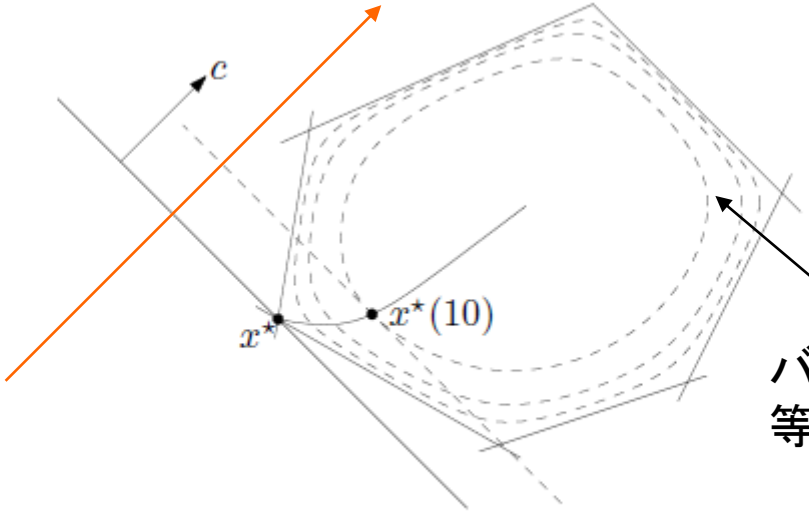
制約条件： $f_i(x) \leq 0, \quad i = 1, \ldots, m$

バリア関数： $\sum_{i=1}^{m} -(1/t)\log(-f_i(x))$





バリア関数と(線形)評価関数を合成



バリア関数の
等高線

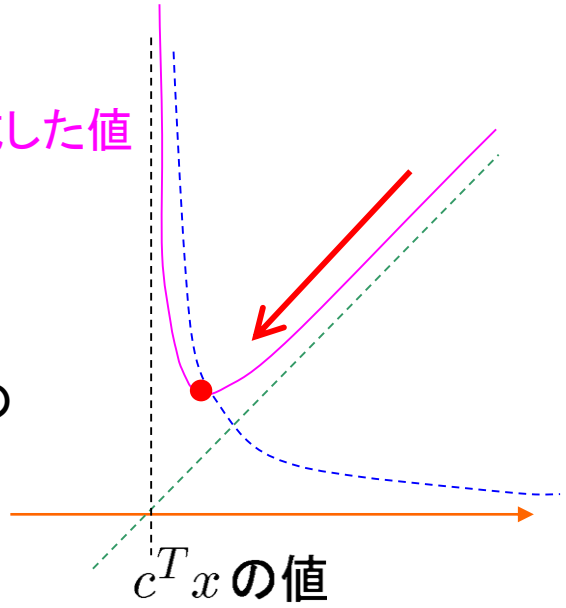


Figure 11.2 Central path for an LP with $n = 2$ and $m = 6$. The dashed curves show three contour lines of the logarithmic barrier function ϕ. The

$$\text{minimize} \quad f_0(x) + \sum_{i=1}^{m} -(1/t)\log(-f_i(x))$$

バリア関数（ペナルティ）法

原問題 (制約つき)

$$\begin{array}{ll}\text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le 0, \quad i = 1, \ldots, m\end{array}$$

バリア関数 $I_-(u) = \begin{cases} 0 & u \le 0 \\ \infty & u > 0. \end{cases}$

制約なし問題 （等価）

$$\text{minimize} \quad f_0(x) + \sum_{i=1}^m I_-(f_i(x))$$

対数バリア関数（微分可能）

$$\widehat{I}_-(u) = -(1/t)\log(-u),$$

制約なし問題 （近似）

$$\text{minimize} \quad f_0(x) + \sum_{i=1}^m -(1/t)\log(-f_i(x))$$

制約なし非線形最適化

Karmarkar法

射影変換法

アフィン変換法

60年代に先行研究(ディキン(ロシア))

AT&Tの特許戦略: 制約領域の内部を通るものなら
どんなものであれ特許侵害とみなす

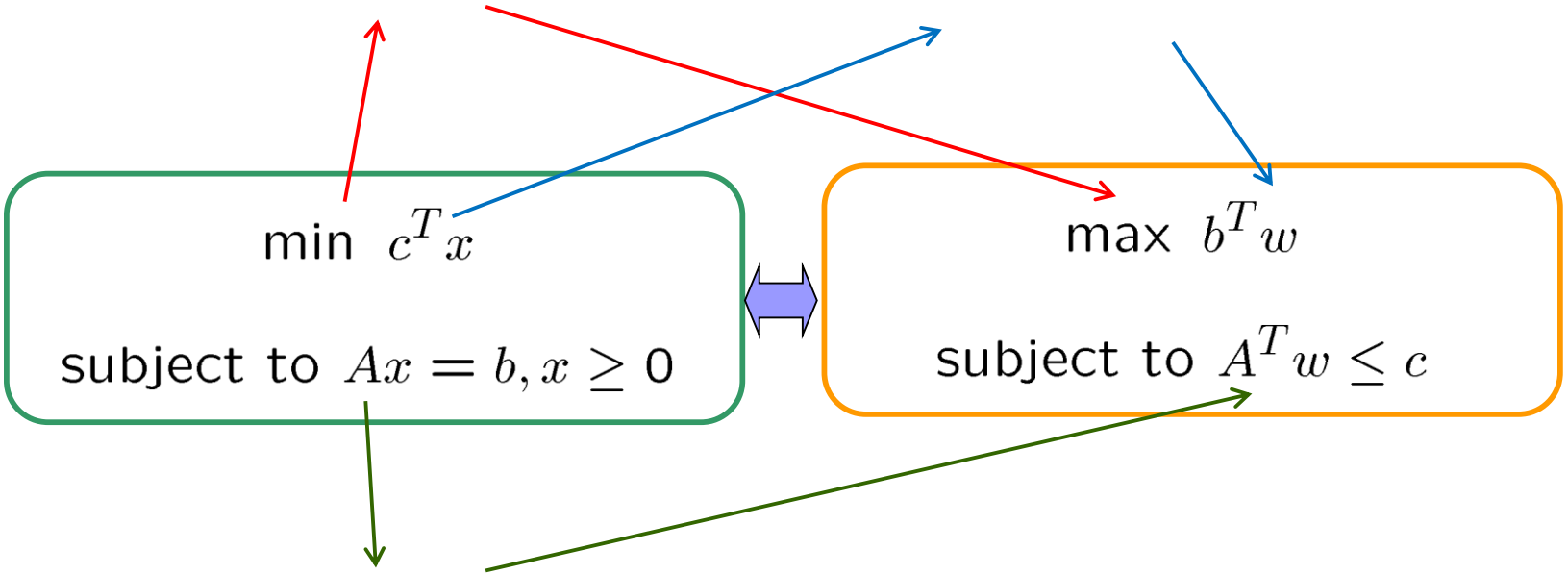
主双対内点法 → 対抗ソフト　OB1 (5万ドル) 登場

線形計画問題における双対問題

変数 $x \to w$

「最小化」を「最大化」に

目的関数の係数　$c \to b$



制約条件　$A \to A^T$, $b \to c$, 「$=$」$\to$「$\leq$」

非負制約　あり$\to$なし

主問題　primal problem (P)　　双対問題 dual problem (D)

