平成 13 年度

学士学位論文

一対比較グラフの頂点の順位づけ問題

A problem of ranking the vertices of a paired comparison digraph

1020278　　亀本　学

指導教員　　坂本 明雄

2002 年 2 月 8 日

高知工科大学 情報システム工学科

要 旨

一対比較グラフの頂点の順位づけ問題

亀本 学

一対比較グラフとは重みつき有向グラフであって，異なる二頂点間に有向枝があればその重みの和が 1 となるものである．本論文では，一対比較グラフの頂点に順位をつける問題について考察する．n 個の頂点を持つ一対比較グラフを D とする．D の頂点集合から集合 $\{1, 2, \cdots, n\}$ の上への一対一の写像を D の順位づけという．このとき，どのような性質を満たす順位づけが最も望ましいかについては意見の分かれるところである．

本論文ではひとつの順位づけに対して，その逆枝および逆枝の長さの総和を定め，D のすべての順位づけの中で逆枝の長さの和が最小となるものを最適順位づけと定義した．特殊な一対比較グラフ (例えば，完全一対比較グラフ，準完全一対比較グラフ等) においては最適順位づけを容易に求めることができ，しかもその結果は我々の日常経験と矛盾しないものであることが示される．

キーワード　一対比較グラフ，逆枝，最適順位づけ

Abstract

A problem of ranking the vertices of a paired comparison digraph

Manabu KAMEMOTO

A paired comparison digraph (PCD) is a weighted digraph in which the sum of the weights of arcs, if any, joining two distinct vertices is exactly one; otherwise, there exist no arcs joining them. We consider a problem of ranking the vertices of a PCD. Let D be a PCD with n vertices. A ranking of D is a one-to-one mapping from the set of vertices of D onto the set of integers $\{1, 2, \cdots, n\}$. There are many different opinions on a definition of the most desirable ranking.

In this paper, we define the set of backward arcs of a ranking and its total length. Then, the optimal ranking is a ranking whose total length is minimum among those of all possible rankings. It is easy to ask for optimal rankings of a certain class of PCDs such as complete PCD and semicomplete PCD. Furthermore, the results are very reasonable.

key words paired comparison digraph, backward arcs, optimal rankings

目次

図目次

第 1 章

序論

いくつかの対象に順位をつけようとするとき，全体の中で各対象の絶対的な優劣の程度を数値的に得ることは困難である．一方，二つの対象間の相対的な優劣の判定は比較的容易に得られる場合も少なくない．しかし，二つの対象間の優劣の程度がたとえ数値で与えられたとしても，それは相対的なものであって，全対象の中での順位の判定にどのように利用するかについては種々の考え方がある．例えば，いくつかのチームがリーグ戦を行い，その戦績から順位をつける場合を考えてみる．

(i) サッカー： 各試合ごとに引き分けがあるので勝ち点を定め，全試合についての勝ち点の合計が多い順に順位がつく．

(ii) プロ野球： 各対戦チームと定められた回数の試合を行なうが，全試合についての勝率あるいは勝利数で順位がつく．

二つの対象間の優劣の判定を一対比較と呼ぶ．この一対比較を用いて全対象の中での順位をつけるというのが本研究の目的である．

例えば，ある対象物として野球の 3 チーム a, b, c が野球の試合をそれぞれ 3 回戦ずつ行なって順位をつける場合を考えてみる．全勝のチームがあればそのチームが 1 位であり，全敗のチームがあればそのチームが 3 位であることは明らかである．しかしながら，3 回戦の結果が 2 勝 1 敗となることはしばしば起こる．全試合を終えての順位は a が 1 位で b が 2 位で c が 3 位であり，a は b に対して 2 勝 1 敗であったとする．このとき，a が得た 2 勝という値は a の順位が b の順位より小さいという結果に反映されているが，b が得た 1 勝という値はこの順位づけには矛盾するものである．この矛盾する値の総和が最小な順位づけを最適な順位づけとするというのが本研究での考え方である．

一対比較によって二つの対象間の相対的な優劣の判定が得られたとき，各対象を頂点で表し，一対比較の結果を頂点間の有向枝で表現することで，対象に順位をつける問題は一対比較グラフの頂点の順位づけの問題として扱うことが出来る．本研究ではこのような一対比較グラフの最適な頂点の順位づけを求める方法を考察する．

2 章以降の概要を以下に示す．

まず最初に，2 章で有向グラフや一対比較グラフに関する基本的な定義と記号について示す．次に 3 章では，本研究の主題である一対比較グラフの頂点の最適な順位づけについて詳しく述べる．4 章では，特殊な一対比較グラフ (完全一対比較グラフ，準完全一対比較グラフなど) の最適順位づけを簡単に求める方法について述べる．そして，5 章では最適順位づけの性質について考察する．4 章と 5 章では特殊な場合の最適順位づけに関するいくつかの定理とその証明が主となる．次に，6 章では一般のグラフの場合の最適順位づけについても考察する．最後に，7 章では本研究についてのまとめ，そしてこの問題をさらに発展させる上で残された課題について述べることとする．

第 2 章

定義と記号

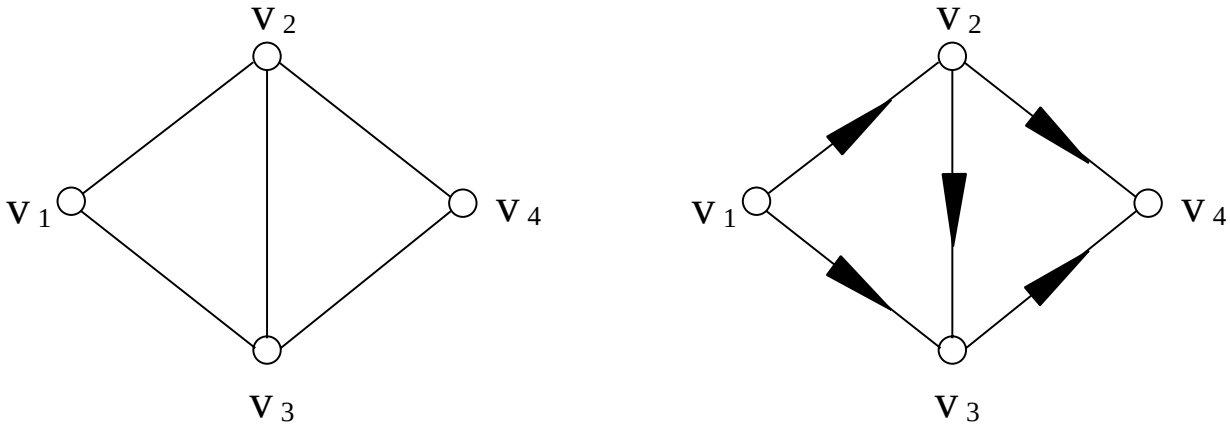
本章では，一対比較グラフの頂点の最適な順位づけ問題を定義し，その性質を議論する上で必要となるいくつかの定義や記号について述べる．

2.1　グラフの定義

グラフは $G = (V, E)$ と表される．ここで，V を頂点の集合，E を枝の集合という．ちなみに G は graph の頭文字である．また，特に断らない限り，議論の対象となっているグラフの頂点の個数を n で表す．V は有限個の要素の集合であり，E は直積集合 $V \times V$ の部分集合である．たとえば頂点集合 $V = \{v_1, v_2, v_3, v_4\}$，枝集合 $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4), (v_3, v_4)\}$ からなるグラフ $G = (V, E)$ を次の図 2.1 の (a) に示す．

2.2　有向グラフの定義

枝表示 (v_i, v_j) を順序対であると考えるとき，グラフの全ての枝 (v_i, v_j) は向きをもつことになり，そのようなグラフを有向グラフという．枝表示 (v_i, v_j) が順序対であるというのは，v_i と v_j の順序を考慮するということである．(v_i, v_j) は v_i から v_j という方向の順序を表しており，同様に (v_j, v_i) は v_j から v_i という方向の順序を表している．また，有向グラフの枝を有向枝という．図 2.1 の (b) に示したグラフ $D = (V, E)$ は有向グラフである．ちなみに D は directed graph (略して digraph) の頭文字である．なお以下では，有向枝 (v_i, v_j) を単に $v_i v_j$ と表す．



(a)グラフ G = (V,E)　　(b)有向グラフ D = (V,E)

図 2.1　グラフと有向グラフ

2.3　単純な有向グラフの定義

単純な有向グラフとは,

- 自己ループを含まない
- 同方向の並列枝を含まない

という条件を満たしたグラフである．次の図 2.2 は頂点 v_2 に自己ループを，頂点 v_1 と頂点 v_3 の間に同方向の並列枝を含んでいるので，有向グラフではあるが単純な有向グラフではない.

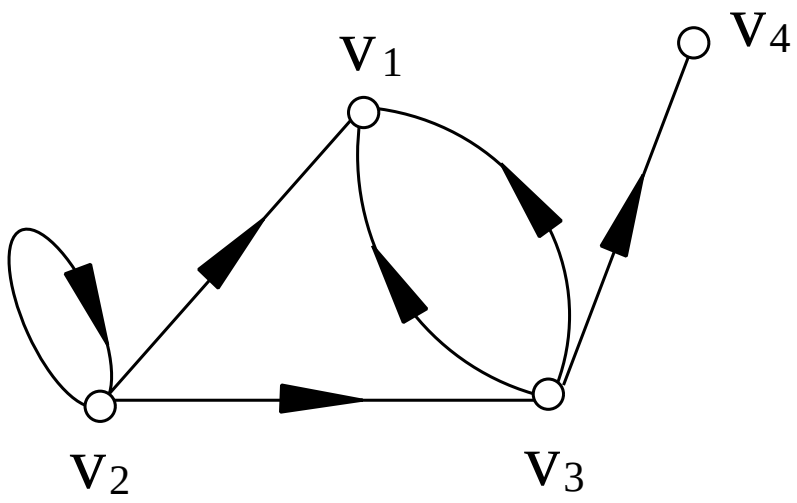


図 2.2　単純な有向グラフ

2.4　一対比較グラフの定義

一対比較グラフとは，すべての有向枝 vw に次の 3 つの条件を満たす重み $\epsilon(vw)$ が与えられている重み付き有向グラフである．

(1) 有向枝 vw の重みは $0 < \epsilon(vw) \leq 1$ の範囲にある．

(2) vw と wv が共に有向枝であるとき，$\epsilon(vw) + \epsilon(wv) = 1$ である．

(3) vw は有向枝だが wv は有向枝ではないとき，$\epsilon(vw) = 1$ である．

2.5　有向グラフの一対比較グラフ化

重みが定められていない単純な有向グラフ D の各有向枝の重みを次のように定めると，D は一対比較グラフであると考えることが出来る．

(1) vw と wv が共に有向枝ならば，$\epsilon(vw) = \epsilon(wv) = 0.5$ とする．

(2) vw は有向枝だが wv は有向枝ではないとき，$\epsilon(vw) = 1$ とする．

この論文では有向グラフを上のように解釈して，一対比較グラフであるとみなす．特に，全ての二頂点間においていずれかの方向に一本の有向枝が存在しているようなトーナメントは一対比較グラフである．

