(学術論文) 日本経営数学会誌, Vol.36, No.1/2, pp.1-16, 2014.

一対比較におけるループの解消と緩和な整合性制約を満たすためのアルゴリズム

Algorithms for Eliminating Loops on Pairwise Comparisons and Satisfying Loose Consistency Constraints

高 萩 栄 一 郎*
Eiichiro TAKAHAGI

概要：AHP などで整合的な一対比較を行うための 3 つのアルゴリズムを提案する．一対比較の各 2 項目間の選好関係に注目すれば，選好関係の推移性が存在しないことがある．そこで，推移性を成立させていないループを発見，指摘する．次に一対比較のやり直しをすることにより，推移性を満たすようにすることにより全順序にする．次に緩和な整合性制約を満たさない一対比較値を指摘し，満たすように改善する一対比較値の候補を提示するアルゴリズムを提案する．また，項目の順序を指定し，推移性，緩和な整合性制約を満たす範囲に一対比較値を制限することにより，整合的な一対比較を行うアルゴリズムを提案する．

Abstract: In this paper, we propose three algorithms for pairwise comparisons in a consistent manner. The results of pairwise comparisons such as AHP models may include the loops of preference relations. The first algorithm identifies the loops that are intransitive and show some alternative pairwise comparison values to satisfy the transitive law. Then, the preference relations of the results satisfy the totally ordered relation. The second algorithm identifies three pairwise comparisons values among any three elements that do not satisfy the loose consistency constraints and show some alternative pairwise comparison values. The third algorithm, by setting the rank of elements and by restricting the range of pairwise comparison values, obtains result that satisfies the transitive law and the loose consistency constraints.

Keywords (キーワード): Analytic Hierarchy Process (AHP), Pairwise Comparison (一対比較), Transitive Law (推移性), Loose Consistency Constraint (緩和な整合性制約), Total Order (全順序)

1. はじめに

AHP[1] での一対比較は，対象の要素 (項目) の組に対して，どちらがどれくらいよいかの一対比較値を回答する．その一対比較値は，重要度を同定するのに最低限必要な回数より多くの回数の比較を行っている．いい替えれば,「余分」（多重）に一対比較を行うことにより，同定する重要度の精度を向上させようとしている．また，回答は「ことば」によるものであり，また，基本的には，17 段階での比較であるので，こまかな差異を表現できず，多重に行うことで精度を向上させようとしている．

* 専修大学（Senshu University）
受付：2014 年 11 月 30 日，受理：2015 年 2 月 17 日

しかし，多重に一対比較を行っていくと，回答状況により，矛盾や整合性を満たさないことが起こる．極端な例は,「じゃんけん」のような 3 すくみの状態である．A, B, C の 3 つの要素に対して，$A \succ B$ かつ $B \succ C$ のとき，$C \succ A$ と回答した場合である．これは，推移性を満たさない回答で，整合性を満たさない．本稿では，まず，このような回答をしたとき，矛盾した回答として，指摘するアルゴリズムを提案する (3 節)．全体から不整合な部分を切り出し，その部分の一対比較をやり直すように促す．

次に，緩和な整合性制約 [2] を導入する．緩和な整合性制約は，3 すくみの状態に程度（一対比較値）をいれ，$A \succ B$ の程度と $B \succ C$ の程度の大きい方よりも，$A \succ C$ の程度より大きくなくてはならないというものである．たとえば,『「かなり」（一対比較値 5）で $A \succ B$』 かつ『「うんと」（一対比較値 7）で $B \succ C$』 のとき,『$A \succ C$ は,「うんと」（一対比較値 7）以上』とならなくてはいけないという制約である．$A \succ C$ が「少し」（一対比較値 3）や「かなり」（一対比較値 5）では，緩和な整合性制約を満たさない．4.2 節で，緩和な整合性制約を満たさない部分を指摘するアルゴリズムを提案する．もし，このような緩和な整合性制約を満たさない部分を発見したら，緩和な整合性制約を満たす範囲を提示し（多くの場合複数ある），その範囲内で一対比較をやり直す．

本稿では，できるだけ正確な一対比較を行うことを目標とする.「できるだけ正確な一対比較」とは，ループが存在しない（推移性を満たす），緩和な整合性制約を満たす一対比較とする．したがって，一対比較回数の低減は目標としない．要素数の増大による整合的な一対比較の困難を減少させることを目的とする．一対比較の終了後，ループの存在や緩和な整合性制約を満たさない部分を指摘し，それを修正することにより精度の良いものにする．

最初から，緩和な整合性制約を満たした一対比較値のみを回答範囲にするという方法も考えられる．5 節で，最初に順位を指定し，緩和な整合性制約を満たす範囲に回答領域を限定しながら一対比較を行う方法も提案する．