LPにおける双対問題

$$\min \ c^T x$$

$$\text{subject to } Ax = b, x \geq 0$$

主問題 primal problem (P)

$$\max \ b^T w$$

$$\text{subject to } A^T w \leq c$$

双対問題 dual problem (D)

弱双対定理: (P) と (D) それぞれの任意の実行可能解 x, w に対して，常に不等式 $c^T x \geq b^T w$ が成り立つ．

証明

双対定理: (P) または (D) の一方が最適解をもてば他方も最適解をもち，(P) の最小値と (D) の最大値は等しい．

証明（PからD）

主双対内点法へ！

パス追跡・主双対内点法

$$\min \ c^T x$$

$$\text{subject to } Ax = b, x \geq 0$$

主問題　primal problem (P)

$$\max \ b^T w$$

$$\text{subject to } A^T w \leq c$$

双対問題 dual problem (D)

スラック変数sを用いて等式化

$$\max \ b^T w$$

$$\text{subject to } A^T w + s = c, \ s \geq 0$$

双対問題（D’）

双対定理より

(x, w, s) が次式を満たす

$$\begin{cases} c^T x = b^T w \\ Ax = b \\ A^T w + s = c \\ x \geq 0, \;\; s \geq 0 \end{cases} \quad \Longleftrightarrow \quad x \text{ と } (w, s) \text{ が (P), (D') の最適解}$$

スカラ量

$$c^T x - b^T w = (A^T w + s)^T x - (Ax)^T w = w^T Ax + s^T x - x^T A^T w = s^T x$$

$x \geq 0, s \geq 0$ より第1式は $s_j x_j = 0, \forall j$，あるいは

$$XSe = 0, \quad X = \begin{bmatrix} x_1 & & 0 \\ & \ddots & \\ 0 & & x_n \end{bmatrix}, \quad S = \begin{bmatrix} s_1 & & 0 \\ & \ddots & \\ 0 & & s_n \end{bmatrix}, \quad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

少なくとも一方は0： 相補性条件

線形相補性問題

$$\begin{cases} XSe = 0 \\ Ax = b \\ A^T w + s = c \\ x \geq 0,\ \ s \geq 0 \end{cases} \cdots(1) \quad \Rightarrow \quad \begin{cases} XSe = \mu e \\ Ax = b \\ A^T w + s = c \end{cases}$$ 非線形 $\cdots(2)$

$x \geq 0, s \geq 0$ を $x > 0, s > 0$ とみなし，パラメータ $\mu > 0$ を用いて相補性条件 $x_j s_j = 0$ を $x_j s_j = \mu$ で置き換え

(2) において $\mu \to 0$ とした極限が線形相補性問題 (1) の解

パス追跡法

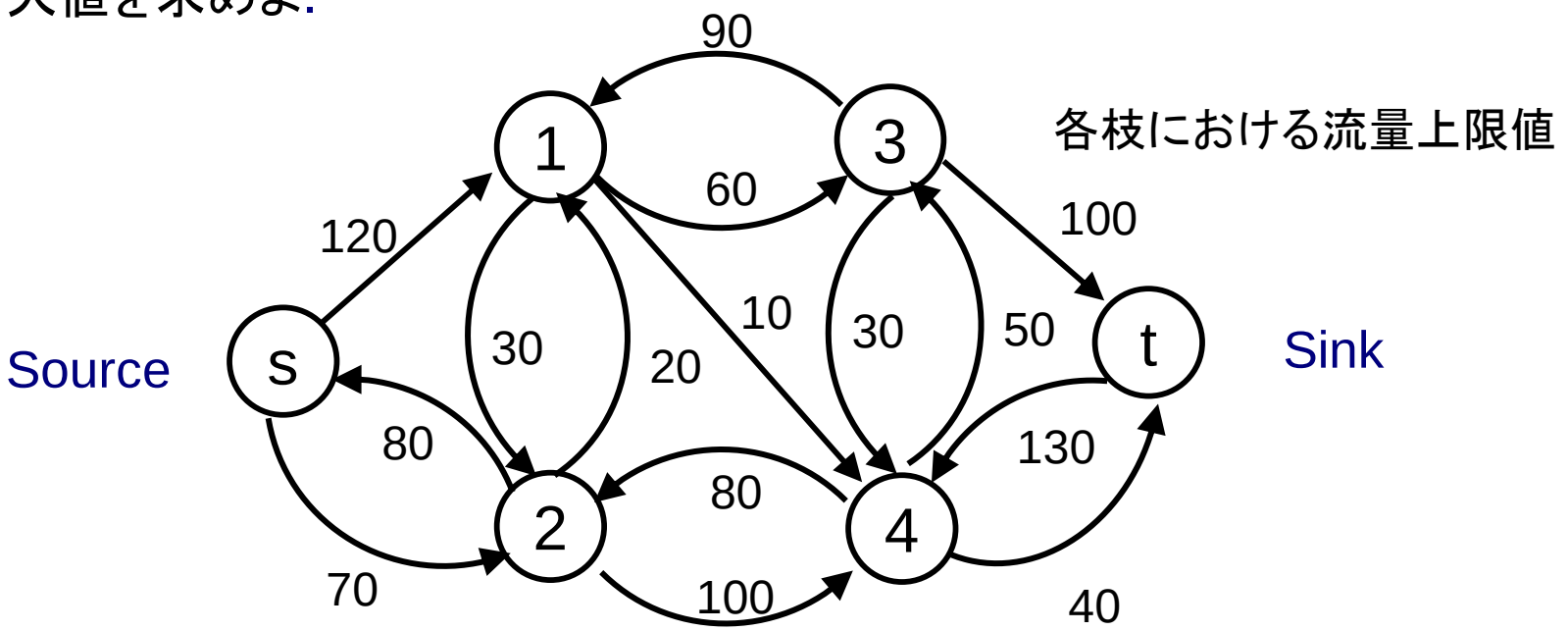
0 に収束する正数列 $\{\mu^{(k)}\}$ を考え，各 $\mu^{(k)}$ に対する非線形連立方程式 (2) の解 $(x(\mu^{(k)}), w(\mu^{(k)}), s(\mu^{(k)}))$ を近似的に求め，(1) の解に収束する点列を求める．