また，次の図 2.3 は有向グラフの一対比較グラフ化の例である．

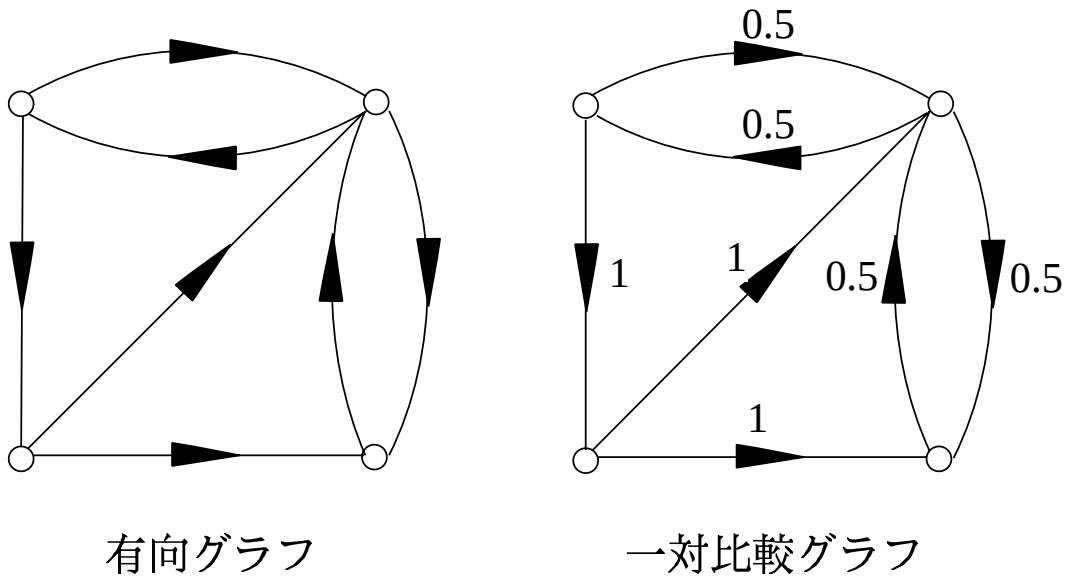


図 2.3　有向グラフの一対比較グラフ化

2.6　記号の定義

X と Y を有限集合とする．X の要素数を $|X|$ と表示する．X と Y が互いに素であるとき，すなわち $X \cap Y = \emptyset$ であるとき，$X \cup Y$ を $X + Y$ と表示する．有向グラフは全ての頂点間でたかだか一本の有向枝しか存在しないとき非対称であるといわれ，全ての頂点間に有向枝が少なくとも一本存在するとき完全グラフといわれる．トーナメントは上の二つのグラフの特徴を共に満たしているので完全非対称グラフである．

有向グラフ D において，頂点 v の出次数，入次数，および次数をそれぞれ $d^+(v)$，$d^-(v)$，および $d(v)$ と表す．ここで，出次数や入次数は頂点から出ているあるいは頂点に入ってく

る枝の本数であり，$d(v) = d^+(v) + d^-(v)$ である．さらに必要に応じて，有向グラフの記号，例えば D を関数の添字につけることにする．例えば $d^+_D(v)$ と書く．

頂点数 n の一対比較グラフを $D = (V, E)$ とする．五つの関数 $\bar{\epsilon} : V \times V \to [0, 1]$; $\mu : V \times V \to \{0, 1\}$; σ^+,$\sigma^- : V \to [0, n-1]$; $d^* : V \to \{0, 1, 2, \cdots, n-1\}$ を次のように定義する．ただし, $[0, 1]$ は 0 以上 1 以下の実数の集合を表し, $\{0, 1\}$ は, 0 と 1 の集合である．

$$\bar{\epsilon}(vw) = \begin{cases} \epsilon(vw) > 0 & vw \text{ が } D \text{ の有向枝のとき,} \\ 0 & \text{それ以外のとき} \end{cases}.$$

$$\mu(vw) = \mu(wv) = \bar{\epsilon}(vw) + \bar{\epsilon}(wv),$$

$$\sigma^+(v) = \sum_{x \in V} \bar{\epsilon}(vx), \sigma^-(v) = \sum_{x \in V} \bar{\epsilon}(xv),$$

$$d^*(v) = n - 1 - (\sigma^+(v) + \sigma^-(v)) = n - 1 - \textstyle\sum_{x \in V} \mu(vx).$$

v と w が比較されているとき，$\mu(vw) = 1$ であり，そうでなければ $\mu(vw) = 0$ である．$d^*(v)$ は v と比較されていない頂点の数である．v の一般化したスコアをとして $\sigma^+(v)$ を考えて，$\sigma^+(v)$ を v の正のスコア, $\sigma^-(v)$ を v の負のスコアと呼ぶ．D が非対称有向グラフであるとき，v の正および負のスコアはそれぞれ v の出次数，入次数となることに注意する．

例えば次の図 2.4 において，v_3 の正のスコアは $\sigma^+(v_3) = 2.8$ で，負のスコアは $\sigma^-(v_3) = 1.2$ である．それに対して，出次数 $d^+(v_3) = 4$ で，$d^-(v_3) = 2$ となる．つまり，正や負のスコアは頂点から出ているあるいは頂点に入ってくる枝の重みの和である．

また，同じ図 2.4 において，$\epsilon(v_4v_3) = 0.4$ であるが，$\epsilon(v_1v_4) = 0$ は定義されない．なぜなら，一対比較グラフにおいて重み ϵ は有向枝に対してのみ定義されているからである．そのため，枝が無い場合は重みを 0 とし，全ての頂点対について定義される関数を $\bar{\epsilon}$ として新たに定めている．よって，$\bar{\epsilon}(v_1v_4) = 0$ であり，同時に $\bar{\epsilon}(v_4v_1) = 0$ でもある．

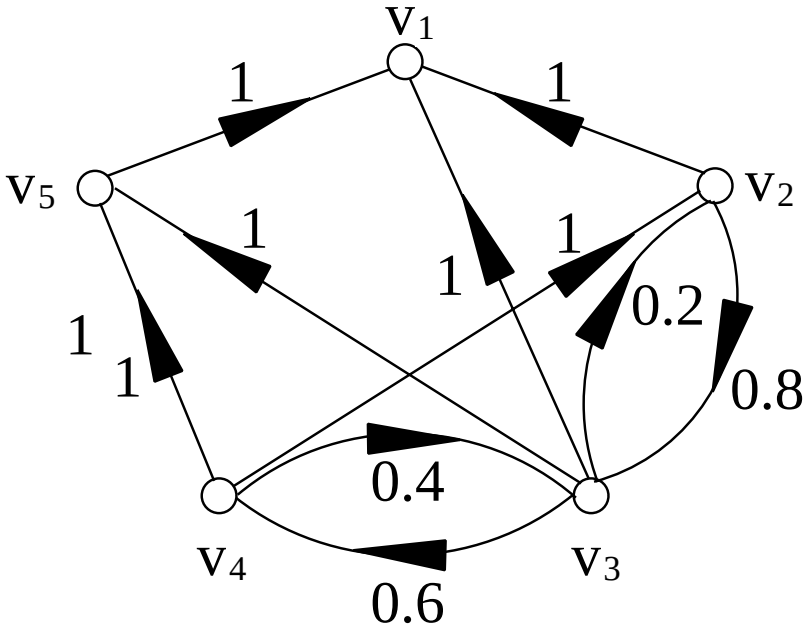


図 2.4　ある一対比較グラフ

2.7　一対比較グラフの解釈

一対比較グラフにはいくつかの解釈が可能と思われるが，もっともわかりやすい解釈は，その名の示すように一対比較の結果をグラフ化したものであるという解釈である．

もし，v が w より優れていれば (v が w に勝てば) $0.5 < \epsilon(vw) \leq 1$ という重みをもつ枝 vw を加える．このとき，$\epsilon(vw) = 1$ であれば枝 wv は存在しないが，$0.5 < \epsilon(vw) < 1$ であれば $\epsilon(wv) = 1 - \epsilon(vw)$ という重みをもつ枝 wv を同時に加える．

もし，v と w が対等 (引き分け) であれば 0.5 という重みをもつ 2 つの枝 vw と wv を加える．

もし，v と w が未比較 (未試合) であれば v と w の間には枝を加えない．そして，一般に枝の重み $\epsilon(vw)$ は，v の w に対する勝率 (v が w より優れているとみなされる割合) を表しているとみればよい．これは野球などに例えると分かり易い．重み $\epsilon(vw) = 0.56$ であるなら，それはチーム v がチーム w に対して 5 割 6 分の割合で勝利していることを表し，これ

は同時にチーム w がチーム v に対して 4 割 4 分の割合でしか勝利していないことも表している．

またここで，図 2.5 を用いて上記のことを次のようなモデルとして考えることができる．4 選手 a, b, c, d がある格闘技の試合をしたとする．c は a に KO 負けしたが，b には KO 勝ちした．一方，b と a の試合は 9 対 1 で b の判定勝ちであり，a は d に KO 勝ちして，d は c に 8 対 2 で判定勝ちした．b と d はまだ試合を行なっていない．

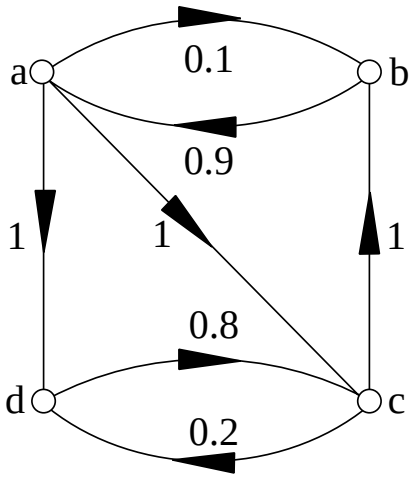


図 2.5　一対比較グラフのモデル

第 3 章

最適順位づけ

本章では，主題でもある最適な頂点の順位づけの概念について議論する．この章での議論を準用すれば，最適な順位づけとして，逆枝最適順位づけ，順枝最適順位づけ，相互最適順位づけなどを考えることが出来る．しかしながら，本論文では逆枝最適順位づけを最適な順位づけとして選んだ．よって，逆枝最適順位づけを今後本論文では単に最適順位づけと呼ぶことにする．

3.1 D の頂点の順位づけ α

n 個の頂点を持つ一対比較グラフを D とする．D の頂点の順位づけ α は D の頂点集合から整数集合 $\{1, 2, \cdots, n\}$ への一対一の対応 (全単射) である．また，D の順位づけ α が整数 $i(1 \leq i \leq n)$ に対して $\alpha(v_i) = i$ であるとき, $\alpha = [v_1, v_2, \cdots, v_n]$ と書く．さらに, 順位づけが α であるとき, 頂点 v の像 $\alpha(v)$ は α による v の順位であるという．また, 整数 $k(1 \leq k \leq n)$ に対し $\alpha^{-1}(k)$ は順位 k の頂点を表す．例えば，$\alpha(a) = 1, \alpha(b) = 4, \alpha(c) = 3, \alpha(d) = 2$ であるとき，$\alpha^{-1}(1) = a, \alpha^{-1}(2) = d, \alpha^{-1}(3) = c, \alpha^{-1}(4) = b$ を表す．さらに，D のすべての順位づけの集合を $\mathrm{R}(D)$ と書く．また，明らかに $|\mathrm{R}(D)| = n!$ となる．