2. 記号・定義・概略

一対比較の対象の集合を $X = \{x_1, \ldots, x_n\}$ とする．n を一対比較の対象の要素数とする．ただし，$n \geq 3$ とする．

定義 1 (一対比較値) a_{ij} を x_i の x_j に対する一対比較値とする．*AHP* の慣例に従い，

$$V \equiv \{9, 8, \ldots, 2, 1, 1/2, 1/3, \ldots, 1/9\} \tag{1}$$

$$a_{ij} \in V, \forall i > j \tag{2}$$

とする．また，$a_{ii} = 1, \forall i$，$a_{ij} = 1/a_{ji}, \forall i, j$ とする．a_{ij} の一対比較行列を A とする．

一対比較値とことばの対応表は，表 1 を想定している．

定義 2 (要素間の選好関係 $\succeq$) 要素間の選好関係を $\succeq$ で表す．ただし，$\succ$ と $\sim$ を次のように定義する．

$$x_i \succeq x_j \text{ かつ } x_j \not\succeq x_i \text{ のとき } x_i \succ x_j \text{ と記す.} \tag{3}$$

$$x_i \succeq x_j \text{ かつ } x_j \succeq x_i \text{ のとき } x_i \sim x_j \text{ と記す.} \tag{4}$$

表 1: 一対比較値とことばの対応

9	8	7	6	5	4	3	2	1
圧倒的によい	中間	うんとよい	中間	かなりよい	中間	少しよい	中間	同じくらいよい

定義 3 (全順序集合) 集合 X 上の選好関係 $\succeq$ が次の関係を満たすとき，全順序集合という．

反射性 任意の $x_i \in X$ について，$x_i \succeq x_i$

反対称性 任意の $x_i, x_j \in X$ について，$x_i \succeq x_j$ かつ $x_j \succeq x_i$ ならば，$x_i \sim x_j$

推移性 任意の $x_i, x_j, x_k \in X$ について，$x_i \succeq x_j$ かつ $x_j \succeq x_k$ ならば，$x_i \succeq x_k$

比較可能性 任意の $x_i, x_j \in X$ について，$x_i \succeq x_j$ または $x_j \succeq x_i$

定義 4 (ループ) X から任意の $m(\geq 3)$ 個の要素を取り出し，$x_1 \succ \ldots \succ x_m$ かつ $x_m \succ x_1$ という関係が存在し，推移性が満たされないときループが存在するという．

定義 5 (緩和な整合性制約 [2]) *]* $A = \{x_i, x_j, x_k\} \subseteq X$ を全順序集合とし，$a_{ij} > 1, a_{jk} > 1 (x_i \succ x_j \succ x_k)$ とする．

$$a_{ik} \geq \max(a_{ij}, a_{jk}) \tag{5}$$

を満たすとき，A は緩和な整合性制約を満たすという．X について要素数 3 のすべての部分集合 A で緩和な整合性制約を満たすとき，X は緩和な整合性制約を満たすという．

次の手順で，できるだけ正確な一対比較を行う．

(1) X 間の一対比較を行う．

- 同じ要素は比較せず，$a_{ii} = 1$ とし，$x_i \sim x_i$ とする．反射性を満たす．
- 異なる 2 要素間の一対比較は必ず行われるとする．a_{ij} は与えられ，比較可能性は満たされる．

(2) ループが存在する部分の一対比較をやり直し，推移性を満たすようし，X を順序集合にする (節).

(3) X の要素数 3 のすべての部分集合 A で，緩和な整合性制約を満たすように一対比較をやり直し，X を緩和な整合性制約を満たすようする (4 節).

3. ループの発見とその除去のための一対比較のやり直しアルゴリズム

一対比較を行ってもその一対比較値が全体として，全順序集合になっているとは限らない．本節では，全順序集合になるように一対比較を修正するアルゴリズムを提案する．一対比較値は，回答拒否を許さず，すべてに対して存在するとする．

3.1 概略

X の中の互いに異なる 3 つの要素 x_i, x_j, x_k を取り出し，もし，ループがあれば，その 3 つの要素を 1 つの集合にまとめていく．この作業を繰り返していくと，X をループが存在する要素の同値類に分割できる．要素の数が 1 ではない同値類は，ループが存在することになるので，その同値類内のすべて要素間の一対比較をループが存在しないようにやり直す．この作業を，すべての同値類の要素数が 1 になるまで繰り返す．

Q_i を X の同値類とする．$X = Q_1 \cup Q_2 \cup \ldots \cup Q_m$，$Q_i \cap Q_j = \emptyset, \forall i, j \in \{1, \ldots, m\}$ となるようにしていく．

