数理計画法と計算の複雑さ

筑波大学 システム情報工学研究科　　久 野　誉 人

2007 年　9 月
2013 年　4 月改訂

1　数理計画法とは何ぞや

数理計画法 (mathematical programming) とは，

> **最適化**
>
> **入力:** ユークリッド空間 $\mathbb{R}^n$ の部分集合 X，関数 $f : D \to \mathbb{R}^1$（$X \subset D \subset \mathbb{R}^n$）
>
> **出力:** 関数 f の値を最小にするベクトル $\mathbf{x}^* \in X$

を数学的に解決する方法，主として**アルゴリズム** (algorithm) をさす．この**最適化問題** (optimization problem) は**数理計画問題** (mathematical programming problem, or mathematical progeram) ともよばれ，数式で

$$\left| \begin{array}{ll} \text{最小化} & f(\mathbf{x}) \\ \text{条　件} & \mathbf{x} \in X \end{array} \right. \tag{1.1}$$

のように簡潔に表現される．関数 f を**目的関数** (objective function)，集合 X を**実行可能領域** (feasible region) とよび，X に属するベクトル $\mathbf{x} \in X$ を**実行可能解** (feasible solution) とよぶ．特に，f の値を最小にする実行可能解 $\mathbf{x}^* \in X$ を**最適解** (optimal solution)，その値 $f(\mathbf{x}^*)$ を最適値とよぶ．したがって，最適解 $\mathbf{x}^*$ に対しては

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in X \tag{1.2}$$

が成り立つが，実行可能解 $\mathbf{x}^\circ \in X$ が近傍 $B_\epsilon(\mathbf{x}^\circ) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x} - \mathbf{x}^\circ\| < \epsilon\}$ で

$$f(\mathbf{x}^\circ) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in X \cap B_\epsilon(\mathbf{x}^\circ)$$

を満たす $\epsilon > 0$ が存在するときには，$\mathbf{x}^\circ$ も**局所最適解** (locally optimal solution) とよぶ．これと区別するため，(1.2) を満たす $\mathbf{x}^* \in X$ を**大域的最適解** (globally optimal solution) ともよぶ．一般に実行可能領域 X も複数の関数 $g_i : D \to \mathbb{R}^1$ $(i = 1, \ldots, m)$ によって

$$X = G \cap \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \leq 0,\ \ i = 1, \ldots, m\} \tag{1.3}$$

と表されるが，ここで G は $\mathbb{R}^n$ や整数ベクトル全体の集合 $\mathbb{Z}^n$ などである．

最適化問題は，以下に示す関数 f と集合 X の性質によって 2 つのクラスに大別され，それぞれに適用できるアルゴリズムも大きく異なる．

関数の凸性　関数 $f : D \to \mathbb{R}^1$ が任意の 2 点 $\mathbf{x}^1, \mathbf{x}^2 \in D$ に対して

$$f[(1-\lambda)\mathbf{x}^1 + \lambda\mathbf{x}^2] \leq (1-\lambda)f(\mathbf{x}^1) + \lambda f(\mathbf{x}^2), \quad \forall\lambda \in [0,\ 1]$$

を満たすとき，f は**凸関数** (convex function) であるという．

集合の凸性　集合 $X \subset \mathbb{R}^n$ が任意の 2 点 $\mathbf{x}^1, \mathbf{x}^2 \in X$ に対して

$$(1-\lambda)\mathbf{x}^1 + \lambda\mathbf{x}^2 \in X, \quad \forall\lambda \in [0,\ 1]$$

を満たすとき，X は**凸集合** (convex set) であるという．

問 1.1.　式 (1.3) において g_i がすべて凸関数で $G = \mathbb{R}^n$ であれば，X は凸集合となることを示せ．

1.1　凸計画問題

関数 f と集合 X がともに凸である最適化問題 (1.1) を**凸計画問題** (convex program) といい，そのアルゴリズムは数理計画法の中で最も発達している．凸計画問題は，さらに次の 2 つのクラスに細分される．