ここで D の頂点 v に対し，D の最適順位づけにおける v の順位の平均値を v の順位値と言う．

3.2　逆枝の集合と長さ

$\alpha(v) < \alpha(w)$ である有向枝 vw は順位づけ α の順枝であるという．逆に，$\alpha(w) < \alpha(v)$ である有向枝 vw は α の逆枝であるという．逆枝は，順位づけ α と矛盾する方向の有向枝である．一対比較グラフ D の順位づけ α に対し，

$$B_D(\alpha) = \{vw \in E | \alpha(w) < \alpha(v)\}$$

という枝集合を α の逆枝集合という．

図 3.1 の (c) を使って具体的に説明する．ここでは順位づけに従って左から頂点を一直線上に並べて一対比較グラフを描いてある．よって a が 1 位で，b が 2 位で，c が 3 位で，d が 4 位である．矢印が右向きの枝は順枝であり，左向きの枝が逆枝となる．つまり図 3.1 の (c) において逆枝は dc, cb, ba の 3 本だけなので，この順位づけの逆枝集合は $B_D(\alpha) = \{dc, cb, ba\}$ である．

次に，順位づけ α の逆枝の長さを $\epsilon(vw)\{\alpha(v) - \alpha(w)\}$ と定め，α の逆枝の長さの総和を $\parallel B_D(\alpha) \parallel$ と表示する．すなわち $\parallel B_D(\alpha) \parallel$ は次式で表される．

$$\parallel B_D(\alpha) \parallel = \sum_{vw \in B_D(\alpha)} \epsilon(vw)\{\alpha(v) - \alpha(w)\}.$$

3.3　最適順位づけ

全ての順位づけの中で逆枝の長さの総和が最小となる順位づけを最適順位づけという．また，最適順位づけは 1 つであるとは限らず複数個ある場合もあるため，最適順位づけの集合を $\mathrm{OR}(D)$ と書く．

ここで，D の最適順位づけの逆枝の長さの総和を $l(D)$ と表記する．このとき，

$$l(D) = \min_{\alpha}\{\parallel B(\alpha) \parallel\}\ ,\ \mathrm{OR}(D) = \{\alpha \in \mathrm{R}(D) | \parallel B(\alpha) \parallel = l(D)\}$$

である．

H が $V(H) = V(D)$ である D の部分有向グラフであるとき，全ての α について

$B_H(\alpha) \subseteq B_D(\alpha)$ であるから $l(H) \leq l(D)$ であることに注意する．

ここで，例を用いてこの最適順位づけを説明する．図 3.1(a) の一対比較グラフに対して，順位づけが分かり易いように図 (a) を図 (b) と図 (c) のように書き直した．(b) の順位づけにおいて，順位づけは左から昇順で頂点 a が 1 位，c が 2 位，b が 3 位，d が 4 位であり，これを $[a, c, b, d]$ と表す．このとき，(b) の逆枝の長さの総和は $0.9 \times 2 + 0.8 \times 2 = 3.4$ であり，(c) の逆枝の長さの総和は $0.9 \times 1 + 1 \times 1 + 0.8 \times 1 = 2.7$ となる．なお，(c) は D の最適順位づけであり，$l(D) = 2.7$ となる．

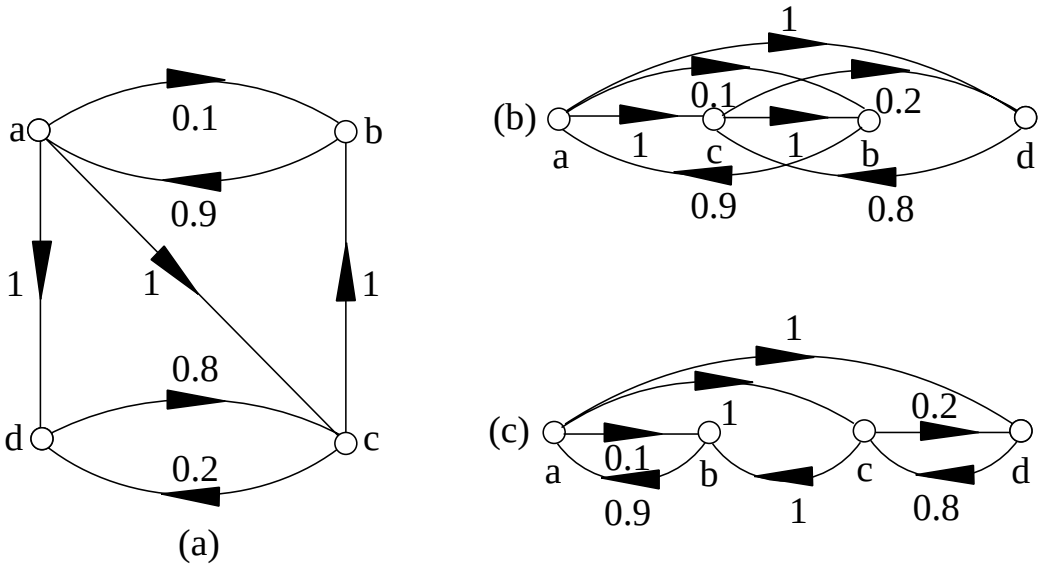


図 3.1　一対比較グラフと順位づけの例

一対比較グラフである D の順位づけを α とする．m，k を $1 \leq k < k + m \leq n$ である整数とする．ここで，n は D の頂点数である．このとき順位づけ α_m^k を次のように定義する．

$$\alpha_m^k(v) = \begin{cases} k+m & v = \alpha^{-1}(k) \text{ のとき}, \\ k & v = \alpha^{-1}(k+m) \text{ のとき}, \\ \alpha(v) & \text{それ以外のとき} \end{cases}$$

[補題 3.1] 頂点数 n の一対比較グラフの順位づけを α とする．$\alpha(v) = k$ および $\alpha(w) = k + m$ であるとき，

$$\begin{aligned}\parallel B(\alpha_m^k) \parallel - \parallel B(\alpha) \parallel &= m\Bigg(\sigma^+(v) - \sigma^+(w) + \sum_{\alpha(z)>k+m} \{\mu(wz) - \mu(vz)\}\Bigg) \\ &\quad + \sum_{k<\alpha(y)<k+m} \{\alpha(y) - k\}\{\mu(wy) - \mu(vy)\} \\ &= m\Bigg(\sigma^-(w) - \sigma^-(v) + \sum_{\alpha(x)<k} \{\mu(vx) - \mu(wx)\}\Bigg) \\ &\quad + \sum_{k<\alpha(y)<k+m} \{(k+m) - \alpha(y)\}\{\mu(vy) - \mu(wy)\}.\end{aligned}$$

(証明) 定義から，α の逆枝の長さは次のように表される．

$$\parallel B(\alpha) \parallel = \sum_{\alpha(x)<\alpha(y)} \bar{\epsilon}(yx)\{\alpha(y) - \alpha(x)\}$$

ここで $\sum$ は $\alpha(x) < \alpha(y)$ である頂点 x と y の全ての対についての総和を求めるものとする．$X = \{x \in V | \alpha(x) < k\}, Y = \{y \in V | k < \alpha(y) < k+m\}, Z = \{z \in V | k+m < \alpha(z)\}$, $B = \{st \in B(\alpha) | \{s,t\} \cap \{v,w\} = \emptyset\}$ とする．このとき, 次のように表される．

$$\begin{aligned}\parallel B(\alpha) \parallel &= \sum_{x\in X} [\bar{\epsilon}(vx)\{k - \alpha(x)\} + \bar{\epsilon}(wx)\{(k+m) - \alpha(x)\}] \\ &\quad + \sum_{y\in Y} [\bar{\epsilon}(yv)\{\alpha(y) - k\} + \bar{\epsilon}(wy)\{(k+m) - \alpha(y)\}] \\ &\quad + \sum_{z\in Z} [\bar{\epsilon}(zv)\{\alpha(z) - k\} + \bar{\epsilon}(zy)\{\alpha(z) - (k+m)\}] \\ &\quad + m\bar{\epsilon}(wv) + \sum_{st\in B} \bar{\epsilon}(st)\{\alpha(s) - \alpha(t)\}.\end{aligned}$$

さらに，$\parallel B(\alpha_m^k) \parallel$ は上の式の v と w を交換するだけで得られる．それゆえに次の式を得る．

$$\| B(\alpha_m^k) \| - \| B(\alpha) \| = m\Bigg(\sum_{x\in X} \{\bar{\epsilon}(vx) - \bar{\epsilon}(wx)\} + \sum_{y\in Y} \{\bar{\epsilon}(vy) - \bar{\epsilon}(wy)\} + \sum_{z\in Z} \{\bar{\epsilon}(zw) - \bar{\epsilon}(zv)\} + \bar{\epsilon}(vw) - \bar{\epsilon}(wv) \Bigg) + \sum_{y\in Y} \{\alpha(y) - k\}\{\mu(wy) - \mu(vy)\}.$$

一方，次のようにも表される．

$$\sigma^+(v) - \sigma^+(w) = \sum_{x\in X} \{\bar{\epsilon}(vx) - \bar{\epsilon}(wx)\} + \sum_{y\in Y} \{\bar{\epsilon}(vy) - \bar{\epsilon}(wy)\} + \sum_{z\in Z} \{\bar{\epsilon}(vz) - \bar{\epsilon}(wz)\} + \bar{\epsilon}(vw) - \bar{\epsilon}(wv)$$

これらの式から補題の 1 つ目の等式が導かれる．第 2 式も $\sigma^-(v) - \sigma^-(w)$ を考えれば，同様にして得られる．(証明終)

一対比較グラフ D の順位づけ α について，α の逆枝の長さと同様に順枝の長さを定義することができる．D の順位づけ α は，順枝の長さの総和がこれら全ての D の順位づけの中で最大であるとき，順枝最適順位づけと定義できる．順枝最適順位づけは D の頂点の順位づけにも適用できるかもしれない．しかし，もし D を試合の結果として考えるならば，このときこの順位づけの方法は本論文で議論する逆方向の場合とぴったり同じではないようだ．しかしながら，順枝の長さを用いる順位づけの方法は異なった特性であり，そして他の応用には役に立つかもしれない．

第 4 章

特殊な一対比較グラフ

ここでは特殊な一対比較グラフ (完全一対比較グラフ，準完全一対比較グラフ) の最適順位づけの集合 $\mathrm{OR}(D)$ を簡単に求める問題について議論する．

4.1 完全一対比較グラフ

完全一対比較グラフとは全ての頂点間に枝が存在するグラフである．また，完全一対比較グラフにおいてすべての頂点 v に対して $d^*(v) = 0$ である．これは例えるなら総当たりリーグ戦の結果に対応する．

[補題 4.1] 頂点 v から出ている枝の重みの総和を $\sigma^+(v)$ とすると，頂点数 n の完全一対比較グラフの順位づけ α における逆枝の長さの総和 $\parallel B(\alpha) \parallel$ は次のように表される．

$$\parallel B(\alpha) \parallel = \sum_{v \in V} \sigma^+(v)\alpha(v) - \frac{1}{6}n(n^2-1)$$

(証明) n についての帰納法で等式を証明する．$n = 1$ のときは明らかである．$n = k$ であるとき，等式が成立すると仮定し，$n = k + 1$ とする．$\alpha(x) = n$ である頂点を x とし，$W = V(K)\backslash\{x\}$ とする．完全一対比較グラフ $K - x$ における帰納法の仮定から，次の式を得る．

$$\parallel B_K(\alpha) \parallel = \sum_{v \in W} \{\sigma_K^+(v) - \bar{\epsilon}_K(vx)\}\alpha(v) - \frac{1}{6}k(k^2-1) + \sum_{v \in W} \bar{\epsilon}_K(xv)\{n - \alpha(v)\}$$