本節では 2 項関係に基づいてループを発見していく．

$$x_i \succ x_j \ \text{ if } \ a_{ij} > 1 \tag{6}$$

$$x_i \prec x_j \ \text{ if } \ a_{ij} < 1 \tag{7}$$

$$x_i \sim x_j \ \text{ if } \ a_{ij} = 1 \tag{8}$$

$a_{ij} = 1$ は,「同じくらい重要」を表しているので，$\sim$ を割り当てる．しかし，$x_i \succ x_j$ または $x_i \prec x_j$ の可能性があるとする．他の 2 項関係から，$\succ$ または $\prec$ のどちらかを仮定する必要があるときは，アルゴリズムの中でそれを仮定し，割り当て直していく．

2 項関係としたとき，推移性を満すようにするために，一部の $\sim$ を $\succ$ と仮定する．$x_i \sim x_j$ は，$a_{ij} = 1$ を意味する．これは，一対比較の言葉では,「同じくらい重要」であるので，おおよそ 1, すなわち $a_{ij} \simeq 1$ であり，$\sim$ は，$\succeq$ と $\preceq$ の両方が考えられる．ループの存在や推移律の不成立が解消される場合，片方のみを仮定する．

仮定するのは次の 2 つの関係があるときである．

(1) $x_i \succ x_j$ かつ $x_j \succ x_k$ かつ $x_i \sim x_k$ のとき，$x_i \succ x_k$ を仮定する．

(2) $x_i \succ x_k$ かつ $x_i \sim x_j$ かつ $x_j \sim x_k$ のとき，推移律が成立しない．すなわち $x_i \preceq x_j$ かつ $x_j \preceq x_k$ であるのに，$x_i \preceq x_k$ は成立していない．$x_i \succ x_j$ と $x_j \succ x_k$ を仮定して，$x_i \succ x_j \succ x_k$ とし，推移律を満たすようにする．x_j は，x_i と x_k とほとんど差がなく，したがって，x_j は x_i と x_k の間に位置すると考えた．$x_j \succ x_i \succ x_k$ と考えるのは，$x_j \sim x_k$ と回答していることから不自然であると考えた．$x_j \succ x_i \succ x_k$ も，$x_j \succ x_k$ と回答していない点から不自然と考えた（$x_i \succ x_k \succ x_j$ や $x_i \succ x_k \sim x_j$ も同様）．この仮定は，3.2 節の Step 7 で利用する．

3.2 ループ解消のアルゴリズム

Step 1: 初期の状態は，それぞれの要素が 1 つの集合とする．$Q_i := \{x_i\}, i = 1, \ldots, n$ とする．一対比較値から次のように 2 項関係を割り当てる．

$$x_i \succ x_j \ \text{ if } \ a_{ij} > 1 \tag{9}$$

$$x_i \prec x_j \ \text{ if } \ a_{ij} < 1 \tag{10}$$

$$x_i \sim x_j \ \text{ if } \ a_{ij} = 1 \tag{11}$$

Step 2: $Loop\_FALG := OFF$ とする．

Step 3: X から異なる 3 つの要素の組を選び，そのすべての組について，以下の Step 8 までを繰り返す．取り出した要素を x_i, x_j, x_k とする．

Step 4: 3 つの要素のうち，2 つ以上が同じ Q_p の要素だったら，Step 8 へ

Step 5: $x_i \succ x_j$ かつ $x_j \succ x_k$ かつ $x_k \succ x_i$ のときループが発生しているので，x_i, x_j, x_k が所属する集合を結合する．
x_i が所属する集合を Q_i, x_j が所属する集合を Q_j, x_k が所属する集合を Q_k とする．$Q_i := Q_i \cup Q_j \cup Q_k$ とし，$Q_j := \emptyset, Q_k := \emptyset$ とする． Setp 8 へ

Step 6: 節の (1) の仮定より，$x_i \succ x_j$ かつ $x_j \succ x_k$ かつ $x_i \sim x_k$ ならば，$x_i \succ x_j$ かつ $x_j \succ x_k$ より，$x_i \succ x_k$ が必要となり，$x_i \succ x_k$ とする．$Loop\_FALG := ON$ とする．

Step 7: 節の (2) の仮定より，$x_i \succ x_k$ かつ $x_i \sim x_j$ かつ $x_j \sim x_k$ のとき，$x_i \succ x_j$ と $x_j \succ x_k$ とする．$Loop\_FALG := ON$ とする．

Step 8: すべてのパターンが終了したら Step 9 へ，残りがあれば，次の x_i, x_j, x_k を取り出し，Step 4 へ

Step 9: $Loop\_FALG = ON$ ならば，Step 2 へ．$Loop\_FALG = OFF$ ならば，Step 10 へ

Step 10: $Q_i = \{x_i\}, i = 1, \ldots, n$ であれば，終了．

Step 11: $\mid Q_i \mid > 1$ となる Q_i について，Q_i の要素間の一対比較をやりなす．Step 1 へ戻る．

修正の一対比較 (Step 11) を繰り返し，各 Q の要素数が 1 となり（$\mid Q_i \mid = 1, i = 1, \ldots, n$）なれば，任意の 3 要素間でループが発生していないことになり，したがってループは存在していない．