とくに変数 (x, s) に関しては常に $x^{(k)} > 0, s^{(k)} > 0$ が成り立つように点列を定める　$\Rightarrow$ 内点法

ネットワークに関連した他の線形計画問題

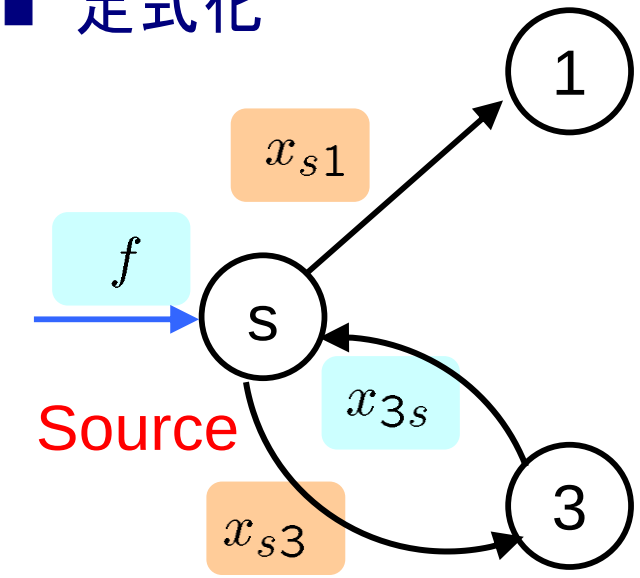
- 最大流問題

各枝の流量制約下でソースからシンクへ流すことのできるフローの最大値を求めよ.



V: 節点(Vertex)の集合, E: 枝(Edge)の集合, s: ソース, t: シンク,
x_{ij}: 枝(i,j)上の流れの大きさ, u_{ij}: 枝(i,j)の容量,
f: ソースからシンクへの総流量

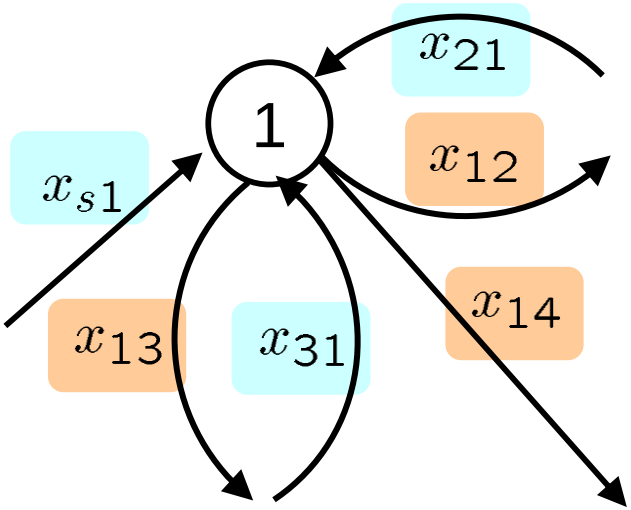
定式化



ソースでの流れの保存則

$$\sum_{\{j|(s,j)\in E\}} x_{sj} - \sum_{\{j|(j,s)\in E\}} x_{js} = f$$

シンクでも同様



中間ノードでの流れの保存則

$$\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = 0,$$

$$(i \in V - \{s,t\})$$

- 最大流問題

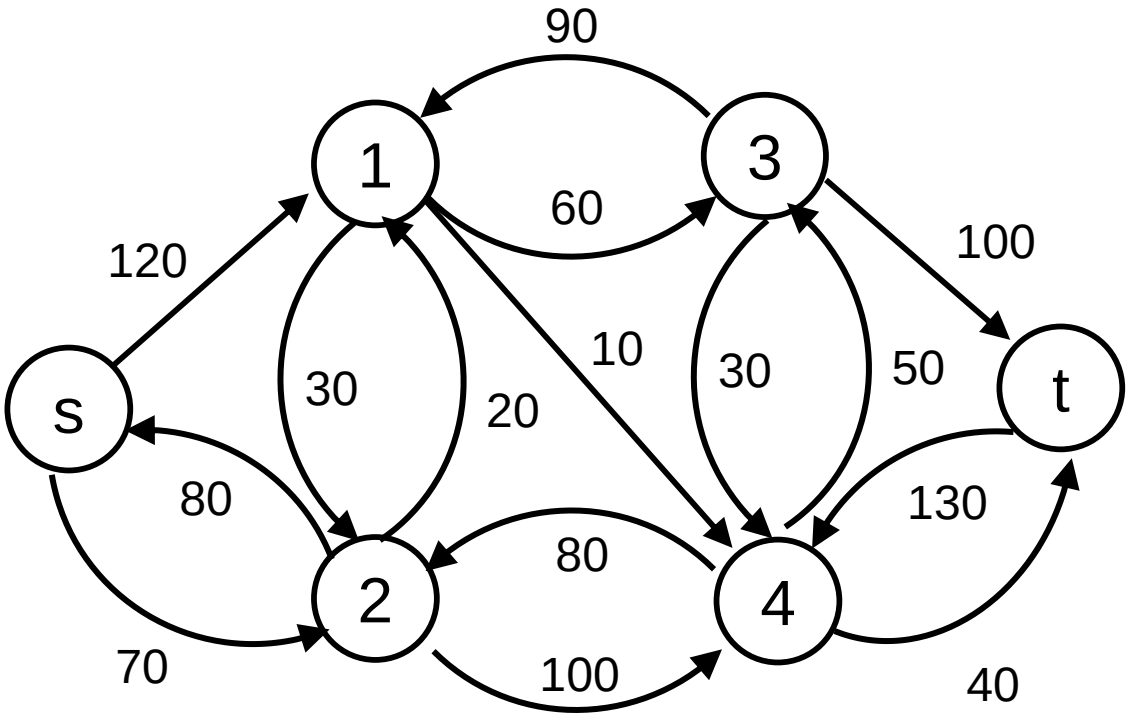
$$\max \quad f$$

$$\begin{aligned}
\textbf{subject to} \quad & \sum_{\{j|(s,j)\in E\}} x_{sj} - \sum_{\{j|(j,s)\in E\}} x_{js} = f \\
& \sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = 0, \ (i \in V - \{s,t\}) \\
& \sum_{\{j|(t,j)\in E\}} x_{tj} - \sum_{\{j|(j,t)\in E\}} x_{jt} = -f \\
& 0 \le x_{ij} \le u_{ij} \ ((i,j) \in E)
\end{aligned}$$

⇒$\{x_{ij}, f\}$ を変数とする線形計画問題

最小費用流問題

最大流問題の条件に加えて，各枝における単位流量あたりのコストが与えられている．



	s	1	2	…
s		2	4	
1	5		1	
2	3	2		
	4	1	3	

ソースからシンクへのフローは与えられているとする． 各枝の輸送量制約下で定められた量の品物をソースからシンクへ流すとき，費用が最小となる配送法を求めよ．

$G = (V, E)$: 節点 (vertex) V と枝 (edge) E からなるネットワーク,
x_{ij}: 枝 (i, j) 上の流れの大きさ, u_{ij}: 枝 (i, j) の容量,
c_{ij}: 枝 (i, j) の 1 単位の流れに対するコスト,
b_i: 節点 i における需要, 供給量