$\bar{\epsilon}_K(xv) + \bar{\epsilon}_K(vx) = 1$ であるので，最後の項を次のように展開できる．

$$\sum_{v\in W}\bar{\epsilon}_K(xv)\{n-\alpha(v)\} = n\sum_{v\in W}\bar{\epsilon}_K(xv) + \sum_{v\in W}\{\bar{\epsilon}_K(vx)-1\}\alpha(v)$$
$$= \sigma_K^+(x)n + \sum_{v\in W}\bar{\epsilon}_K(vx)\alpha(v) - \frac{1}{2}k(k+1)$$

こうして次の式を得る.

$$\| B_K(\alpha) \| = \sum_{v\in W}\sigma_K^+(v)\alpha(v) + \sigma_K^+(x)n - \frac{1}{6}k(k+1)(k+2)$$
$$= \sum_{v\in W}\sigma_K^+(v)\alpha(v) - \frac{1}{6}n(n^2-1)$$

(証明終)

補題 4.1 より，完全一対比較グラフの最適順位づけ，順位値，最適な逆枝の長さの総和は次の定理より容易に得ることができる．ただし，頂点 v の順位値 $\pi(v)$ は次式で定義されるものである.

$$\pi(v) = \frac{1}{|\mathrm{OR}(D)|}\sum_{\alpha\in\mathrm{OR}(D)}\alpha(v)$$

ここで順位値について図 4.1 を用いて具体的に求めてみる．最適順位づけの集合 $\mathrm{OR}(D)$ は補題 4.1 より,

$$\mathrm{OR}(D) = \left\{ \begin{array}{l} [b,c,e,a,d],[b,e,c,a,d],[c,b,e,a,d], \\ [c,e,b,a,d],[e,b,c,a,d],[e,c,b,a,d] \end{array} \right\}$$

となる．最適順位づけは 6 種類あり，よって $|\mathrm{OR}(D)| = 6$ である．b の順位値は $\pi(b) = \frac{1}{6}\times(1+1+2+2+3+3) = 2$ である．同様に $\pi(c)=2, \pi(e)=2, \pi(a)=4, \pi(d)=5$ である.

[定理 4.2] 頂点数 n の完全一対比較グラフを K とする．このとき,

(1) K の順位づけ $\alpha = [v_1, v_2, \cdots, v_n]$ が，最適であるための必要十分条件は $\sigma^+(v_1) \le \sigma^+(v_2) \le \cdots \le \sigma^+(v_n)$ となることである.

(2) 任意の頂点 v に対して, $\xi = \sharp\{x\in V|\sigma^+(x) > \sigma^+(v)\}$, $\eta = \sharp\{y\in V|\sigma^+(y) = \sigma^+(v)\}$ とおくと，$\pi(v) = \xi + \frac{1}{2}(\eta+1)$ である.

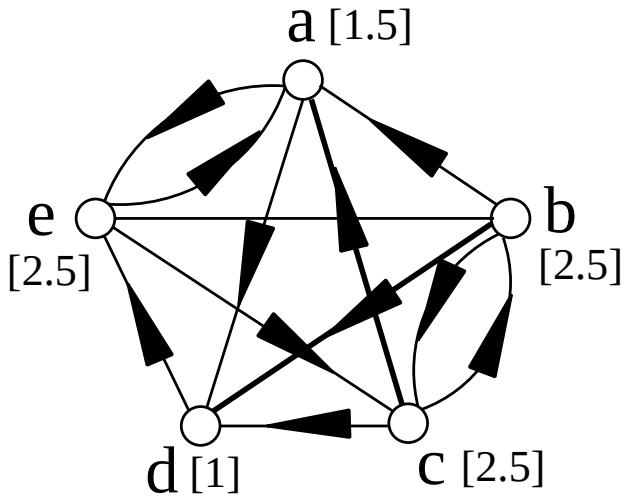


図 4.1　順位値

(3) $\alpha \in \mathrm{OR}(K)$ であれば，$l(K) = \sum_{v \in V} \sigma^+(v)\alpha(v) - \frac{1}{6}n(n^2-1)$ である．

(証明) (1) K の任意の最適順位づけを $\alpha = [v_1, v_2, \cdots, v_n]$ だとしよう．もし $i < j$ かつ $\sigma^+(v_i) < \sigma^+(v_j)$ であれば，補題 3.1 より x と y の全ての頂点間において $\mu(xy) = 1$ であるので $\parallel B(\alpha^i_{j-i}) \parallel - \parallel B(\alpha) \parallel = (j-i)[\sigma^+(v_i) - \sigma^+(v_j)] < 0$ となり，これは α が最適であることに矛盾している．従って，$\sigma^+(v_1) \leq \sigma^+(v_2) \leq \cdots \leq \sigma^+(v_n)$ である．一方，$\sigma^+(w_1) \leq \sigma^+(w_2) \leq \cdots \leq \sigma^+(w_n)$ を満たす順位づけを $\beta = [w_1, w_2, \cdots, w_n]$ とする．このとき，補題 4.1 より $\parallel B(\alpha) \parallel = \parallel B(\beta) \parallel$ が導かれ $\beta \in \mathrm{OR}(K)$ である．

(2) $\{y \in V | \sigma^+(y) = \sigma^+(v)\} = \{v = y_1, y_2, \cdots, y_\eta\}, |\mathrm{OR}(K)| = r$ とおく．このとき，(1) よりすべての $\alpha \in \mathrm{OR}(K)$ について $\{\alpha(y_1), \alpha(y_2), \cdots, \alpha(y_\eta)\} = \{\xi+1, \xi+2, \cdots, \xi+\eta\}$ であり，かつ $\pi(v) = \pi(y_1) = \pi(y_2) = \cdots = \pi(y_\eta)$ となる．

$$\begin{aligned}\eta\pi(v) &= \pi(y_1) + \pi(y_2) + \cdots + \pi(y_\eta) \\ &= \frac{1}{r}\sum_{i=1}^{\eta} \sum_{\alpha \in \mathrm{OR}(K)} \alpha(y_i) = \frac{1}{r}\sum_{\alpha}\sum_{i}\alpha(y_i)\end{aligned}$$

$$= \frac{1}{r}\sum_{\alpha}\sum_{i}(\xi + i) = \frac{1}{r}\sum_{\alpha}\{\xi\eta + \frac{1}{2}\eta(\eta+1)\}$$

$$= \xi\eta + \tfrac{1}{2}\eta(\eta+1)$$

それゆえに $\pi(v) = \xi + \dfrac{1}{2}(\eta+1)$ を得る．

(3) は補題 4.1 から明らかである．　(証明終)

ここで，完全一対比較グラフの最適順位づけを具体例に求める．例えば，次の図 4.2(a) のような完全一対比較グラフが与えられたとする．定理 4.2(1) より，完全一対比較グラフはスコアの大きい頂点の順に順位づけしたものが最適順位づけであることが証明されたので，図 4.2(a) の最適順位づけは $[c, d, a, b]$ となる．

一対比較グラフ D の二頂点 v と w が比較されていないとき (すなわち $\mu(vw) = 0$ のとき)，非順序対 $\{v, w\}$ を D の未比較対という．これは図 4.2(b) においては，頂点 b と d の対，a と c の対が未比較対である．D の全ての未比較対の集合と欠損集合といい $U(D)$ と書く．同じく図 4.2(b) においては，$U(D) = \{bd, ac\}$ である．$\{v, w\}$ が $U(D)$ に含まれているとき，D_{vw} は D に重み 1 の新たな枝 vw を加えた一対比較グラフを表す．すなわち，$D_{vw} = (V(D), E(D) + vw)$ である．$\{x, y\}$ が D の他の未比較対であるとき，$D_{vw,xy}$ は D_{vw} に重み 1 の新たな枝 xy を加えて得られる一対比較グラフである．

$\{v, w\} \in U(D)$ を満たし，D の未比較対 $\{v, w\}$ に対して枝 vw または wv をつけ加えて，完全な一対比較グラフにしたものは D の完全化グラフと言われる．$l(K) = l(D)$ を満たす D の完全化グラフ K を D の正規完全化グラフという．そして，D の正規完全化グラフの集合は $\mathrm{NC}(D)$ と表される．D の正規完全化グラフの存在は次のように容易に示すことができることに注意する．$\alpha \in \mathrm{OR}(D)$ とする．すべての $\{v, w\} \in U(D)$ に対して，$\{x, y\} = \{v, w\}$ かつ $\alpha(x) < \alpha(y)$ となる新しい枝 xy を D つけ加えることができる．この結果得られる完全一対比較グラフは D の正規完全化グラフである．

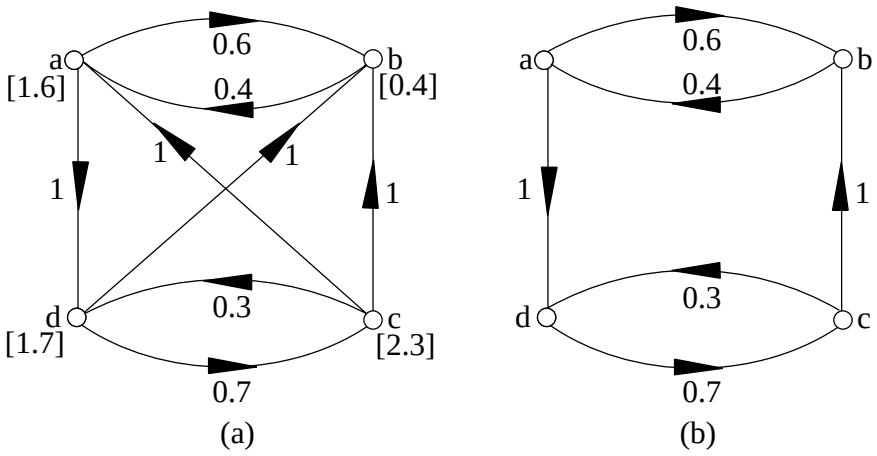


図 4.2　完全一対比較グラフと未比較対

4.2　準完全一対比較グラフ

D のすべての頂点がたかだか一つの未比較対に含まれているとき，つまり D の全ての頂点の v に対して $d^*(v)$ が 0 あるいは 1 であるとき，D は準完全一対比較グラフであるという．準完全一対比較グラフ D を総当たりリーグ戦に例えると，各選手がたかだか一つの未消化試合を残している状態に対応する．

未比較対 $\{v, w\}$ があるとき，一対比較グラフ D に重みが 1 の枝 vw をつけ加えて出来る一対比較グラフを D_{vw} と書く．

[定理 4.3] 準完全な一対比較グラフを D とする．このとき

(1) D の最適順位づけの集合は D の正規完全化グラフの最適順位づけの排他的和集合となる．つまり，

$$\mathrm{OR}(D) = \sum_{K \in \mathrm{NC}(D)} \mathrm{OR}(K) \qquad \text{(排他的和集合)}$$

である．

(2) D の完全化グラフ K が正規である必要十分条件は K が次の条件を満たすことである.

(i) $\{v,w\} \in U(D)$ に対して，$\sigma_D^+(v) > \sigma_D^+(w)$ であるとき，$vw \in E(K)$ である.

(ii) $\{v,w\} \in U(D)$ に対して，$\sigma_D^+(v) = \sigma_D^+(w)$ であるとき，$E(K)$ は vw と wv のいずれか一方だけを含む.