3.3 全順序集合

3.2 節のアルゴリズムが終了した時点で，$\sim$ を $\succ$ とした（仮定した）部分は，他の全 3 要素間でループは発生していないので，そのまま $\succ$ を採用する．

定理 1 で示したアルゴリズムを適用した X は全順序集合である．

[証明]
仮定した $\succ$ を含んだ $\succeq$ の 2 項関係で，X が全順序集合になっているか検討する．反射律，反対称律，比較可能性は明らかに成立し，推移律

$$x_i \succeq x_j \text{かつ } x_j \succeq x_k \text{のとき } x_i \succeq x_k \tag{12}$$

について，任意の x_i, x_j, x_k について，それらの関係について，3 つが $\succ$ であれば，Step 5, 6 によりループはなく，推移律は成立する．3 つが $\sim$ であれば，推移律は成立する．
2 つが $\succ$ で 1 つが $\sim$ の場合，x_i, x_j, x_k の添え字を入れ替えれば，次の 3 つのパターンが考えられる．

(1) $x_i \succ x_j$**,** $x_j \succ x_k$**,** $x_i \sim x_k$　この場合は，Step 6 より，このような関係は存在しない．

(2) $x_i \succ x_j$**,** $x_i \succ x_k$**,** $x_j \sim x_k$　この場合，推移性を崩す関係ではない．

(3) $x_j \succ x_i$**,** $x_k \succ x_i$**,** $x_j \sim x_k$　この場合，推移性を崩す関係ではない．

1 つが $\succ$ で 2 つが $\sim$ では，Step 7 より，推移律が不成立の関係は存在しない．したがって，推移律は成立する．(証明終)

したがって，X は，$\succeq$ に関して，全順序集合となり，(同順位を認めて）順位を振ることができる．

また，$\sim$ については，Step 7 より，$x_i \sim x_j$ かつ $x_j \sim x_k$ のとき，$x_i \not\sim x_k$ ということはなく，推移律が成立し，したがって X は $\sim$ に関して同値関係になる．

4. 緩和な整合性制約を満たすように一対比較のやり直しを行うアルゴリズム

4.1 概要

すでに，3 節で，ループが除去され，全順序集合となっているものとする．X の要素の番号を $x_1 \succeq x_2 \succeq \ldots \succeq x_n$ となるように添え字を付け直す．

x_i と x_k ($x_i \succ x_k$) を選び，$x_i \succ x_j \succ x_k$ となるすべての x_j について，緩和な整合性制約をチェックし，満たしていなければ，選択できる範囲を示しながら，一対比較を修正する．

チェックする順番は，できるだけ順位の差が大きい要素同士を行う．$k-i$ が大きい順に行う．大きな差異の要素間の一対比較値 (a_{ik}) を示し，その間の要素間の一対比較値 (a_{ij} と a_{jk}) は，大きな差異の要素間の一対比較値以下にしていく．

緩和な整合性制約が満たされたいないとき，$a_{ik} \geq \max(a_{ij}, a_{jk})$ のうち，a_{ij} または a_{jk}（または両方) を a_{ik} 以下に修正することを基本とする．ただし，a_{ik} も変更できるとし，そのとき修正できる範囲は，i,k を含む他の 3 項間の緩和な整合性制約を崩さないように $i:k$ よりも広い範囲での一対比較値（順位の差が大きい要素間の一対比較値）の最小値以下とする．

4.2 緩和な整合性制約を満たさない部分を発見するアルゴリズム

本節では，3 節で求めた選好関係の順序に基づいて，緩和な整合性制約を満たすように一対比較を修正していくアルゴリズムを示す．

Step 1: 節で要素間の順位が決まった．また $\sim$ に関して同値関係にあるので m 個の同値類 $Z_1, \ldots, Z_m$ を構成する．また，集合間については，強意の選好関係 ($\succ$) が成立し，選好される順に，$Z_1, \ldots, Z_m$ とする．

$$x_i \sim x_j \text{ if } x_i \in Z_p, x_j \in Z_p \tag{13}$$

$$x_i \succ x_j \text{ if } x_i \in Z_p, x_j \in Z_q, p < q \tag{14}$$

Step 2: 差 $distance$ を $m-1$ から はじめ，1 ずつ減らしながら $distance = 2$ まで 繰り返し行う (Step 2 $\sim$ 5 まで)．

Step 3: $r - p = distance$ となる 集合 Z_p と Z_r $(p < r)$ のすべての組み合わせについて， Step 4 から Step 5 まで繰り返す.