線形計画問題

関数 f が 1 次関数で，X を定義する g_i もすべて 1 次，さらに $G = \mathbb{R}^n$ である凸計画問題を**線形計画問題** (linear program, or LP) とよぶ．行列 $\mathbf{A} \in \mathbb{R}^{m\times n}$ とベクトル $\mathbf{b} \in \mathbb{R}^m$，$\mathbf{c} \in \mathbb{R}^n$ に対して $f(\mathbf{x}) = \mathbf{c}^\mathsf{T}\mathbf{x}$，$[g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})]^\mathsf{T} = \mathbf{A}\mathbf{x} - \mathbf{b}$ であれば，線形計画問題は

$$\left|\begin{array}{ll} \text{最小化} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{条　件} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{array}\right.$$

と表される．非常に大規模な場合（m, n が数万！）でも短時間のうちに効率よく解くことができるだけでなく，集合 X が**グラフ** (graph) によって特徴づけられる**ネットワーク流問題** (network flow problem) をはじめとする応用上重要な各種の問題を含んでいる．

LP 全般の参考書

田村・村松, 最適化法, 共立出版 (2002)
今野　浩, 線形計画法, 日科技連 (1987)
V. Chvátal (訳: 坂田・藤野), 線形計画法 (上), 啓学出版 (1986)
伊理正夫, 線形計画法, 共立出版 (1986)

ネットワーク流の参考書

藤重　悟, グラフ・ネットワーク・組合せ論, 共立出版 (2002)
V. Chvátal (訳: 坂田・藤野), 線形計画法 (下), 啓学出版 (1988)
伊理・藤重・大山, グラフ・ネットワーク・マトロイド, 産業図書 (1986)

1.2　(凸) 非線形計画問題

線形計画問題以外の凸計画問題の総称が (凸) **非線形計画問題** ((convex) nonlinear program) である．中でも，f が 2 次の凸関数で，線形計画問題と同じ X をもつ (凸) 2 次計画問題 ((convex) quadraftic program) は，資産運用や最小二乗法などに応用され，線形計画問題に次いで効率的に解くことができる．

非線形計画問題の参考書

田村・村松, 最適化法, 共立出版 (2002)
今野・山下, 非線形計画法, 日科技連 (1978)
福島雅夫, 非線形最適化の理論, 産業図書 (1980)

1.3　非凸計画問題

関数 f か集合 X のいずれか一方，あるいは両方が凸性を満たさない最適化問題 (1.1) を**非凸計画問題** (nonconvex program) とよぶ．関数 f と g_i が線形計画問題と同様にすべて 1 次であっても，$G = \mathbb{Z}^n$ である**整数計画問題** (integer program) はこのクラスに含まれ，巡回セールスマン問題に代表される**組合せ最適化** (combinatorial optimization) にも関連して応用上きわめて重要である．しかし，現在この分野の知識は凸計画問題のそれに比べてごくわずかであり，今後の数理計画法における大きな研究課題となっている．

組合せ最適化の参考書

今野・鈴木 他, 整数計画法と組合せ最適化, 日科技連 (1982)
茨木俊秀, 組合せ最適化 – 分枝限定法を中心として, 産業図書 (1983)
大山・久保, 巡回セールスマン問題への招待, 朝倉書店 (1997)
今野　浩, 整数計画法, 産業図書 (1981)

一般の非凸計画問題を扱った参考書は洋書に優れたものが多いが，残念ながら日本語で書かれたものは今のところ存在しない．

一般の非凸計画の参考書

R. Horst and H. Tuy, *Global Optimization (3rd ed.)*, Springer-Verlag (1995)
R. Horst, P.M. Pardalos and N.V. Thoai, *Instroduction to Global Optimization*, Kluwer Academic Publishers (1995)

1.4　数理計画問題全般

以上の他に，数理計画全般を扱ったものとして，

数理計画全般の参考書

G.L. Nemhausear, A.H.G. Rinnooy Kan and M.J. Todd (監訳: 伊理・刀根・今野), 最適化ハンドブック, 朝倉書店 (1995)
茨木・福島, 最適化の手法, 共立出版 (1993)
久野・繁野・後藤，数理最適化，オーム社 (2012)
刀根　薫, 数理計画法, 朝倉書店 (1978)
森　他, オペレーションズ・リサーチ I, 朝倉書店 (1991)