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\textbf{subject to} \sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = b_i, \ (i \in V)$$

$$0 \le x_{ij} \le u_{ij} \ ((i,j) \in E)$$

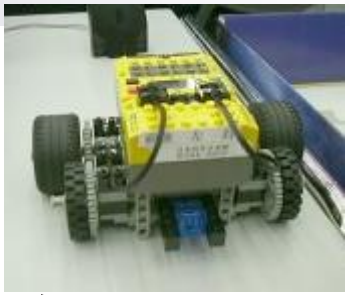
⇒線形計画問題

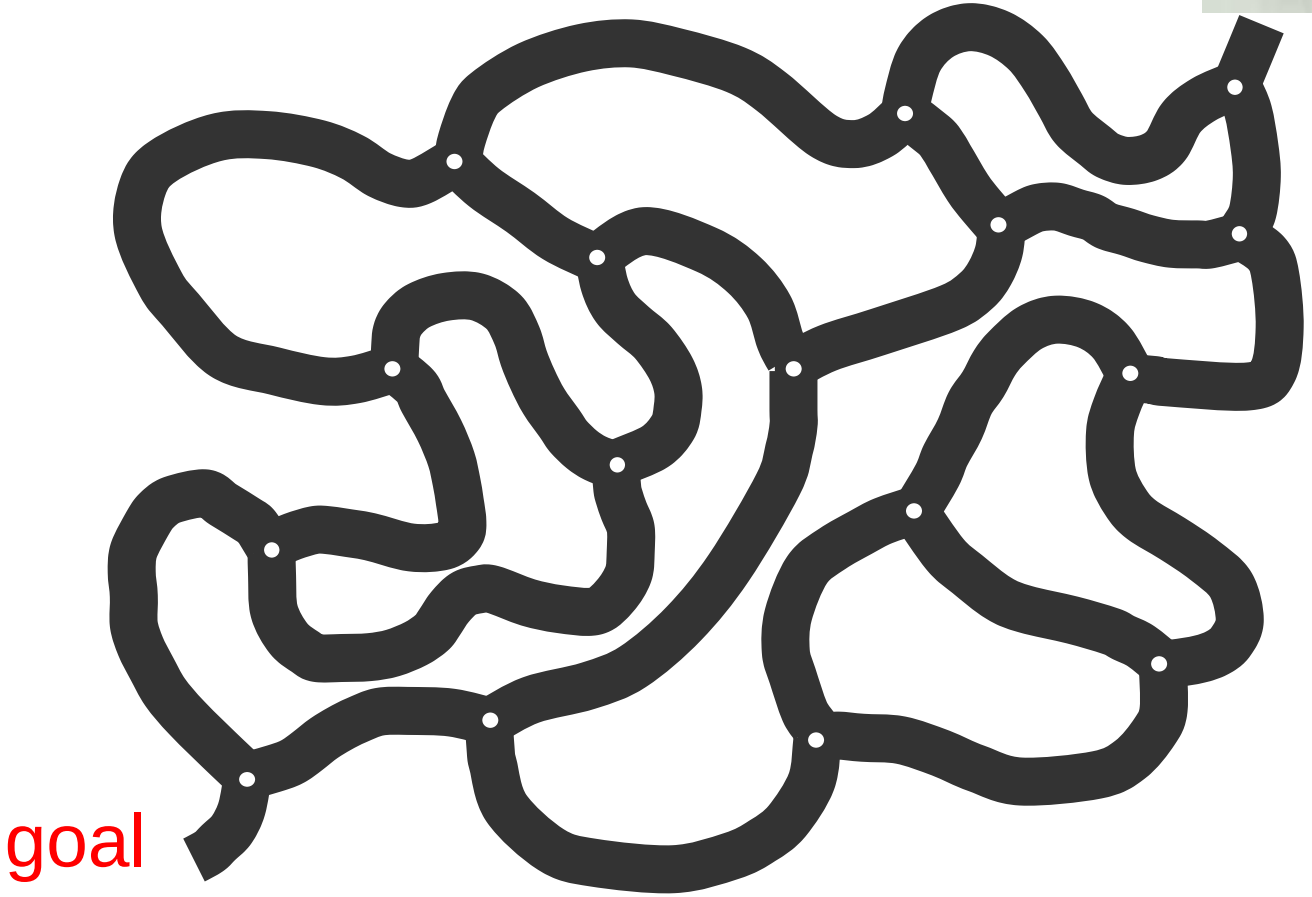
ソースとシンクのみ非零の b_i を設定し, 容量制約をなくすと輸送問題になる

最短経路問題

最短経路問題の例

ライントレース
ロボット





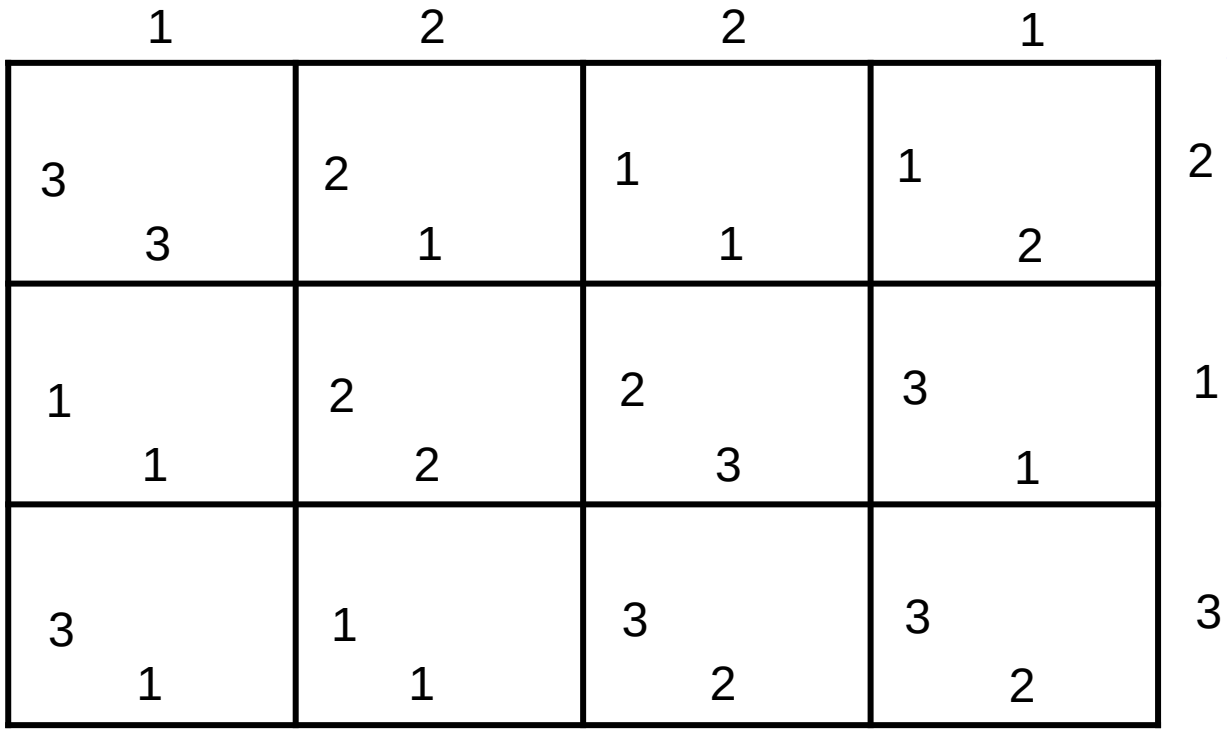
最短時間でゴールできるルートを選んで走行せよ！