Step 4: $\forall x_i \in Z_p, \forall x_k \in Z_r$ について，a_{ik} の最小値を求め，それを a^*_{pr} とする.

$$a^*_{pr} = \min_{x_i \in Z_p, x_j \in Z_r} a_{ij} \tag{15}$$

Step 5: $p < q < r$ とし，$\forall (x_i, x_j, x_k) \in (Z_p \times Z_q \times Z_r)$ に対して $a^*_{pr} \geq \max(a_{ij}, a_{jk})$ となるか調べる．もし，成立しない場合，その (x_i, x_j, x_k) について緩和な整合性を満たすように，節のアルゴリズムで修正入力させる.

Step 5 において，Z_p または Z_r の要素数が 2 以上の場合，緩和な整合性制約より厳しい制約になることがある．たとえば，$Z_p = \{x_1\}, Z_q = \{x_2\}, Z_r = \{x_3, x_4\}$ の場合，$x_1 \succ x_2 \succ (x_3 \sim x_4)$ となる．$a_{13} < a_{14}$ の場合，$a^*_{pr} = a_{13}$ となる．このとき，$a_{24} \leq a^*_{pr} = a_{13}$ という制約がかかる．これは，$a_{24} \leq a_{14}$ より厳しい制約である.

4.3 緩和な整合性制約を満たさない場合の一対比較のやり直し

緩和な整合性制約を満たしていない場合，満たすように一対比較のやり直しを行う．4.2 節で，$p < q < r$ のとき，ある $x_i \in Z_p, x_j \in Z_q, x_k \in Z_r$ について，緩和な整合性制約 $a_{ik} \geq \max(a_{ij}, a_{jk})$ が満たされていないとする．この場合の修正方法は，**(a)** a_{ij} と a_{jk} との大きい方（または，両方）の一対比較値を a_{ik} 以下にすること, もしくは，**(b)**a_{ik} の値を $\max(a_{ij}, a_{jk})$ 以上にすることである.

(a) の場合:

$a_{ij} \leq a_{ik}$ かつ $a_{jk} \leq a_{ik}$ となるように，a_{ij} または a_{jk} (もしくは両方) を修正する.

(b) の場合:

$x_i \in Z_p$ と $x_k \in Z_r$ の一対比較値 a_{ik} の取り得る値の最大値 $a^\dagger_{pr}$ を求める

$$a^\dagger_{pr} = \min_{(x_l, x_t) \in [[(Z_1 \cup \ldots \cup Z_p) \times (Z_r \cup \ldots \cup Z_m)] \setminus (Z_p \times Z_r)]} a_{lt} \tag{16}$$

ただし，$p = 1, r = m$ の場合，$a^\dagger_{pr} = 9$（取り得る一対比較値の最大値）とする．$V \cap [1, a^\dagger_{pr}]$ が a_{ik} の取り得る範囲である.

4.4 緩和な整合性制約が満たされる理由

このアルゴリズムが終了した時点で，緩和な整合性制約は満たされる．任意の 3 つの要素 x_i, x_j, x_k について，$x_i \succ x_j \succ x_k$ であれば，すべて異なる集合に含まれる．$x_i \in Z_p, x_j \in Z_q, x_k \in Z_r$, とすれば，$a_{ik} \geq a^*_{pr}$ である．4.2 節の Step 5 の条件より成立する.

5. 順序を指定して，緩和な整合性制約を満たす一対比較

5.1 概要

4 節で，全体の一対比較を行った後に，ループを除去し，順序を確定し，緩和な整合性制約を満たすように一対比較を行い，修正する方法を示した．本節では，最初に順序を指定し，その順序に従った範囲で一対比較を行う方法を示す．

最初に，各要素を（同順位を認めた）順序を指定する．その順序に従った一対比較値の範囲のみで一対比較を行う．4 節と同様に，比較する x_i と x_k について，$(k-i)$ が大きい順に行う．

5.2 アルゴリズム

入力した順位に基づき，要素の添え字を $x_1 \succeq x_2 \succeq \ldots \succeq x_n$ となるように付け直す．

Step 1: 同順位の要素を 1 つの集合 Z_p に纏め，選好される順に，$Z_1, \ldots, Z_m$(m は集合の数) とする．

$$x_i \sim x_j \text{ if } x_i \in Z_p, x_j \in Z_p \tag{17}$$

$$x_i \succ x_j \text{ if } x_i \in Z_p, x_j \in Z_q, p < q \tag{18}$$

Step 2: $distance := m - 1$ とする．

Step 3: $\forall(p, r)$ ただし，$p + distance = r, 0 < p < m, 0 < r \leq m$ について，Z_p と Z_r の要素間で一対比較を行う．

$$a^{\dagger}_{pr} := \min_{(x_l, x_t) \in [[(Z_1 \cup \ldots \cup Z_p) \times (Z_r \cup \ldots \cup Z_m)] \setminus (Z_p \times Z_r)]} a_{lt} \tag{19}$$

ただし，$p = 1, r = m$ の場合，$a^{\dagger}_{pr} := 9$ とする．
$\forall(x_i, x_j) \in Z_p \times Z_r$ について，一対比較を行う．ただし，一対比較値の範囲を $[1, a^{\dagger}_{pr}]$ に制限する ($a_{ij} \in V \cap [1, a^{\dagger}_{pr}]$).