また，数理計画法の応用を扱ったものとしては，

数理計画応用の参考書

今野　浩, 数理決定法入門 – キャンパスのＯＲ, 朝倉書店 (1992)
今野　浩, 実践数理決定法, 日科技連 (1997)
H.P. Williams (監訳: 前田), 数理計画モデルの作成法, 産業図書 (1995)

この３冊は，厄介な問題に直面したとき，それを数理計画法によって解決するためのヒントを与えてくれるが，中でも「数理決定法入門」は，その副題通り，某工業大学の学生たちが自ら解決した (!) 身近な問題を例に数理計画法をはじめとする**オペレーションズ・リサーチ** (operations research, OR) の手法をわかりやすく解説した好著である．

日本語の入門書を中心に紹介したが，日本語にこだわらなければ，

数理計画の洋書

R. Ahuja, T.L. Magnanti and J.B. Orline, *Network Flows – Theory, Algorithms, and Applications*, Prentice Hall (1993)
M.S. Bazaraa, J.J. Jarvis and H.D. Sherali, *Linear Programming and Network Flows (2nd ed.)*, John Wiley and Sons (1990)
M.S. Bazaraa, H.D. Sherali and C.M. Shetty, *Nonlinar Programming – Theory and Algorithms (2nd ed.)*, John Wiley and Sons (1993)
C.H. Papadimitriou, *Combinatorial Optimization – Algorithms and complexity*, Prentice Hall (1982)

などの名著があり，どれも 1 冊読むだけでその分野の権威になることができる（!?）．

2 計算効率の評価

数理計画法のアルゴリズム開発において，最も注意の払われるのが計算効率の善し悪しである．通常，この計算効率の評価には次の 3 つの方法が用いられている．

経験的解析 (empirical analysis)**:** 問題例の分布を特定してサンプリングを行い，それらに対するアルゴリズムの振る舞いを実際に計測する．

最悪計算量解析 (worst-case analysis)**:** 任意の問題例に対し，アルゴリズムが要する計算量の上界を解析する．

平均計算量解析 (average-case analysis)**:** 問題例の分布に特定の仮定を設け，アルゴリズムが要する計算量の平均を解析する．

このうち，平均計算量は一般に解析が難しく，アルゴリズムの性能評価に広く用いられているのは経験的解析と最悪計算量解析である．ここでは後者について，組合せ最適化の汎用アルゴリズムである動的計画法 (dynamic programming) を 2 つの問題に適用しながら説明しよう．

2.1 ナップサック問題と最悪計算量

最悪計算量解析では，四則演算や代入，比較などの基本演算 1 回が単位時間で実行されるものと仮定し，問題サイズを表すいくつかのパラメータを用いてアルゴリズムの実行時間の上界を算定する．小学生の頃に，誰もが遠足の前夜に頭を悩ませた次の問題を例に考えてみよう．

最適化ナップサック

入力: 品物の集合 N, 各品物 $j \in N$ の大きさ $a_j \in \mathbb{Z}_+$，価値 $c_j \in Z_+$, およびナップサックの容積 $b \in \mathbb{Z}_+$

出力: $\sum_{j\in M} a_j \le b$ を満たし，価値の和 $\sum_{j\in M} c_j$ が最大となる N の部分集合 $M \subset N$

最適化ナップサックの問題サイズは，品物の数 $n = |N|$，その大きさ a_j，価値 c_j，ナップサックの容積 b によって定まり，$N = \{1, \ldots, n\}$ とすれば，次の整数計画問題に定式化できる：

$$\left|\begin{array}{ll} \text{最大化} & \displaystyle\sum_{j=1}^{n} c_j x_j \\ \text{条　件} & \displaystyle\sum_{j=1}^{n} a_j x_j \le b \\ & x_j \in \{0,\ 1\},\quad j = 1, \ldots, n. \end{array}\right. \tag{2.1}$$