特に，$r = \sharp\{\{v,w\} \in U(D) | \sigma_D^+(v) = \sigma_D^+(w)\}$ であるとき，$|\,\mathrm{NC}(D)| = 2^r$ である.

(3) D の頂点 v に対して,

$$\pi_D(v) = \frac{1}{|\mathrm{NC}(D)|} \sum_{K \in \mathrm{NC}(D)} \pi_K(v)$$

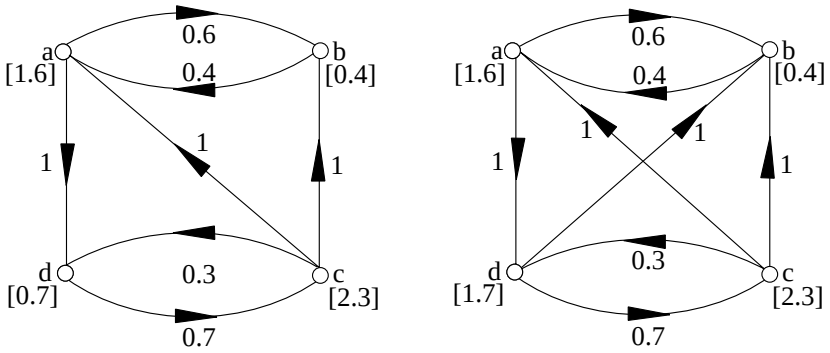
である.

ここで準完全一対比較グラフの最適順位づけを具体的に求める．定理 4.3 より，まず未比較対のスコアを比較する．図 4.3(a) の準完全一対比較グラフにおいて，未比較対は頂点 b と頂点 d の対である．スコアは $\sigma^+(b) = 0.4$,$\sigma^+(d) = 0.7$ であり，$\sigma^+(b) < \sigma^+(d)$ より，重み 1 の枝 db を図 4.3(a) に加える．すると，図 4.3(a) の準完全一対比較グラフは図 4.3(b) のような正規完全化グラフ K になる．よって，図 4.3(a) の準完全一対比較グラフの最適順位づけは $[c,d,a,b]$ である.

また，定理 4.3 より図 4.3(a) に重み 1 の枝 bd を加えたグラフは完全化グラフではあるが，正規完全化グラフではない.

ここで，もう一つ準完全一対比較グラフの例を用いて最適順位づけを説明する．先ほどの例は，未比較対が一つで頂点対 vw の出次数 $\sigma^+(v)$ と $\sigma^+(w)$ の値が異なっていた．そこで，今回は未比較対が複数でさらに出次数が等しいというやや複雑なケースについて考察する.

図 4.4 において準完全一対比較グラフ D を正規完全化グラフに変換する経過を説明する．まず最初に，$\sigma^+(b) < \sigma^+(d)$ より重み 1 の枝 bd を追加する．次に，$\sigma^+(a) = \sigma^+(c)$ より重み 1 の枝 ac, ca の 2 通りの正規完全化グラフが出来上がる．ここで，枝 ac を加えた正規完全化グラフを K_1, 枝 ca を加えた正規完全化グラフを K_2 と表す．つまりこの場合は．正規



(a)準完全一対比較グラフ　　(b)正規完全化グラフ

図 4.3　準完全一対比較グラフの正規完全化

完全化グラフは二つ存在する．よって，最適順位づけの集合 $\mathrm{OR}(D)$ は次のように表される．$\mathrm{OR}(D) = \mathrm{OR}(K_1) + \mathrm{OR}(K_2) = \{\alpha = [x_1, x_2, x_3, c, d] | \{x_1, x_2, x_3\} = \{a, b, e\}\} + \{\beta = [y_1, y_2, y_3, a, d] | \{y_1, y_2, y_3\} = \{b, c, e\}\}$ であり，$|\mathrm{OR}(K_1)| = 6, |\mathrm{OR}(K_2)| = 6$ なので $|\mathrm{OR}(D)| = 12$ となる．

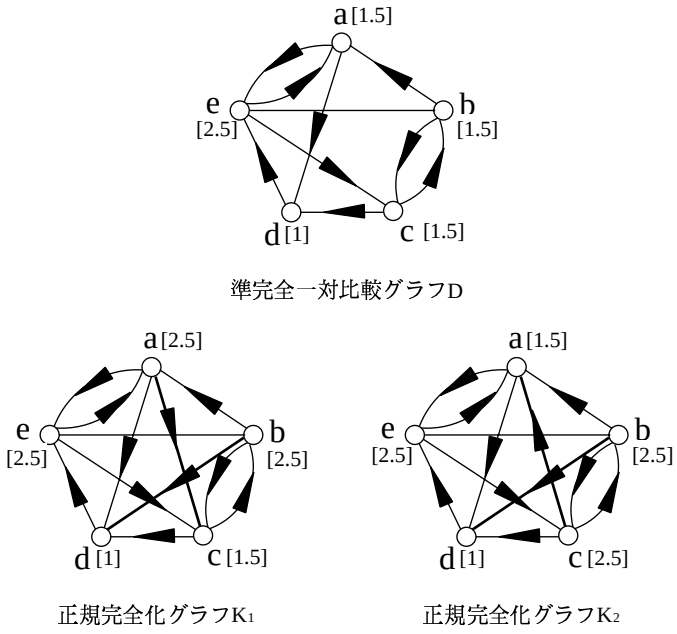


図 4.4　準完全一対比較グラフの正規完全化 (2)

第 5 章

最適順位づけの性質

この章の中では一対比較グラフの最適順位づけの性質を調べ，今まで求めてきた定理を用いて新たな定理の証明をする．そして，最終的には前章で示した定理 4.3 の証明をする．

5.1 最適順位づけの集合と最適な逆枝の長さ

[補題 5.1] 一対比較グラフ D において $\{v, w\} \in U(D)$ であるとする．このとき，次の (1), (2), (3) は等価である．

(1) $l(D_{vw}) < l(D_{wv})$

(2) すべての $\alpha \in \mathrm{OR}(D)$ について $\alpha(v) < \alpha(w)$ である．

(3) $\mathrm{OR}(D) = \mathrm{OR}(D_{vw})$ かつ $l(D) = l(D_{vw})$

さらに，次の (4), (5), (6) も等価である．

(4) $l(D_{vw}) = l(D_{wv})$

(5) $\alpha(v) < \alpha(w)$ かつ $\beta(v) < \beta(w)$ となるような二つの最適順位づけ α と β が存在する．

(6) $\mathrm{OR}(D) = \mathrm{OR}(D_{vw}) + \mathrm{OR}(D_{wv})$ かつ $l(D) = l(D_{vw}) = l(D_{wv})$

(証明) (1)$\Longrightarrow$(2) の証明．$l(D_{vw}) < l(D_{wv})$ とする．$\alpha(w) < \alpha(v)$ のような $\alpha \in \mathrm{OR}(D)$ が存在すると仮定する．このとき，次の式が得られる．

$$l(D) = \| B_D(\alpha) \| = \| B_{D_{wv}}(\alpha) \| \geq l(D_{wv}) > l(D_{vw})$$

ところが D は D_{vw} の部分有向グラフであるから，明らかに $l(D) \leq l(D_{vw})$ である．こ

れは上の不等式と矛盾する．従って，すべての $\alpha \in \mathrm{OR}(D)$ について $\alpha(v) < \alpha(w)$ である．

(2)$\Longrightarrow$(3) の証明．$\alpha \in \mathrm{OR}(D)$ とする．$\alpha(v) < \alpha(w)$ であるとき，次の式が得られる．

$$l(D_{vw}) \geq l(D) = \| B_D(\alpha) \| = \| B_{D_{vw}}(\alpha) \| \geq l(D_{vw})$$

この式から $l(D) = l(D_{vw})$ かつ $\alpha \in \mathrm{OR}(D_{vw})$ であることがわかる．特に，$\mathrm{OR}(D) \subseteq \mathrm{OR}(D_{vw})$ である．逆に，$\beta \in \mathrm{OR}(D_{vw})$ とする．このとき，

$$l(D) = l(D_{vw}) = \| B_{D_{vw}}(\beta) \| \geq \| B_D(\beta) \| \geq l(D)$$

となる．それゆえ，$\beta \in \mathrm{OR}(D)$ であり，$\mathrm{OR}(D) \supseteq \mathrm{OR}(D_{vw})$ が成立する．

(3)$\Longrightarrow$(1) は後で証明する．

(4)$\Longrightarrow$(5) の証明．$l(D_{vw}) = l(D_{wv})$ とする．一般性を失うことなく，$\alpha'(v) < \alpha'(w)$ のような $\alpha' \in \mathrm{OR}(D)$ があるとする．すべての $\alpha \in \mathrm{OR}(D)$ について $\alpha(v) < \alpha(w)$ であると仮定して矛盾を導く．このとき，(2) から (3) への証明から $l(D) = l(D_{vw}) = l(D_{wv})$ となる．$\gamma \in \mathrm{OR}(D_{wv})$ とする．もし $\gamma(v) < \gamma(w)$ であるなら，次の式が得られる．

$$l(D) = l(D_{wv}) = \| B_{D_{wv}}(\gamma) \| = \| B_D(\gamma) \| + \gamma(w) - \gamma(v) > \| B_D(\gamma) \| \geq l(D)$$

これは矛盾であり，従って $\gamma(w) < \gamma(v)$ である．それゆえ．

$$l(D) = l(D_{wv}) = \| B_{D_{wv}}(\gamma) \| = \| B_D(\gamma) \|$$

であり，これは $\gamma \in \mathrm{OR}(D)$ であることを意味している．従って，すべての $\alpha \in \mathrm{OR}(D)$ について $\alpha(v) < \alpha(w)$ とした仮定が誤りであることが示された．

(5)$\Longrightarrow$(6) の証明．$\mathrm{OR}(D)$ を次の二つの部分集合に分割する．

$$\mathrm{OR}_1 = \{\alpha \in \mathrm{OR}(D) | \alpha(v) < \alpha(w)\},\ \mathrm{OR}_2 = \{\alpha \in \mathrm{OR}(D) | \alpha(w) < \alpha(v)\}$$

$\alpha(v) < \alpha(w)$ を満たす $\alpha \in \mathrm{OR}(D)$ が存在するとき，$\mathrm{OR}(D_{vw}) = \mathrm{OR}_1$ かつ $l(D) = l(D_{vw})$ であることを証明する．$\alpha \in \mathrm{OR}_1$ とする．このとき，次の式が得られる．

$$l(D_{vw}) \geq l(D) = \| B_D(\alpha) \| = \| B_{D_{vw}}(\alpha) \| \geq l(D_{vw})$$

上の式から $l(D) = l(D_{vw})$ かつ $\alpha \in \mathrm{OR}(D_{vw})$ である．特に，$\mathrm{OR}(D_{vw}) \supseteq \mathrm{OR}_1$ である．一方，$\beta \in \mathrm{OR}(D_{vw})$ とする．このとき，

$$l(D) = l(D_{vw}) = \| B_{D_{vw}}(\beta) \| \geq \| B_D(\beta) \| \geq l(D)$$

は，$\beta \in \mathrm{OR}(D)$ と $\beta(v) < \beta(w)$ を意味する．それゆえ，$\beta \in \mathrm{OR}_1$ を得て，$\mathrm{OR}(D_{vw}) \subseteq \mathrm{OR}_1$ が成立する．従って，$\mathrm{OR}(D_{vw}) = \mathrm{OR}_1$ が証明された．それゆえ，(6) は証明された．