最小費用流問題において，ソースとシンクのみ非零のb_iを設定，
枝上の容量制約を外し，各枝上の単位流量当たりのコストを枝の
長さ（節点間の距離）とする．

最小費用流となるには，距離の短い経路から選ばれるが，容量制
約がなければ，全流量がある経路（最短経路）に集中する．

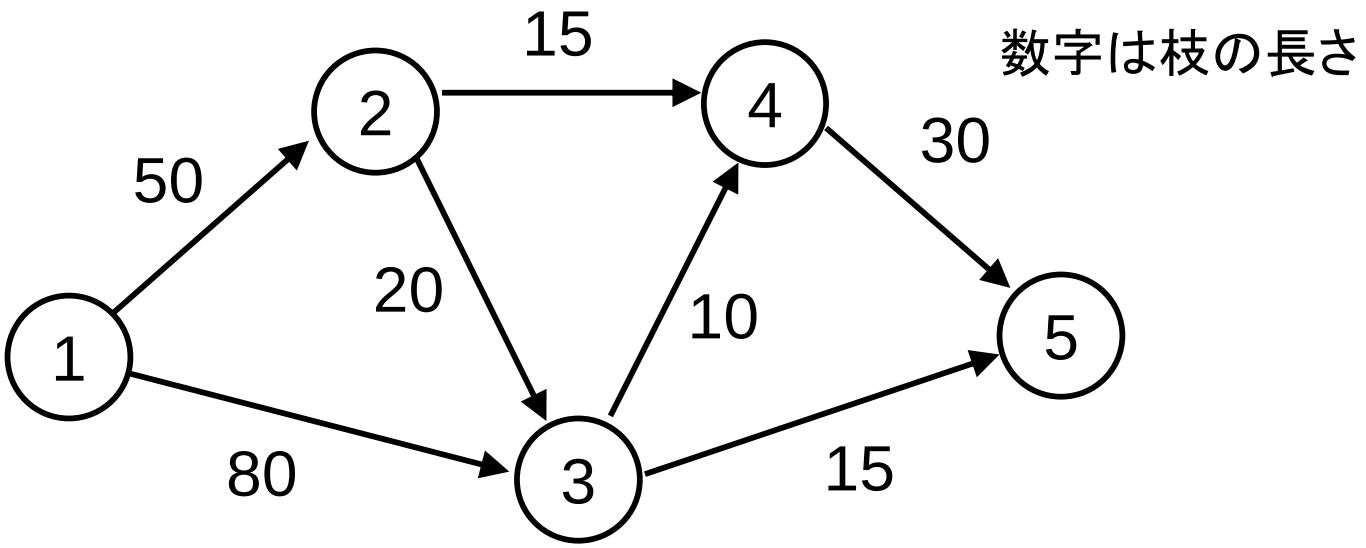
すなわち，最短経路問題はLPとして解くこともできる．

Quiz: 数値は格子点間の距離を表す．左方向，下方向以外には進めないとき，スタートからゴールまでの最小ルートの長さを求めよ．



■列挙法

節点1から5への全ての経路を数え上げて, 最短のものを求めよ.



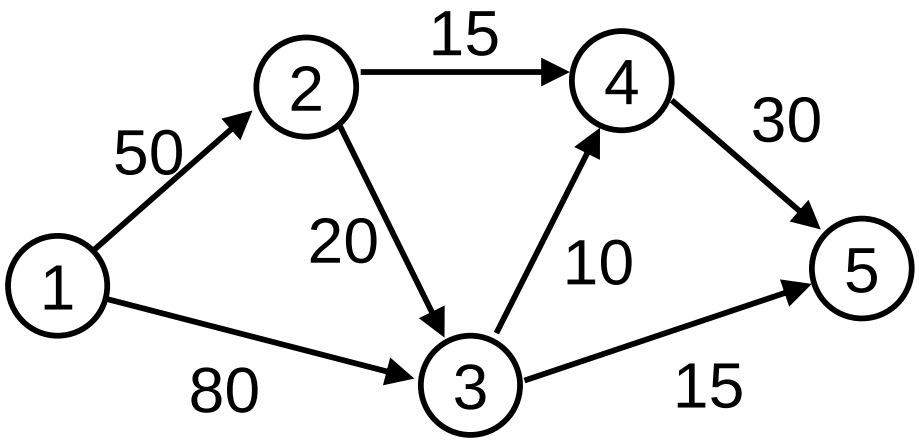
path: 1→2 →3 →4 →5, length: 110
1→2 →4 →5, 95
1→2 →3 →5, 85 ←最短
1→3 →4 →5, 120
1→3 →5, 95