これを動的計画法で解決するには,

$D^k(y)$: 品物を $\{1, \ldots, k\}$ に限定し，容積 y のナップサックに詰めるときの価値の和の最大値

と定めて，これを $n \times b$ の表に $(k, y) = (1, 1)$ から昇順に，$(1, 2), \ldots, (1, b); (2, 1), \ldots, (n, b)$ と埋めていけばよい．最適値は $D^n(b)$ で与えられるので，表が最後まで埋まれば (2.1) は解けたことになる．実際,

$$\begin{cases} D^0(y) = 0, & y \ge 0 \text{ 場合} \\ D^k(y) = -\infty, & y < 0 \text{ の場合} \end{cases}$$

と定義しておけば，漸化式

$$D^{k+1}(y) = \max\{D^k(y),\ c_{k+1} + D^k(y - a_{k+1})\},\quad k = 0, 1, \ldots, n-1 \tag{2.2}$$

が成り立つ (なぜか？) ので，表の成分を昇順に埋めることができる．

問 2.1. 漸化式 (2.2) が成り立つ理由を説明せよ．

さて，$D^k(y)$ が $y = 1, \ldots, b$ に対して計算してあれば，$D^{k+1}(y)$ の値は，引き算，足し算，比較，代入の 4 回の基本演算で求めることができる．これを $k = 0, 1, \ldots, n-1;\ y = 1, \ldots, b$ に対して行うので，表を埋めるのに必要な基本演算回数の上界は $4nb$ であることがわかる．使用する計算機の違いなどから，この上界は必ずしも実際の計算時間を厳密に予測するものではないが，最悪の場合に必要となる計算時間が nb に比例することは概ね検討がつく．次の概念は，このこ

とを正確に表すのに用いられる:

$$
\begin{aligned}
O(f(n)) &= \{g(n) \mid \exists c, n_0 > 0,\ \forall n > n_0,\ g(n) \leq cf(n)\} \\
\Omega(f(n)) &= \{g(n) \mid \exists c, n_0 > 0,\ \forall n > n_0,\ g(n) \geq cf(n)\} \\
\Theta(f(n)) &= O(f(n)) \cap \Omega(f(n)) \\
o(f(n)) &= \{g(n) \mid \forall c > 0,\ \exists n_0 > 0,\ \forall n > n_0,\ g(n) \leq cf(n)\} \\
\omega(f(n)) &= \{g(n) \mid \forall c > 0,\ \exists n_0 > 0,\ \forall n > n_0,\ g(n) \geq cf(n)\}
\end{aligned}
$$

普通，例えば $g(n) \in O(f(n))$ である場合，$g(n) = O(f(n))$ と書く．また，$g(n) = \Theta(f(n))$ のときには，$g(n)$ と $f(n)$ は**同じオーダ**，あるいは**漸近的に同じ**であるといい，$g(n) = o(f(n))$ ならば，$f(n)$ は $g(n)$ よりも**オーダが高い**，あるいは $f(n)$ は $g(n)$ よりも**漸近的に早く増加する**という．この概念にしたがえば，最適化ナップサックを解くのに $4nb$ 回の基本演算を必要とする動的計画法だが，その実行時間は $O(nb)$ と書かれることになる．