Step 4: $distance$ を 1 減じ，2 以上だったら，Step 1 へ，1 だったら Step 5 へ

Step 5: 残りの一対比較値を次のように設定し終了する．

$$a_{ij} := 1, \qquad \forall i, j \in Z_p, p = 1, \ldots, m, \tag{20}$$

$$a_{ii} := 1, \qquad i = 1, \ldots, n, \tag{21}$$

$$a_{ji} := 1/a_{ij}, \qquad i = 1, \ldots, n-1, j = i+1, \ldots, n \tag{22}$$

6. 他の整合性指標，不整合箇所の指摘法との比較

6.1 C.I.(整合度, consistency index)

C.I.[1] は，現在 AHP でもっともよく使われている指標である．A を一対比較値 a_{ij} からなる $n \times n$ 行列とし，A の最大固有値を $\lambda_{\max}$ として，

$$C.I. = \frac{\lambda_{\max} - n}{n-1} \tag{23}$$

で定義されている．完全に整合的であるとき 0 で，値が高くなるにつれ，不整合になるとされている．経験的な値として，0.1 もしくは 0.15 以上のとき，不整合の度合いが高いとして，一対比較のやり直しが推奨されるとされている．

また，n が大きくなるにつれ，この C.I. が大きくなる傾向があるとして，乱数で定めた一対比較行列による C.I. の値を基に，n により異なるランダム整合度 R.I. を求めている．C.I. と R.I. の比，整合比 C.R.

$$C.R. = \frac{C.I.}{R.I.} \tag{24}$$

を整合性の指標として使われている．

両指標とも，一対比較行列全体をみて，整合的かどうか指摘している．本稿では，基本的には，3 要素に注目し，その要素間で推移性が成立するか，緩和な整合性制約を満たすか検討する方法を提案している．全 3 要素間で推移性，緩和な整合性制約を満たすようにしている．したがって，C.I. とは別の考え方である．

経験的な言い方ではあるが，推移性が満たさないことが解消されたり，緩和な整合性制約を満たさない部分が解消されると，C.I. の値は減少する．

6.2 整合性を悪化させている一対比較の指摘

整合性を悪化させている一対比較の指摘法として，刀根の方法 [4] がある．AHP では，行列 A の第 1 固有ベクトル $(w_1, \ldots, w_n)^t, \sum w_i = 1$ を求め，w_i を i 番目の要素の重要度としている．整合的な一対比較行列を

$$b_{ij} = w_i / w_j \tag{25}$$

として，B とする．この B の C.I. を求めると 0 なる．刀根の方法は，b_{ij} と a_{ij} の値を比較し，差異が大きい一対比較値をやり直しの対象とし，より整合的にする方法である．

中島の方法 [5] は，一対比較行列 A のうち，1 つの要素 $a_{ij}(i > j)$ を欠損値とし，ハーカーの方法 [6] で重要度を求め，その時の $C.I_{ij}$ を求める．ハーカーの方法は，欠損値を $a_{ij} = w_i/w_j$ として補うので，C.I. の値は改善する．この改善の度合いがもっとも大きい a_{ij} を一対比較のやり直しの候補とする方法である．

両方法とも全体を見て，不整合な一対比較値を指摘する方法である．

本手法は，基本的に 3 要素間で整合的になるように一対比較を検討させる方法である．すべての 3 要素間で整合的（推移性および緩和な整合性制約を満たす）であれば，全体でも整合的であると考える．経験的ではあるが，全体で整合的であるとき，C.I. の値はかなり小さくなる．

7. 作成したシステム

作成したシステムは,

http://cgi.isc.senshu-u.ac.jp/%7thc0456/AHPnewPC/

で公開している (または, ループ 緩和な整合性制約 で Web 検索). 本稿での説明した手法に,「理想点・満足点を考慮」した手法 [3] を加えたプログラムを提供している.

7.1 ループの発見と除去

(1) 上記 URL にアクセスし,「ループと緩和な整合性制約をチェック」,「理想点満足点を指定しない」を選ぶ（上記 URL の Web ページ中の一覧表の「E」を選択）.

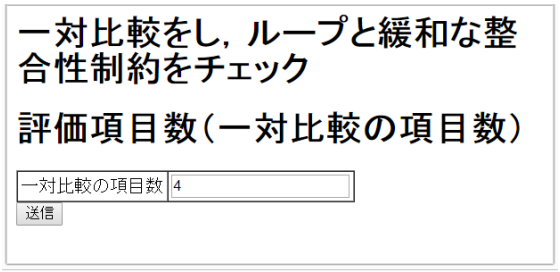

図 1: 項目数の入力

(2) 図 1 のように，一対比較の項目数を入力する．例えば,「4」と入力した.