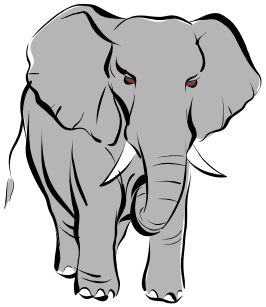
1. 7本の辺それぞれを通る/通らないとして，経路パターンを 2^7 生成

2. 1から出発して5に到達できるかチェック，到達できるなら経路長を記録

3. すべての実行可能解のうち，経路長が最小のものを出力

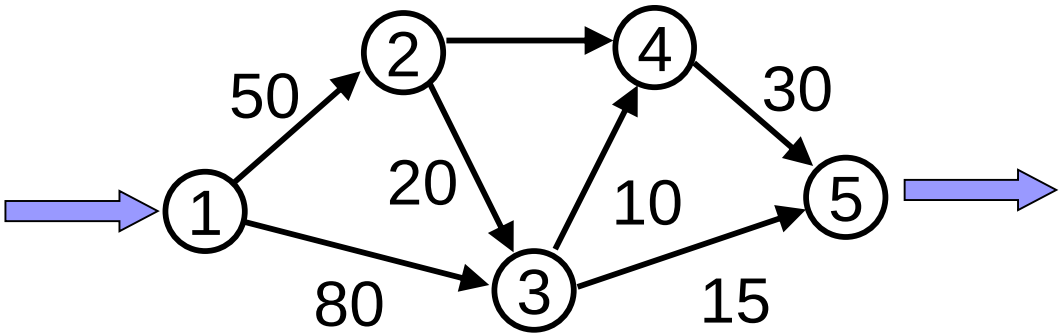
明らかに指数時間アルゴリズム





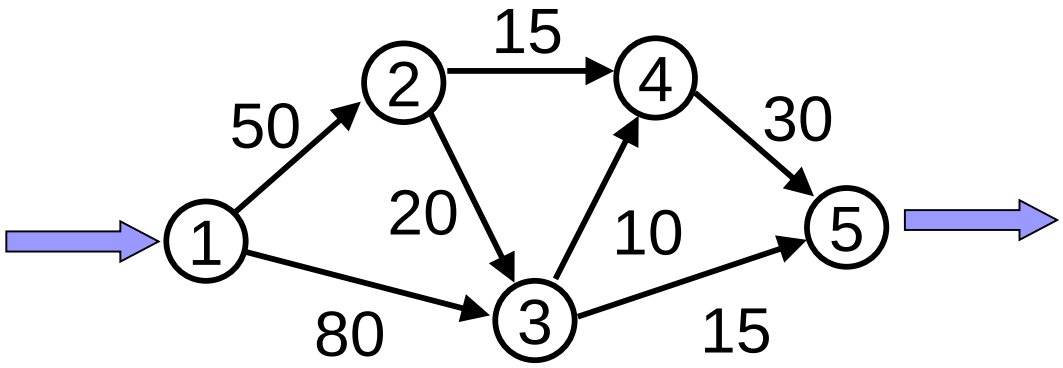
Extremely elephant!

線形計画問題として解く



最も早い主双対内点法によれば，変数の数n，入力のビット数Lに対して計算量は$\mathcal{O}(n^3 L)$になることが知られている

L は問題を計算機に記述するのに必要な総ビット数
(行列・ベクトルの要素を2進数で表現した際の総容量)



実行可能基底解が退化
（0解が7-5より多い）
⇒注意が必要

$$x := [x_{12}\ x_{13}\ x_{23}\ x_{24}\ x_{34}\ x_{35}\ x_{45}]^T,$$

$$c := [50\ 80\ 20\ 15\ 10\ 15\ 30]^T,$$

$$A := \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}, \quad b := \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix},$$

$$\min c^T x \quad \textbf{s.t.}\ Ax = b,\ x \geq 0.$$

Matlab計算例： shortest_path.m

ダイクストラ法/Dijkstra Method

多項式時間アルゴリズム $O(n^2)$

1. $S = \Phi$, $\bar{S} = V$, $d(s) = 0$, $d(i) = \infty (i \in V - \{s\})$ とする.

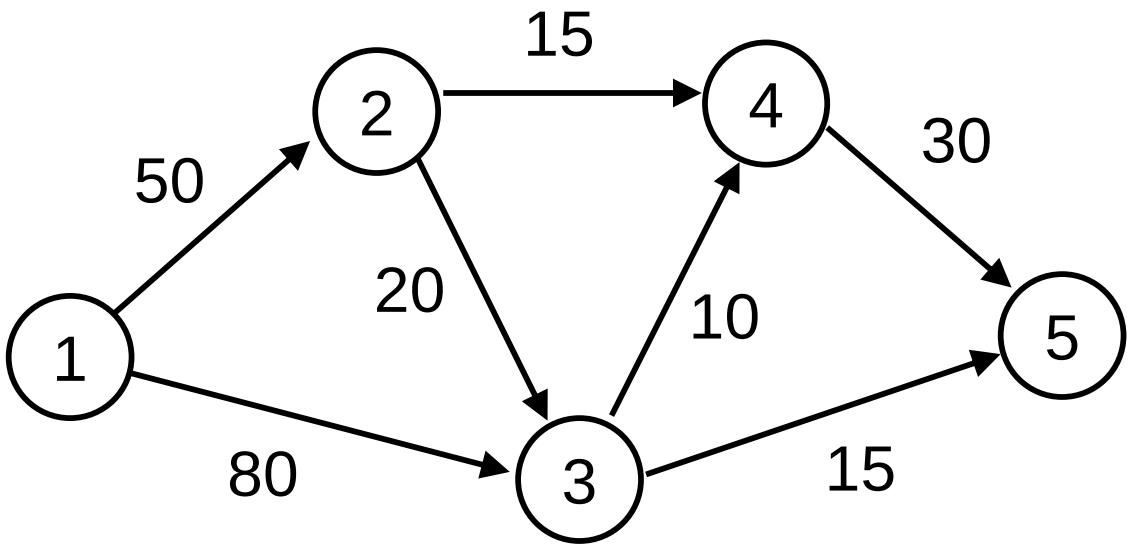
2. $S = V$ ならば停止. そうでないとき $d(v) = \min\{d(i) | i \in \bar{S}\}$ であるような $v \in \bar{S}$ を選ぶ.

3. $S = S \cup \{v\}$, $\bar{S} = \bar{S} - \{v\}$ とする. $(v, j) \in E$ かつ $j \in \bar{S}$ であるようなすべての枝に対して, $d(j) > d(v) + a_{vj}$ ならば $d(j) = d(v) + a_{vj}$, $p(j) = v$ とする.

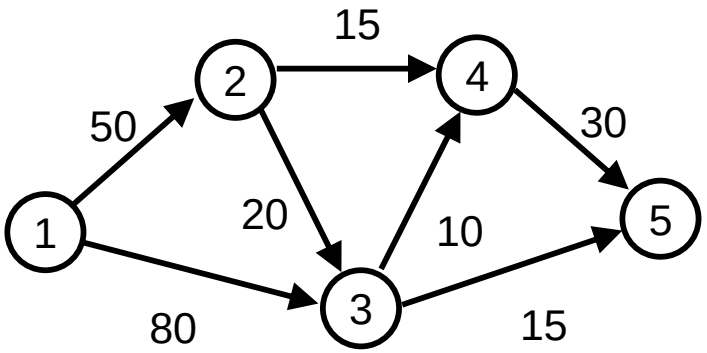
4. 2に戻る.

ダイクストラ法の大規模な実行例：java program

例題: 以下の例に Dijkstra 法を適用してみる



- $S = \Phi$, $\bar{S} = V$, $d = [0, \infty, \infty, \infty, \infty]$.