ナップサック問題から最長路問題へ

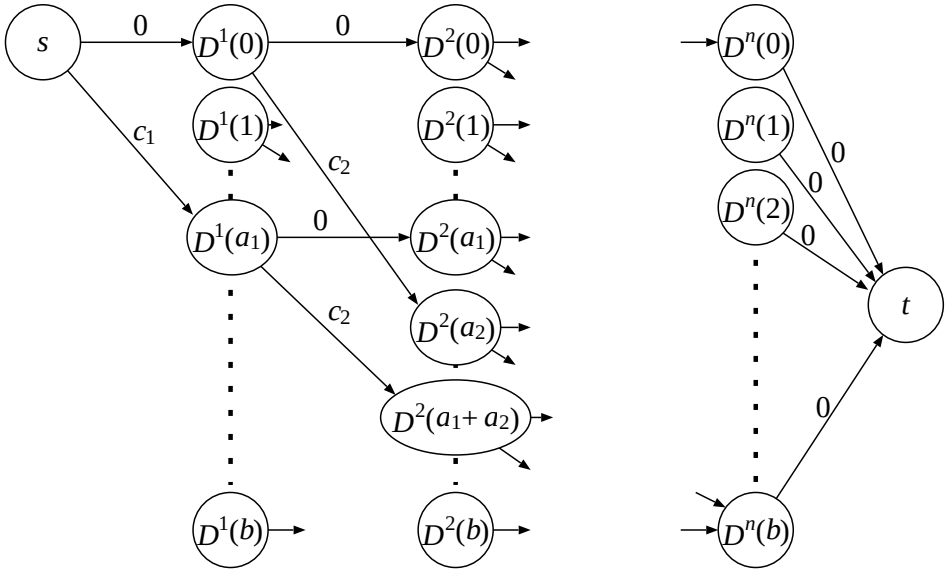

図 2.1: ナップサック問題のネットワーク．

最適化ナップサックは，一見するとネットワーク上の問題には思えないが，$n(b+1)+2$ 個の**頂点** (node) をもつ**非巡回的な** (acyclic) ネットワーク流問題に帰着させることができる（図 2.1）．各頂点 $D^{k+1}(y)$ へは，2 つの**有向枝** (directed arcs) を頂点 $D^k(y)$ と $D^k(y - a_{k+1})$ から接続させ，前者の長さをゼロ，後者の長さを c_{k+1} と定める．次に始点 s を与え，そこから頂点 $D^1(0)$ へ長さゼロの，また頂点 $D^1(a_1)$ へは長さ c_1 の有向枝を延ばす．そうすれば，始点 s から頂点 $D^k(y)$

への各**有向路** (directed path) は，容積 y のナップサックに入る品物 $\{1,\ldots,k\}$ の部分集合に対応し，有向路に含まれる枝の長さの和，つまり有向路の長さが部分集合の価値となる．最後に，終点 t を与えて，頂点 $D^n(0)$, $D^n(1)$, …, $D^n(b)$ から長さゼロの有向枝で結ぶ．始点 s から終点 t への有向路はいずれも，容積 b のナップサックに収まる品物 $N=\{1,\ldots,n\}$ の部分集合に対応するので，その中で長さの最大のものがナップサック問題の最適解を与えてくれる．

問 2.2. 任意の非巡回的な有向グラフ $G=(V,E)$ に対して各枝 $(i,j)\in E$ の長さ c_{ij} が与えられたとき，G に含まれる有向路の中で最も長いもの（最長路）を求める動的計画法を提案せよ．

2.2 最短路問題と動的計画法

ナップサック問題はネットワーク流問題に帰着することがわかったところで，もっと有名なネットワーク流問題についても考えてみよう．

最短路

入力: 有向グラフ $G=(V,E)$，各枝 $(i,j)\in E$ の長さ $c_{ij}\in\mathbb{Z}$，および始点 $s\in V$

出力: 始点 s から各頂点 $j\in V$ への有向路の中で長さ $\sum_{(i,k)\in P_j} c_{ik}$ が最小の有向路 $P_j\subset E$

最短路問題は，有向グラフ G の頂点の数 $n=|V|$，枝の本数 $m=|E|$，その長さ c_{ij} によってサイズが定まり，線形計画問題として定式化できる：

$$\left|\begin{array}{ll} \text{最小化} & \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} \\ \text{条　件} & \displaystyle\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = \begin{cases} n-1, & i=s \\ -1, & \forall i\in V\setminus\{s\} \end{cases} \\ & x_{ij}\geq 0, \quad \forall (i,j)\in E. \end{array}\right. \tag{2.3}$$

したがって，**シンプレックス法** (simplex algorithm) や**内点法** (interior-point algorithm) などの線形計画法のアルゴリズムでも解くことはできる．しかし，注意したいのは枝の長さ c_{ij} が負値を取ることも許している点で，このために最短路問題専用のアルゴリズムである **Dijkstra 法** (Dijkstra's algorithm) では必ずしも正しい答が得られない．ここで紹介する**ラベル修正法** (label-correcting algorithm) は，$c_{ij}<0$ となる枝 (i,j) が G に存在しても，(2.3) に最適解があれば，それを正しく生成することができる．以下では，説明を簡単にするために次の仮定をおく：