(6)$\Longrightarrow$(4) の証明．これは自明である．

(3)$\Longrightarrow$(1) の証明．$l(D_{vw}) \geq l(D_{wv})$ が成立すると仮定する．このとき，$l(D_{wv}) \geq l(D) = l(D_{vw})$ なので，$l(D_{vw}) = l(D_{wv}) = l(D)$ である．従って，(4) が成立し，(6) も成立する．(6) は (3) と逆なので，よって，(3) ならば (1) が証明できた．　(証明終)

[補題 5.2] 一対比較グラフを D，$\{v, w\} \in U(D)$ とする．このとき，次の式を得る．

(1) $l(D) = \min\{l(D_{vw}), l(D_{wv})\}$

(2)$\mathrm{OR}(D) = \begin{cases} \mathrm{OR}(D_{vw}) & l(D_{vw}) < l(D_{wv}) \text{ のとき}, \\ \mathrm{OR}(D_{wv}) & l(D_{vw}) > l(D_{wv}) \text{ のとき}, \\ \mathrm{OR}(D_{vw}) + \mathrm{OR}(D_{wv}) & l(D_{vw}) = l(D_{wv}) \text{ のとき} \end{cases}$

(3) $l(D) = l(D_{vw})$ であるとき，$\mathrm{OR}(D) \supseteq \mathrm{OR}(D_{vw})$ であり，すべての $\alpha \in \mathrm{OR}(D_{vw})$ について，$\alpha(v) < \alpha(w)$ である．

(証明) これらは補題 5.1 の結果より容易に求められる．　(証明終)

$l(D) < l(D_{vw})$，$\mathrm{OR}(D) \supseteq \mathrm{OR}(D_{vw})$ のような D や D_{vw} が存在することに注意する．それゆえ，$l(D) < l(D_{vw})$ であることが $\mathrm{OR}(D) \not\supseteq \mathrm{OR}(D_{vw})$ や $\mathrm{OR}(D) \cap \mathrm{OR}(D_{vw}) = \emptyset$ を導くのではない．さらに，$\mathrm{OR}(D) \supseteq \mathrm{OR}(D_{vw})$ ならば，$l(D) = l(D_{vw})$ ではない．

たとえば，図 5.1 のトーナメントを T，T から有向枝 ad を取り除いて得た準完全な一対比較グラフを D とする．このとき，定理 3.2，定理 3.3 から $l(D) = l(T) = 7 < l(D_{da}) = 10$，$\mathrm{OR}(D) \supset \mathrm{OR}(D_{da})$ を得る．

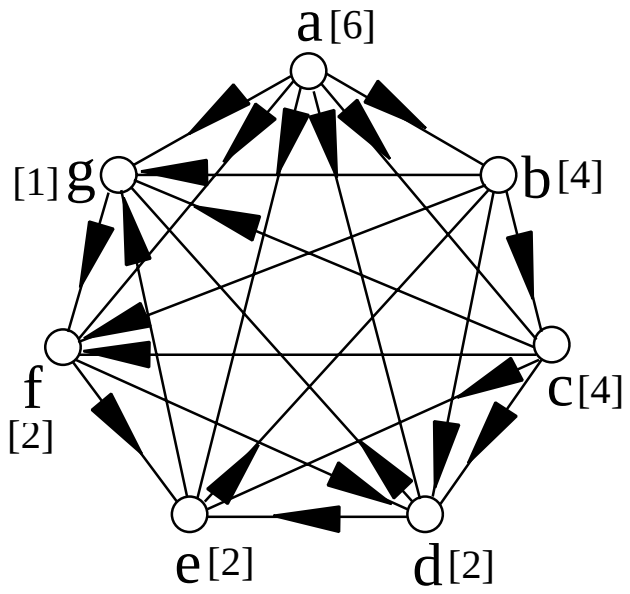


図 5.1　トーナメント T

5.2　正規完全化グラフ

次の定理における $\mathrm{NC}(D)$ は，$l(K) = l(D)$ を満たす D の正規完全化グラフ K の集合である．

[定理 5.3] 一対比較グラフを D とする．$\mathrm{NC}(D) = \{K_1, K_2, \cdots, K_r\}$ であるとき，$\mathrm{OR}(D) = \mathrm{OR}(K_1) + \mathrm{OR}(K_2) + \cdots + \mathrm{OR}(K_r)$ である．

(証明) $\{a, b\}$ と $\{c, d\}$ が D において未比較対とする．このとき，補題 5.2 より次のこと

を得る.

$$\mathrm{OR}(D) = \sum \mathrm{OR}(D_{vw}) = \sum \mathrm{OR}(D_{vw,xy}) \text{ (排他的和集合)}$$

最初の $\sum$ は $\{v,w\} = \{a,b\}$ かつ $l(D_{vw}) = l(D)$ である D_{vw} についての総和であり，二つ目の $\sum$ は $\{v,w\} = \{a,b\}$，$\{x,y\} = \{c,d\}$ かつ $l(D_{vw,xy}) = l(D)$ である $D_{vw,xy}$ についての総和である．この手続きの繰り返しにより，この定理を得る．(証明終)

5.3　未比較対

これまでの結果から得られるこの方法の長所を示す．たとえば，次の定理の (1) は，w が消化していない $d^*(w)$ 試合のすべてに勝ったとしても，w のスコアが v のスコアより大きくならない，つまり $\sigma^+(v) > \sigma^+(w) + d^*(w)$ であれば，v は w より優れていることを教えてくれる.

[定理 5.4] 一対比較グラフを D とし，D の二頂点を v と w とする．このとき，

(1) $\mu(vw) = 1$ かつ$\sigma^+(v) > \sigma^+(w) + d^*(w)$ であるとき，すべての $\alpha \in \mathrm{OR}(D)$ において $\alpha(v) < \alpha(w)$ である.

(2) $\mu(vw) = 0$ かつ$\sigma^+(v) > \sigma^+(w) + d^*(w) - 1$ であるとき，すべての $\alpha \in \mathrm{OR}(D)$ において $\alpha(v) < \alpha(w)$ である．特に，$\mathrm{OR}(D) = \mathrm{OR}(D_{vw})$ である.

(3) $\sigma^+(v) > \sigma^+(w)$ かつ $\{x \in V | \mu(vx) = 1\} = \{y \in V | \mu(wy) = 1\}$ であるとき，すべての $\alpha \in \mathrm{OR}(D)$ において $\alpha(v) < \alpha(w)$ である.

(証明) 最初に (1) の証明をする．定理 4.2 より D は完全グラフではないと仮定する．定理 5.3 より D のすべての最適順位づけ α は D のある正規完全化グラフ K の最適順位づけの一つである．$\sigma_K^+(w) \leq \sigma_D^+(w) + d_D^*(w) < \sigma_D^+(v) \leq \sigma_K^+(v)$ なので，定理 3.2 から $\alpha(v) < \alpha(w)$ が導ける．従って (1) が証明される.

次に (2) の証明をする．補題 5.2 より D のすべての最適順位づけ α は，$\{x,y\} \in U(D)\backslash\{\{v,w\}\}$ であるそれぞれの x と y について有向枝 xy と yx のいずれか一方のみを D に加え，かつ $l(H) = l(D)$ であるようなある一対比較グラフ H の最適順位づけの一つである．$\alpha(w) = k < \alpha(v) = k + m$ と仮定する．頂点 $x(\neq v,w)$ について $\mu_H(vx) = \mu_H(wx) = 1$ なので，補題 3.1 より $\sigma_H^+(w) \leq \sigma_D^+(w) + d_D^*(w) < \sigma_D^+(v) \leq \sigma_H^+(v)$ として $\parallel B_H(\alpha_m^k) \parallel - \parallel B_H(\alpha) \parallel = m\{\sigma_H^+(w) - \sigma_H^+(v)\} < 0$ を得る．これは $\alpha \in \mathrm{OR}(H)$ に反する．その結果，$\alpha(v) < \alpha(w)$ である．

(3) は補題 3.1 より同様に証明することが出来る．　(証明終)

5.4　定理 4.3 の証明

定理 4.3 を証明するために，次の補題を必要とする．

[補題 5.5] 一対比較グラフ D の頂点を v と w とする．このとき，

(1) $\sigma^+(v) > \sigma^+(w)$ かつ $d^*(v) = d^*(w) = 1, \mu(vw) = 0$ であるとき，すべての $\alpha \in \mathrm{OR}(D)$ について $\alpha(v) < \alpha(w)$ である．特に，$\mathrm{OR}(D) = \mathrm{OR}(D_{vw})$ かつ $l(D) = l(D_{vw})$ である．

(2) $\sigma^+(v) = \sigma^+(w), d^*(v) = d^*(w) = 1, \mu(vw) = 0$, かつ $\alpha \in \mathrm{OR}(D)$ について $\alpha(v) = k < \alpha(w) = k + m$ であるとき，$\alpha_m^k \in \mathrm{OR}(D)$ である．特に，$\mathrm{OR}(D) = \mathrm{OR}(D_{vw}) + \mathrm{OR}(D_{wv})$ かつ $l(D) = l(D_{vw}) + l(D_{wv})$ である．

(証明) (1) は定理 5.4 の中の (3) から導かれる．(2) に関しては，補題 3.1 から $\parallel B(\alpha) \parallel - \parallel B(\alpha_m^k) \parallel = 0$ を得るので，補題 5.1 から導かれる．　(証明終)

(定理 4.3 の証明)

(1) は定理 5.3 よりすぐに求められる.

(2) は補題 5.5 の簡単な結果である.

これから (3) を証明する．D の二つの正規完全化グラフを K,K' とする．$\{v,w\} \in U(D)$ であるとき，(2) より $\{\sigma^+_K(v), \sigma^+_K(w)\} = \{\sigma^+_{K'}(v), \sigma^+_{K'}(w)\}$ を得る．したがって，任意の正の実数 t に対して，$\sharp\{v \in V(K) | \sigma^+_K(v) = t\} = \sharp\{v \in V(K') | \sigma^+_{K'}(v) = t\}$ であり，定理 3.2 より $|\mathrm{OR}(K)| = |\mathrm{OR}(K')|$ を得る．よって定理 5.3 より次の式を得る.

$$\begin{aligned}
\pi_D(v) &= \frac{1}{|\mathrm{OR}(D)|} \sum_{\alpha \in \mathrm{OR}(D)} \alpha(v) \\
&= \frac{1}{|\mathrm{NC}(D)| \cdot |\mathrm{OR}(K)|} \sum_{K \in \mathrm{NC}(D)} \sum_{\alpha \in \mathrm{OR}(K)} \alpha(v) \\
&= \frac{1}{|\mathrm{NC}(D)|} \sum_K \left\{ \frac{1}{|\mathrm{OR}(K)|} \sum_\alpha \alpha(v) \right\} \\
&= \tfrac{1}{|\mathrm{NC}(D)|} \textstyle\sum_K \pi_K(v)
\end{aligned}$$

(定理 4.3 の証明終)

第 6 章

一般の一対比較グラフの最適順位づけ

ここでは完全一対比較グラフや準完全一対比較グラフなどの特殊なクラスに限定しない一般の一対比較グラフの最適順位づけについて議論する．一般の最適順位づけを求める方法は二種類考えられる．