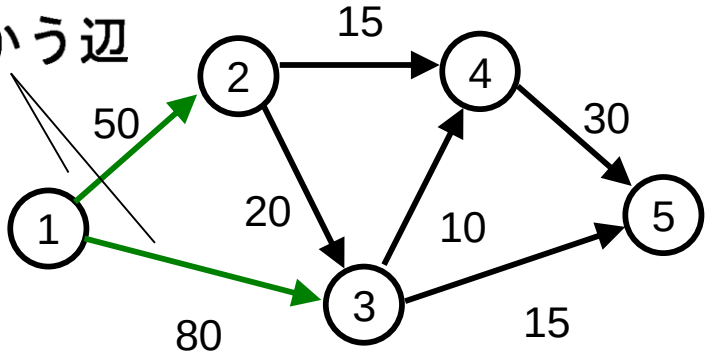


$v = \arg\min\{d(i) | i \in \bar{S}\} = 1$

ノード1を$\bar{S}$からSへ

- $S = \{1\}$, $\bar{S} = \{2, 3, 4, 5\}$

vから$\bar{S}$内へ向かう辺



$d(2) \leftarrow \min\{d(2), d(1) + a_{12}\}$

∞　　$0 + 50$

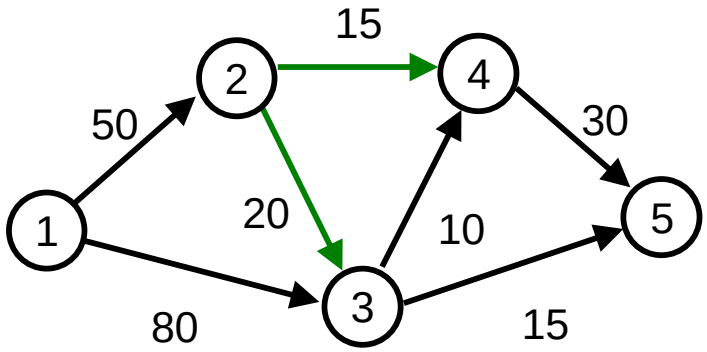
よって$d(2)$は50に更新，
同様に$d(3) \leftarrow 80$

$\Rightarrow$ $d = [0, 50, 80, \infty, \infty]$

$d = [0, 50, 80, \infty, \infty]$

$v = \arg\min\{d(i) | i \in \bar{S}\} = 2$　　ノード2を$\bar{S}$からSへ

- $S = \{1, 2\}$, $\bar{S} = \{3, 4, 5\}$



vから$\bar{S}$内へ向かう辺は$(2,3)$と$(2,4)$

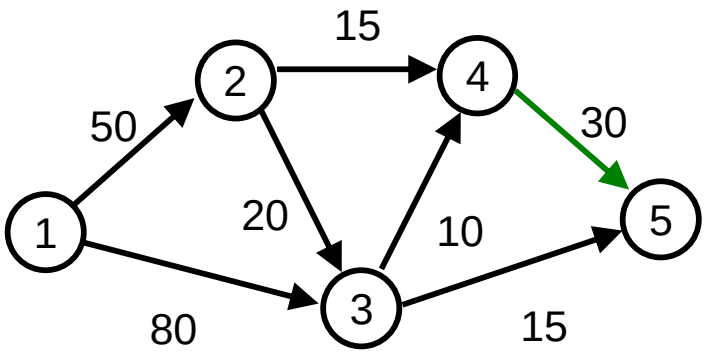
$d(3) \leftarrow \min\{d(3), d(2) + a_{23}\}$