仮定 2.1. グラフ G には，始点 s から他のすべての頂点 $j\in V\setminus\{s\}$ への有向路が存在する．

もしも s から j への有向路が存在しなければ，新たに枝 (s,j) を G に追加し，その長さを $c_{sj}=+\infty$ と定めればよい．したがって，この仮定によって問題の一般性が失われることはない．

最短路最適性条件

始点 s から各頂点 $j \in V$ への有向路の長さ，つまり j への距離を $d(j)$ で表すことにしよう．始点 s への距離は $d(s) = 0$ と定める．このとき，以下の定理が成り立つ．

定理 2.1.　すべての頂点 $j \in V$ の距離 $d(j)$ が始点 s から j への最短距離に等しくなる必要十分条件は，

$$d(j) \leq d(i) + c_{ij}, \quad \forall (i,j) \in E. \tag{2.4}$$

証明:　条件 (2.4) の必要性から示そう．距離 $d(i), d(j)$ を与える有向路をそれぞれ P_i, P_j とする．もし，

$$d(j) > d(i) + c_{ij}$$

となる枝 (i,j) があれば，頂点 s から P_i を経由し，(i,j) を通って頂点 j へ向かう方が P_j よりも距離が短い．これは $d(j)$ が最短距離であることに矛盾．

次に十分性．頂点 s から j への任意の有向路を $P = (s = i_1, i_2, \ldots, i_h = j)$ とする．条件 (2.4) から，

$$\begin{aligned} d(i_2) &\leq d(i_1) + c_{i_1 i_2} \\ d(i_3) &\leq d(i_2) + c_{i_2 i_3} \\ &\vdots \\ d(i_h) &\leq d(i_{h-1}) + c_{i_{h-1} i_h} \end{aligned}$$

頂点 s のラベルが $d(s) = 0$ であることに注意して，これらを辺々足しあわせれば

$$d(j) = d(i_h) \leq \sum_{h=2}^{h} c_{i_{h-1} i_h} = \sum_{(i,k) \in P} c_{ik}$$

が得られる．右辺は有向路 P の長さであり，P の任意性から長さ $d(j)$ を与える有向路が最短であることが示された． ■

頂点 j が始点 s でなければ，j への有向路は j の前に必ず別の頂点 $i \neq j$ を通るはずである．したがって，条件 (2.4) は，少なくとも 1 本の枝 $(i,j) \in E$ で等号が成り立つことに注意しよう．このことから，各頂点 $j \in V$ への最短距離は，**Bellman の方程式** (Bellman's equations) として知られる

$$\left.\begin{aligned} &d(s) = 0 \\ &d(j) = \min\{d(i) + c_{ij} \mid (i,j) \in E\}, \quad j \in V \setminus \{s\} \end{aligned}\right\} \tag{2.5}$$

を解けば得られることがわかる．また，ネットワークに負の長さの有向閉路が存在する場合，条件 (2.4) を満たす距離 $d(i)$ はない（なぜか？）ことにも注意しよう．

問 2.3. 長さ $\sum_{(i,j)\in Q} c_{ij} < 0$ の有向閉路 $Q = (i_1, i_2, \ldots, i_h = i_1)$ が G に存在するとき，条件 (2.4) を満たす距離 $d(i)$ のないことを示せ．

ラベル修正法

Bellman の方程式 (2.5) を解く一つの方法は逐次近似である．始点 s から頂点 $j \in V$ への最短距離の近似値には

$$D^k(j) : \text{始点 } s \text{ から高々 } k \text{ 本の枝を通り，} j \text{ へ向かう有向路の中で最短なものの長さ}$$

を用い，これを頂点 j の**ラベル** (label) とよぶ．初期ラベルは

$$\left.\begin{aligned} &D^1(s) = 0 \\ &D^1(j) = c_{sj}, \quad j \in V \setminus \{s\} \end{aligned}\right\} \tag{2.6}$$

で与えられるが，$(s, j) \notin E$ であれば $c_{sj} = +\infty$ とする．各頂点 $j \in V \setminus \{s\}$ のラベルを漸化式：

$$D^{k+1}(j) = \min\{D^k(j),\ \min\{D^k(i) + c_{ij} \mid (i, j) \in E\}\}, \quad k = 1, 2, \ldots \tag{2.7}$$