6.1 一般の一対比較グラフの最適順位づけの求め方

第 3 章で最適順位づけについて述べた．このとき，最適順位づけとは全ての順位づけの中で逆枝の長さの総和が最小の順位づけであると定義した．しかし，特殊なグラフは全ての順位づけを求めなくても，容易に最適順位づけが求められることを示した．だが，一般のグラフはそのように容易に最適な順位づけが求められない．よって本研究では全ての順位づけの中で逆枝の長さの総和が最小の順位づけを求めるプログラムを作成した．

6.2 疑問点

前章で完全一対比較グラフと準完全一対比較グラフはスコアの大きい頂点の順に順位づけをしたものが最適順位づけとなっていた．そこで，一般の一対比較グラフにおいてもスコアの大きい順に順位づけしたものが必ず最適順位づけとなるのかどうかという疑問を抱くのは自然である．

6.3　結果

まず，一般の一対比較グラフの例として次の図 6.1 を考察する．この一対比較グラフは頂点数が 8 である．この一対比較グラフの全ての順位づけが 8 の階乗で表されることは明確である．よって，$8! = 40320$ 通りの順位づけの逆枝の長さの最小値を求めた．ここで得られた最適順位づけを以下に示す．

$$\mathrm{OR}(D) = \left\{ \begin{array}{l} [a, f, g, e, b, h, d, c], [a, f, g, h, b, e, d, c], \\ [e, a, f, g, b, h, d, c], [h, a, f, g, b, e, d, c] \end{array} \right\}$$

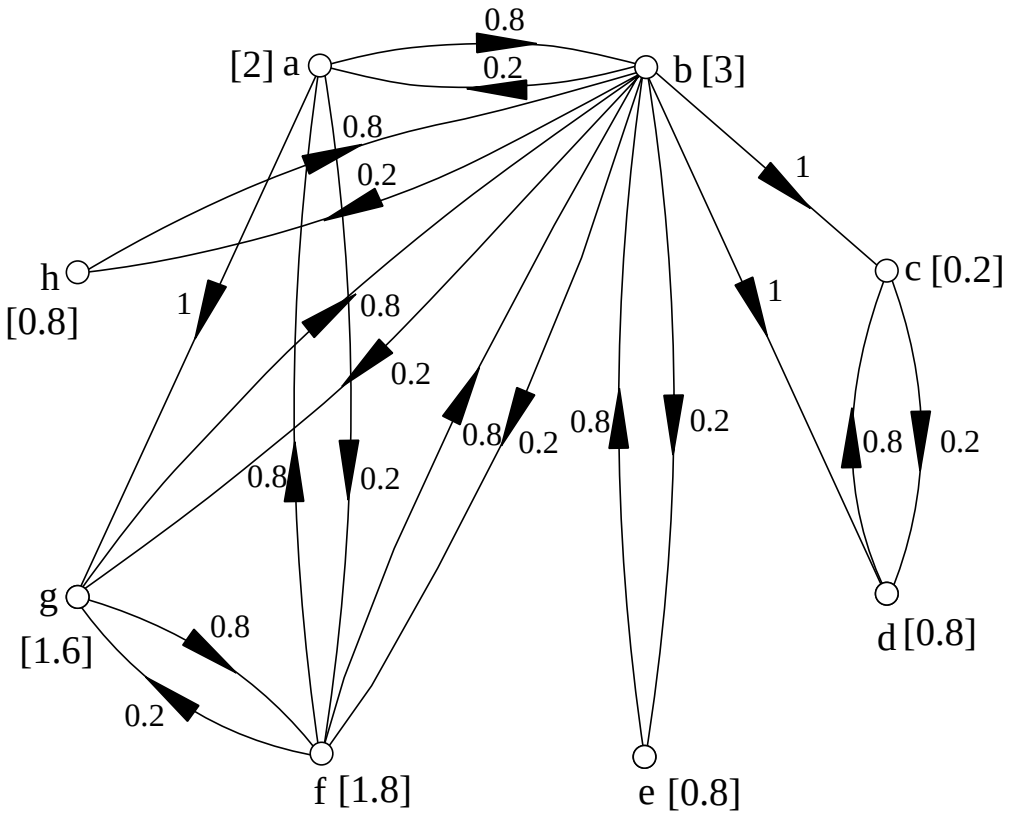


図 6.1　一般の一対比較グラフ

図 6.1 の一対比較グラフの頂点 a と b に注目する．ここで頂点 a と b のスコアを見てみると，頂点 a のスコアは 2 で b のスコアは 3 となっており，b のスコアの方が大きい値となっている．しかし，上に示した最適順位づけの順位を見てみると頂点 a が頂点 b の順位を全て上回っている．これより，一般の一対比較グラフはスコアの大きい頂点の順の順位づけが最適順位づけになるとは限らないということを立証できた.

6.4　一般の一対比較グラフの最適順位づけを求める方法の長所と欠点

この全ての順位づけを求めてから最適順位づけを求める方法の長所は枝の数によって多少は異なるが，計算時間がある程度予測がつけられるということである.

しかし，この方法は残念ながらいいアルゴリズムではないと考えられる．なぜなら，次の図 6.2 のような図を説明すると理解できる．ただし，ここでは [　] の中の値はスコアではなく $d^*(v)$(比較されていない頂点数) の値である．この図は準完全一対比較グラフから枝を 1 本取り除いただけである．このような場合は，全ての順位づけを求めるよりは完全化グラフにして最適順位づけを求めた方が数段早いからである．しかし，この方法は枝がほとんど存在しない場合，逆に遅くなってしまう．よって，どちらも完全に優れた求め方ではないので，頂点に対する枝の数に対して，どちらの方法が適しているかを判断できれば，一般の一対比較グラフの最適順位づけはもっと高速に求めることが出来ると考えられる．ちなみに図 6.2 の最適順位づけは [a, d, c, h, b, e, f, g] の一つだけである.

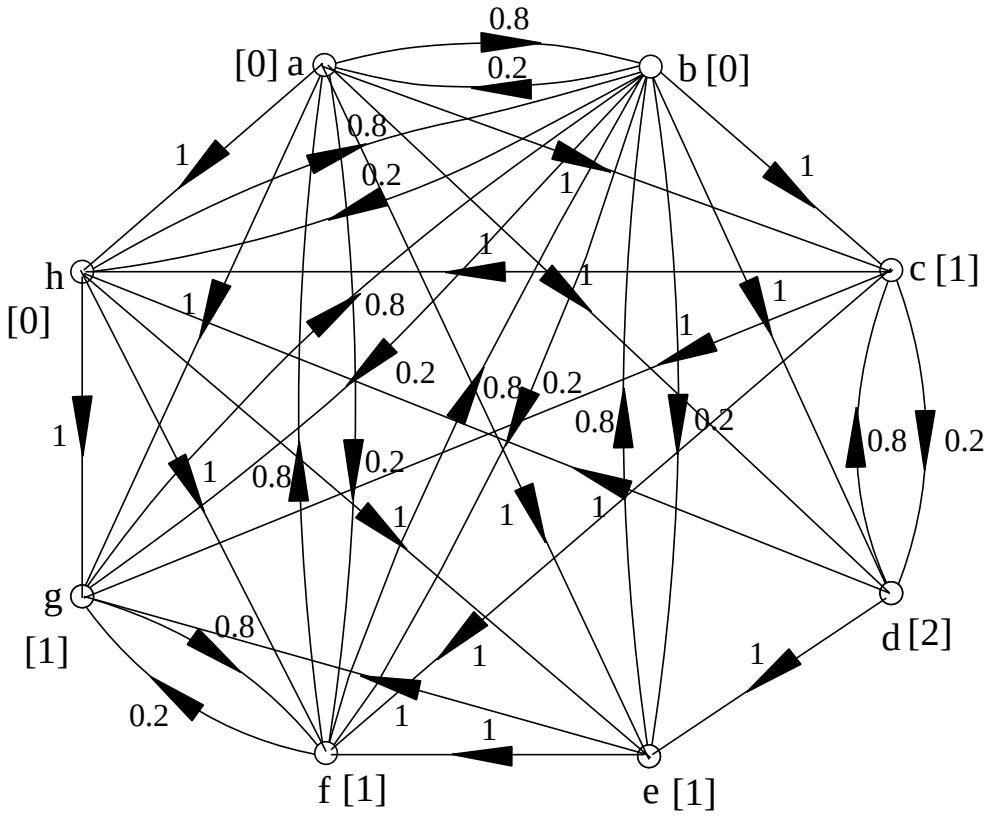


図 6.2　一般の一対比較グラフ (2)

第 7 章

まとめ

本論文では，一対比較グラフの頂点の順位づけについて考察した．順位づけに矛盾した有向枝を逆枝として，その長さの総和が最小な順位づけを最適順位づけと定義した．

特殊な一対比較グラフにおいては最適順位づけが容易に求められることが分かった．

さらに，任意の一対比較グラフの最適順位づけを求めるプログラムを作成した．このプログラムは付録として最後に載せている．なお，このプログラムは第 6 章の一対比較グラフの最適順位づけを求める際に用いた．

次にこのプログラムを用いて特殊な一対比較グラフにはない特徴を一般の一対比較グラフの最適順位づけの中から発見した．

最後に，今後残された課題としては完全あるいは準完全ではない一般の一対比較グラフの最適順位づけを効率よく求めるアルゴリズムを考案することである．一般の一対比較グラフの最適順位づけを求めるプログラムは作成できたが，その効率は必ずしも良くはない．それは最適順位づけを求める際に，枝の数に関係なく全ての順位づけを求めるからである．もう一つの正規完全化グラフを作る方法と併用できるアルゴリズムを考案すれば，もっと高速に一般の最適順位づけが求まると考えられる．

謝辞

本論文は，著者が 2000 年 7 月から 2002 年 2 月までの高知工科大学情報システム工学科在学中に，同学科坂本研究室において行った研究の成果を記したものである．

全般的にアドバイスを下さった坂本明雄先生，ありがとうございました．先生のことはこの先 10 年は忘れないと思います．

LaTeX について色々教えてくれた登伸一君，あと一年大学院で頑張って下さいね．くれぐれも体調には気を配って下さい．

気がついたらいなくなっていた折橋祐一君，君へ未来の一体どこへ向かって・・・．

C 言語やパソコンの使い方を教えてくれた橋本 学殿，仕事大変だろうけど辞めないで続けて下さいね．あと，一年大学にいて欲しかったです．

一緒に研究を頑張り，某 I 君の悪愚痴いい過ぎていた栃木隆道君，君はいつも研究を熱心にしていましたね．そして，君の勇姿はいつまでも忘れないよ！！！ コナミは君を得てさらに大きく羽ばたくことでしょう！

あまり研究室で見かけなかったけど，研究室での写真を一緒に撮った田村和香那さん，さようなら．

来年度も坂本研かどうか分からない 3 年生のみなさん，来年は卒論が大変だとは思いますけど，頑張って下さい！ その前に，就職活動もね．

あと，隣の福本研の山本真弘君，パチンコ頑張って下さい．セントラル万歳！！ 嶋岡哲夫君，サングラス似合ってますよ！ 秋山由佳様，色々手伝って下さいましてありがとうございました．心の奥底から感謝いたしております．あと，東京弁と Little 毒舌をまき散らしていた福本昌弘先生，教授目指して頑張って下さい！！！

最後に，外見は奇麗だけど中の壁には微妙にヒビが入っている高知工科大学と私を色々お世話して下さった先生方に感謝致します．

参考文献

[1] Mikio Kano and Akio Sakamoto, “Ranking the vertices of a paired comparison digraph”, *SIAM J. Alg. Disc. Meth* , Vol. 6, No. 1, pp.79-92, January 1985.