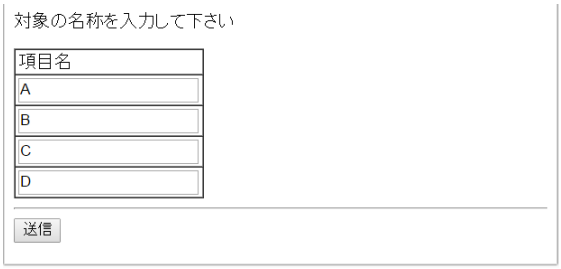

図 2: 項目名の入力

(3) 図 2 のように，一対比較の対象の名称（評価項目名もしくは代替案名）を入力する．ここでは,「A」,「B」,「C」,「D」とした.

(4) 図 3 のように一対比較値の選択を行う.

(5) 図 4 のように，3 つ（もしくはそれ以上）の要素間で，ループがないか調べ，あればその部分の一対比較をやり直す.

図 3: 一対比較値の入力

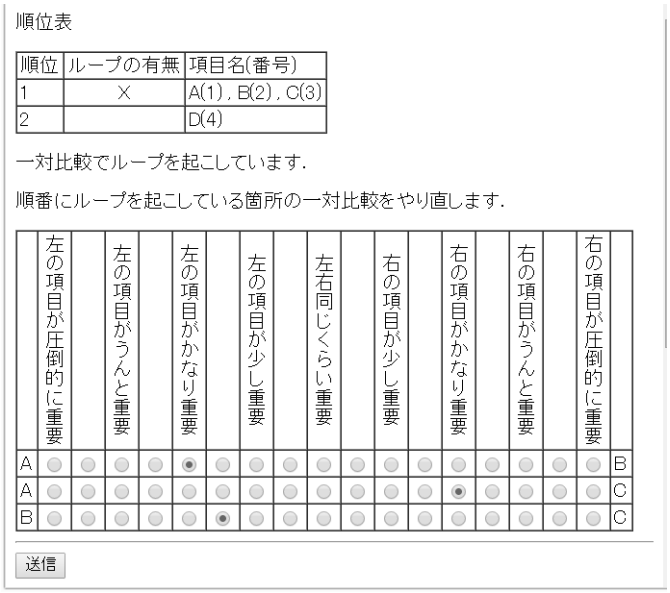

図 4: ループの指摘と修正

図 3 では，$A \succ B$ かつ $B \succ C$ で，$C \succ A$ であり，ループを起こしている．順位表の欄 (図 3 の左上) で，項目 (要素)A,B,C 間でループが発生していることを示しており，それを修正するためにその部分の一対比較のやり直し (図 4 の下部) を促している．

7.2 緩和な整合性制約のチェックと修正

(1) 7.1 節と同様に，上記 URL にアクセスし,「ループと緩和な整合性制約をチェック」,「理想点満足点を指定しない」を選ぶ（上記 URL の Web ページ中の一覧表の「E」を選択）．

(2) 一対比較の項目数を 5, 一対比較の項目名を「A」,「B」,「C」,「D」,「E」とした．

(3) 図 5 のような一対比較値の選択を行った．

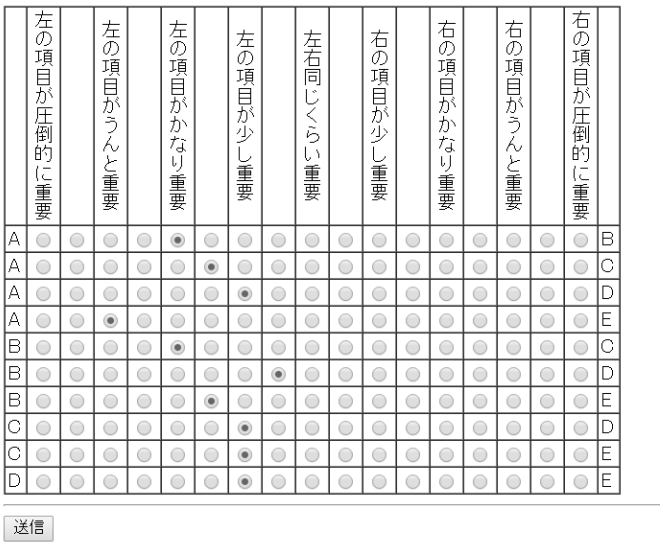

図 5: 一対比較値の入力

(4) この一対比較から $A \succ B \succ C \succ D \succ E$ という順序が定まる．

(5) 緩和な整合性制約のチェックを行う．図 6 は，図 5 のように $a_{AB} = 5, a_{BD} = 2, a_{AD} = 3$ と回答した例である．この 3 つの要素間では，次のように緩和な整合性制約を満たしていない．