80　　$50 + 20$

$d(3) \leftarrow 70$, $d(4) \leftarrow 65$　　$d = [0, 50, 70, 65, \infty]$

ノード4を$\bar{S}$からSへ

- $S = \{1, 2, 4\}$, $\bar{S} = \{3, 5\}$

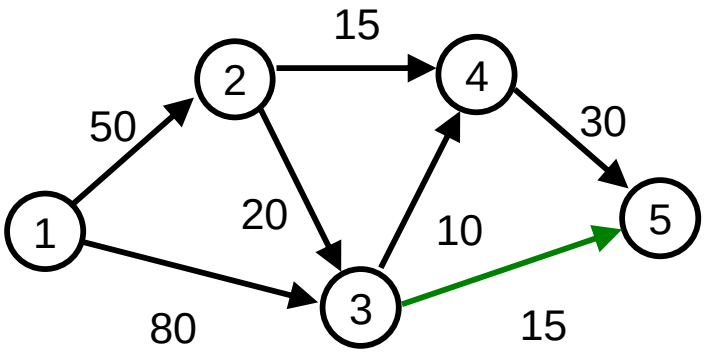


$d(5) \leftarrow 95$

$\Rightarrow$ $d = [0, 50, 70, 65, 95]$

ノード3を$\bar{S}$からSへ

- $S = \{1, 2, 3, 4\}$, $\bar{S} = \{5\}$

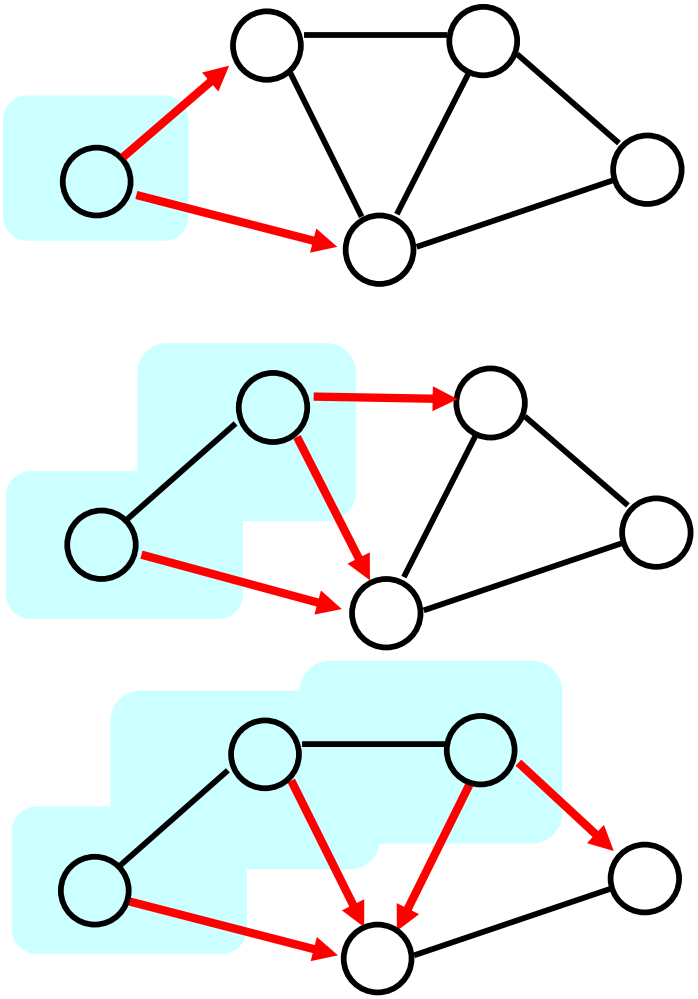


$d(5) = \min\{95, 70 + 15\} = 85$

$\Rightarrow$ $d = [0, 50, 70, 65, 85]$

ノード5をSへ．終了．

Dijkstra法の計算量



ノード数：n, エッジ本数: m

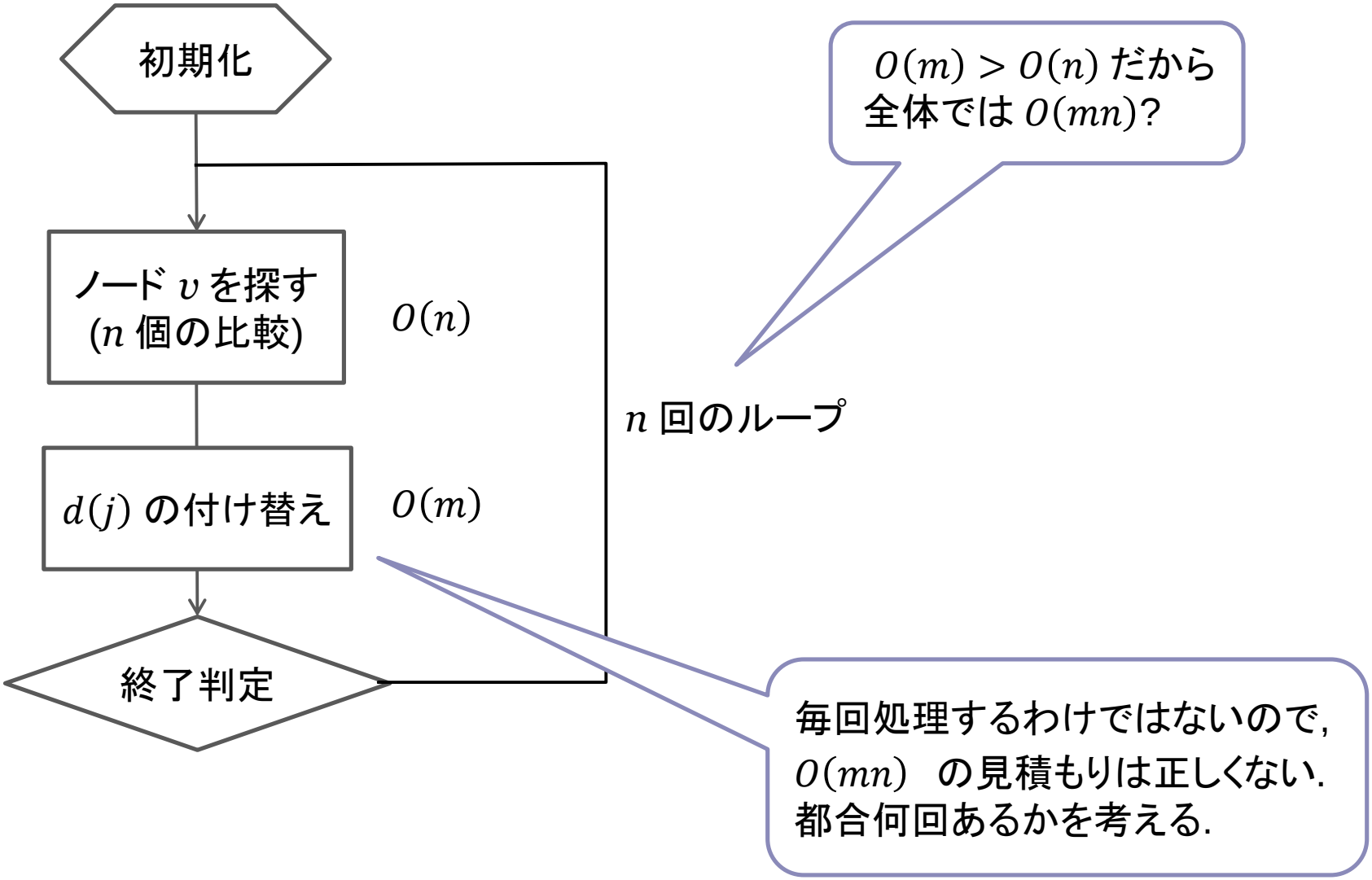
S の要素を1つずつ増やす：n 反復高々
n 個の $d(i)$ を比較

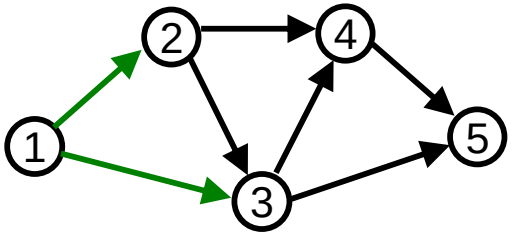
$\Rightarrow O(n^2)$　　オーダー（最高次の次数のみ係数は無視）

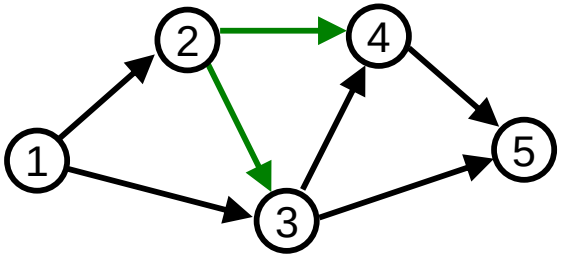
S から外に出るエッジについて $d(i)$ の更新　高々m 回の処理 $\Rightarrow$ $O(m)$

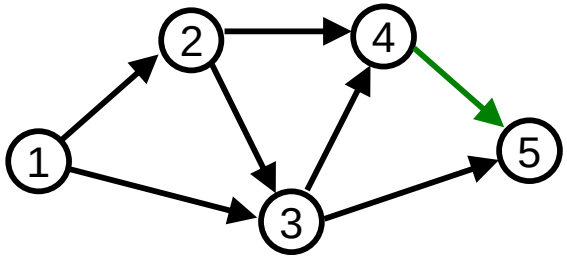
(ループ内の処理に見えるが)
エッジの本数は最大でも $m = \dfrac{n(n-1)}{2}$
(n個のノードをすべて相互につなぐとき)

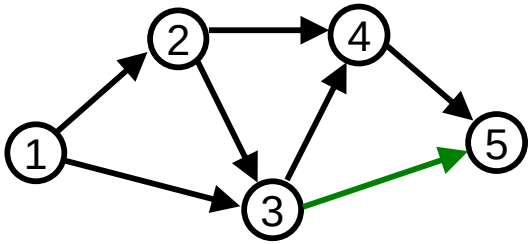
$m < n^2$ より, 全体では $O(n^2)$


初期化
ノード v を探す
(n 個の比較)
$O(n)$
$d(j)$ の付け替え
$O(m)$
終了判定
n 回のループ
$O(m) > O(n)$ だから
全体では $O(mn)$?
毎回処理するわけではないので,
$O(mn)$ の見積もりは正しくない.
都合何回あるかを考える.









$\therefore$ $d(j)$ の付け替えは各ノードに
入ってくる矢印の数だけ行われる.
(行われない場合もある.)

⇒ 最大でも辺の総数だけ