[2] 坂本 明雄，加納 幹雄，“逆枝集合による単純な有向グラフの頂点の順位づけ問題”, 電子通信学会論文誌，Vol. 62-A, No. 12, pp.833-846，December 1979.

[3] 大山 達夫, “パワーアップ離散数学”, 共立出版, 1997.

付録 A

プログラムリスト

```
/* Program name: pcdfile.c */

#include<stdio.h>

#define COMPLETE 1
#define SEMICOMP 2
#define OTHERS 3
#define LT -1
#define EQ 0
#define GT 1
#define MINIMUM 1.0e-6
#define YES 1
#define NO 0
#define TRUE 1
#define FALSE 0

int node, count, opticount;

int compare(float x, float y);
float *getPCD(FILE *fp);
int optimalR(float *sigma);
float *makeSigma(float *epsB);
void swap(int *array, int i, int j);
int *makeUncomp(float *epsB, float *sigma);
int nextOne(int *uncompPair);
float backLength(float *epsB, int *rank);
void genperm(float *epsB);



int main(int argc, char *argv[])
{
    int i, j, numberOf \mathrm{OR}, *uncompP, *rank, sumback;
    float *epsB, *sigma;
    char *ok;
    FILE *fp;
    if ((fp = fopen(argv[1], "r")) == NULL) {
        printf("Input file can't open!! \n");
        return 1;
    }
    epsB = getPCD(fp);
    fclose(fp);
```

```
    switch (checkPCD(epsB)) {
        case COMPLETE:
            printf("このグラフは完全一対比較グラフです！！ \n");
            sigma = makeSigma(epsB);
            numberOf \mathrm{OR} = optimalR(sigma);
            break;
        case SEMICOMP:
            printf("このグラフは準完全一対比較グラフです！！ \n");
            sigma = makeSigma(epsB);
            uncompP = makeUncomp(epsB, sigma);
            if (uncompP[0] == -1)
                numberOf \mathrm{OR} = optimalR(sigma);
            else {
                numberOf \mathrm{OR} = 0;
                do {
                    for (i = 0; uncompP[i] != -1; i += 2)
                        sigma[uncompP[i]] += 1.0;
                    for (i = 0; i < node; i++)
                        printf("sigma[%d] = %f \n", i, sigma[i]);
                    numberOf \mathrm{OR} += optimalR(sigma);
                    for (i = 0; uncompP[i] != -1; i += 2)
                        sigma[uncompP[i]] -= 1.0;
                    for (i = 0; i < node; i++)
                        printf("sigma[%d] = %f \n", i, sigma[i]);
                } while (nextOne(uncompP) == YES);
            }
            printf("the number of total optimal rankings = %d \n", numberOf \mathrm{
            break;
        case OTHERS:
            printf("その他です！！ \n");
            genperm(epsB);
            break;
    }
    return 0;
}


/* グラフの識別 */
int checkPCD(float *epsB)
{
    int i, j;
    int min = node, mu;

    for (i = 0; i < node; i++) {
        mu = 0;
        for (j = 0; j < node; j++) {
            if (epsB[i * node + j] != 0.0 || epsB[j * node + i] != 0.0)
                mu += 1;
        }
        if (min > mu)
            min = mu;
    }
```

```
    if (min == node - 1)
        return COMPLETE;
    else if (min == node - 2)
        return SEMICOMP;
    else
        return OTHERS;

}



/* sigma[] の計算 */
float *makeSigma(float *epsB)
{
    int i, j;
    float *sigma;

    sigma = (float *) calloc(node, sizeof(float));
    for (i = 0; i < node; i++)
        for (j = 0; j < node; j++)
            sigma[i] += epsB[i * node + j];
    for (j = 0; j < node; j++) {
        printf("sigma[%d] = %f", j, sigma[j]);
        printf("\n");
    }
    return sigma;
}



/* float の比較 */
int compare(float x, float y)
{
    float difference = x - y;

    if (difference < -MINIMUM)
        return LT;
    if (difference > MINIMUM)
        return GT;
    else
        return EQ;
}



/* 一対比較グラフを入力 */
float *getPCD(FILE *fp)
{
     int i, j;
     float *epsB;

     /* 頂点数を入力 */
     fscanf(fp, "%d", &node);
     epsB = (float *) calloc(node * node, sizeof(float));
```

```
    while (1) {
        fscanf(fp, "%d", &i);
        fscanf(fp, "%d", &j);
        if (i != j) {
            fscanf(fp, "%f", &epsB[i * node + j]);
            if (epsB[i * node + j] >= 0.0 && epsB[i * node + j] <= 1.0) {
                printf(" 始点 %d 終点 %d 重み %f", i, j, epsB[i * node + j]);
                epsB[j * node + i] = 1.0 - epsB[i * node + j];
            }
            else
            {
                printf("caution : 0 <= 重み <=1 の範囲でもう一度入力して下
さい！！ \n");
                epsB[i * node + j] = 0.0;
            }
        }
        else
            break;
        printf("\n");
    }
    printf("\n");
    printf("-----------------------------------\n");
    printf("入力したデータ\n");
    printf("\n");
    printf("頂点数 %d x %d のグラフ\n", node, node);
    printf("      ");
    for (i = 0; i < node; i++)
        printf("%3d      ", i);
    printf("\n");
    for (i = 0; i < node; i++) {
        printf("%2d |  ", i);
        for (j = 0; j < node; j++)
            printf(" %f", epsB[i * node + j]);
        printf("\n");
    }
    return epsB;
}


/* 完全一対比較グラフの最適順位づけを表記 */
int optimalR(float *sigma)
{
    int i, j, count = 0, *temp, tempn, number = 1, factor;
    float max;

    temp = (int *) calloc(node, sizeof(int));
    printf("optimal ranking(s) = [ ");
    while (count < node) {
        temp[0] = 0;
        max = sigma[0];
        tempn = 1;
        for (i = 1; i < node; i++)
```

```
            /* max を保存する */
            switch (compare(max, sigma[i])) {
                case LT:
                    max = sigma[i];
                    tempn = 1;
                    temp[0] = i;
                    break;
                case EQ:
                    temp[tempn++] = i;
                    break;
            }
        count += tempn;
        if (tempn == 1) {
            printf("%d", temp[0]);
            sigma[temp[0]] -= (float) node;
        }
        else {
             printf("{ ");
            factor = 1;
            for (j = 0; j < tempn; j++) {
                number *= factor++;
                printf("%d", temp[j]);
                sigma[temp[j]] -= (float) node;
                 if (j < tempn - 1)
                    printf(", ");
            }
             printf(" }");
        }
        if (count < node)
            printf(", ");
    }
    printf(" ] \n");
    printf("the number of optimal rankings = %d \n", number);
    for (i = 0; i < node; i++)
        sigma[i] += (float) node;
    free(temp);
    return number;
}



/* 二つの整数の数値を入れ換える */
void swap(int *array, int i, int j)
{
    int dummy;

    dummy = array[i];
    array[i] = array[j];
    array[j] = dummy;
}



/* 未比較対を抽出する */
```

```
int *makeUncomp(float *epsB, float *sigma)
{
    int *uncompPair, uncompNo = 0, i, j, m;

    uncompNo = 0;
    uncompPair = (int *) calloc(node + 1, sizeof(int));
    for (i = 0; i < node; i++)
        for (j = i + 1; j < node; j++) {
            if (compare(epsB[i * node + j] + epsB[j * node + i], 1.0) == EQ)
                continue;
            switch (compare(sigma[i], sigma[j])) {
                case GT:
                    sigma[i] += 1.0;
                    break;
                case LT:
                    sigma[j] += 1.0;
                    break;
                case EQ:
                    uncompPair[uncompNo++] = i;
                    uncompPair[uncompNo++] = j;
                    break;
            }
        }

    uncompPair[uncompNo] = -1;
    printf("uncompNo = %d \n",uncompNo);
    for (i = 0; i <= uncompNo; i++)
        printf("unconpPair[%d] = %d \n", i, uncompPair[i]);
    for (i = 0; i < uncompNo; i++)
        printf("sigma[%d] = %f \n", uncompPair[i], sigma[uncompPair[i]]);
    return uncompPair;
}



/* 次の完全化グラフの設定 */
int nextOne(int *uncompPair)
{
    int uncompNo = 0;

    do {
        swap(uncompPair, uncompNo, uncompNo + 1);
        if (uncompPair[uncompNo] > uncompPair[uncompNo + 1])
            return YES;
        uncompNo += 2;
    } while (uncompPair[uncompNo] != -1);
    return NO;
}



/* 逆枝の和を求める */
float backLength(float *epsB, int *rank)
{
```

```
    int i, j;
    float sumback = 0.0;

    for (j = 0; j < node - 1; j++)
        for (i = j + 1; i < node; i++)
            sumback += (i - j) * epsB[(rank[i] - 1) * node + (rank[j] - 1)];
    printf("sumback = %f \n", sumback);
    return sumback;
}



/* 順列生成 */
float put(float *epsB, int pos, int k, int *rank, char *ok,
          int *optirank,float minimum)
{
    int i, j;
    float sumback;

    rank[pos] = k;
    if (pos == node - 1) {
        count++;
        printf("%5d: ", count);
        for (i = 0; i < node; i++)
            printf(" %d", rank[i]);
        printf("\n");
        sumback = backLength(epsB, rank);
        /* 最小な逆枝の長さの総和とその順位づけを保存する */
        switch (compare(minimum, sumback)) {
            case EQ:
                opticount++;
                for (i = 0; i < node; i++)
                optirank[opticount * node + i] = rank[i];
                break;
            case GT:
                minimum = sumback;
                opticount = 0;
                for (i = 0; i < node; i++)
                optirank[opticount * node + i] = rank[i];
                break;
        }
        printf("now, optimal backward length = %f \n",minimum);
    }
    else {
        ok[k] = FALSE;
        /* その数はもう使えないことを記憶 */
        for (j = 1; j <= node; j++)
            if (ok[j])
                minimum = put(epsB, pos + 1, j, rank, ok, optirank, minimum);
        /* 次の位置にまだ使える数をいれるように自分自身の複製に頼む */
        ok[k] = TRUE;
    }
    return minimum;
}
```

```
/* 並び替え */
void genperm(float *epsB)
{
    int k, i, *rank, *optirank;
    float minimum;
    char *ok;

    opticount = 0;
    count = 0;
    rank = (int *) calloc(node, sizeof(int));
    optirank = (int *) calloc(node, sizeof(int));
    ok = (char *) calloc(node + 1, sizeof(char));
    minimum = (float) node * node;

    for (k = 1; k <= node; k++)
        ok[k] = TRUE;
    /* 各数 1,..,node がまだ使えるかどうかを示す標識 ok に
       初期値として true を代入する */
    for (k = 1; k <= node; k++)
        minimum = put(epsB, 0, k, rank, ok, optirank, minimum);
    /* 0 番目の位置に 1 から node までの数を入れるように頼む */
    printf("final optimal backward length = %f \n", minimum);
    for (i = 0; i <= opticount; i++) {
        printf("optimal rank[%d] = [", i);
        for (k = 0; k < node; k++) {
            if (k < node - 1)
                printf(" %d,",optirank[i * node + k] - 1);
            else
                printf(" %d ] \n",optirank[i * node + k] - 1);
        }
    }
}
```