で更新していくと，左辺は右辺の最小値なので

$$D^1(j) \geq D^2(j) \geq \cdots \geq D^k(j) \geq \cdots$$

が成立する．始点 s から j への最短路があれば，高々 $n = |V|$ 個の頂点しか通らないはずなので，そこに含まれる枝の本数も最大で $n - 1$ であり，初期ラベルから $n - 2$ 回目の更新で得られるラベル $D^{n-1}(j)$ が Bellman の方程式 (2.5) の解 $d(j)$ となる．

問 2.4. 漸化式 (2.7) が成り立つ理由を説明せよ．

ナップサック問題のときと同じように，ラベル $D^k(j)$ で $(n-1) \times n$ の表を $(k, j) = (1, 1)$ から昇順に，$(1, 2), \ldots, (1, n)$; $(2, 1), \ldots, (n-1, n)$ まで埋めていってもよいが．同様の操作をもう少しスマートに行なうアルゴリズムをここで紹介しよう：

algorithm LABEL_CORRECTING
begin
  $D(s) := 0$; $\mathrm{prev}(s) := 0$; e_h $(h = 1, \ldots, m = |E|)$ で枝を表す;
  for 各 $j \in V \setminus \{s\}$ **do begin**
    $D(j) := c_{sj}$; $\mathrm{prev}(j) := s$;

end;
repeat
for $h = 1$ **to** m **do begin**
if 枝 $e_h = (i, j)$ に対して $D(j) > D(i) + c_{ij}$ **then**
$D(j) := D(i) + c_{ij}$ と $\mathrm{prev}(j) := i$ を更新する
end
until 条件 (2.4) が満たされる
end;

ラベル $D^k(j)$ は $D(j)$ で表し，長さ $D(j)$ の有向路が頂点 j の前に通る頂点を $\mathrm{prev}(j)$ に格納している．上記の説明と問 2.4 から次の命題は明らかであろう．

定理 2.2. 長さが負の有向閉路が存在しなければ，LABEL_CORRECTING は高々$n-2$ 回の反復ののち，始点 $s \in V$ から他のすべての頂点 $j \in V \setminus \{s\}$ への最短路を与える．

最悪計算量の解析

アルゴリズムの各反復で枝集合 $E = \{e_1, \ldots, e_n\}$ の走査に $O(m)$ 回の演算が行われるので，その最悪計算量は，定理 2.2 から $O(nm)$ となる．また，この定理から $n-1$ 回以上の反復が行われる場合には，ネットワークに負の有向閉路が存在することもわかる．つまり，$n-2$ 回の反復ののちも最短路最適性条件 (2.4) が満たされず，$n-1$ 回めの反復で頂点 j のラベル $D(j)$ が更新されたとすれば，$D(j)$ には出発点 s から j への有向路で n 本以上の枝を含むものの長さが記されていることになる．こうした状況は，ネットワークに負の長さの有向閉路が存在しない限り起こりえない．

ラベル修正法の長所は，各反復で行われる枝集合 E の走査の柔軟性にある．ここで紹介した LABEL_CORRECTING では始めに枝の順番を定め，毎回その順番にしたがって走査を行っているが，実は走査の順番を定めなくとも，負の有向閉路が存在しなければ有限回での収束が保証される．例えば，**repeat** から **until** までを

while $D(j) > D(i) + c_{ij}$ を満たす枝 (i, j) が存在 **do**
$D(j) := D(i) + c_{ij}$ と $\mathrm{prev}(j) := i$ を更新する;

と書き換えて枝集合の走査を全くランダムに行っても，$D(j)$ には上界値 $\sum_{(i,h)\in E} |c_{ih}|$ がら下界値 $-\sum_{i,h)\in E} |c_{ih}|$ までの高々$2\sum_{(i,h)\in E} |c_{ih}|$ 回の更新が行われるだけである．したがって，$C = \max\{|c_{ij}| \mid (i, j) \in E\}$ とおけば，反復回数は $O(nC)$ におさえられ，最悪計算量は $O(n^2C)$ となることがわかる．