$$a_{AD} \not\geq \max(a_{AB}, a_{BD}) \tag{26}$$

このアルゴリズムでは，順位差が大きい要素間の一対比較値を優先的に確定させているので，a_{AD} の値を固定することを第 1 の選択としている．$a_{AD} \leq a_{AB}$ であるので，a_{AB} を a_{AD} 以下にするために，A,B 間の元の一対比較値 (5) のラジオボタンをオレンジ色の背景色にし，一対比較値 1,2,3 が選択するように白の背景色で表示している．$a_{BD} \leq a_{AD}$ はす

順位表

順位	ループの有無	項目名(番号)
1		A(1)
2		B(2)
3		C(3)
4		D(4)
5		E(5)

緩和な整合性制約が満たされたいません

AとBとの一対比較(1行目)がAとDより強く(より左)設定されています．BがDより順位が上なので，より弱く(同じか右)設定して下さい．

赤の項目名の一対比較値は，白の選択肢から選べば，3行目を修正する必要はありません．黄色の部分にするときは，3行目も修正しなくてはなりません．

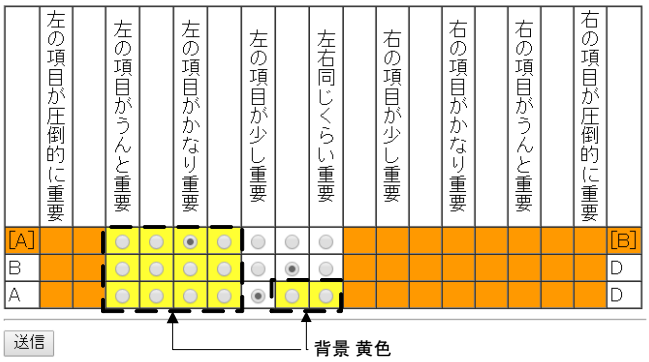

図 6: 緩和な整合性制約をチェック

でに満たされているが，B と D の一対比較値を変更するのであれば，同様に 1,2,3 の範囲が白の背景色である．

a_{AB} の値を変更しないときは，緩和な整合性制約を満たすため a_{AD} に値を 5 以上 ($a_{AD} \geq a_{AB}$) にしなくてはならない．また，a_{AD} の値は，$a_{AE} = 8$ より 8 以下に制約されるので，a_{AD} の値は 8 まで選択できるようにしている．

7.3 順序を指定して，緩和な整合性制約を満たす一対比較

(1) 7.1 節と同様に，上記 URL にアクセスし,「順序を指定」,「理想点満足点を指定しない」を選ぶ（上記 URL の Web ページ中の一覧表の「G」を選択）.

(2) 一対比較の項目数を 5, 一対比較の項目名を「A」,「B」,「C」,「D」,「E」とした.

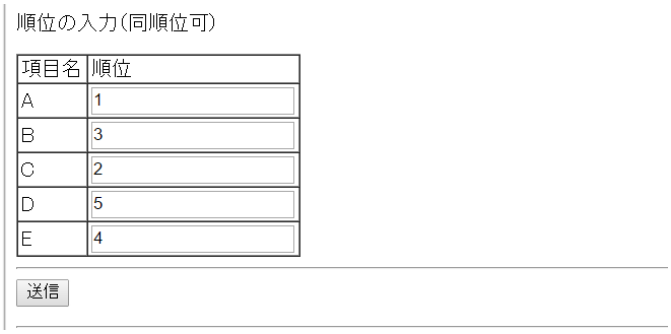

図 7: 順位を指定

(3) 図 7 のように評価項目 (要素) の順位を入力する.

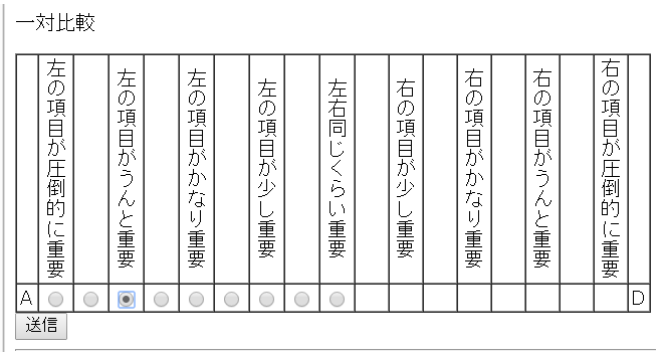

図 8: 1 位と 5 位の一対比較

(4) 順位差が大きいものから順に一対比較を行うので，1 位 (A) と 5 位 (D)（順位差 4）の一対比較を行う (図 8). 一対比較値 7 を設定した.

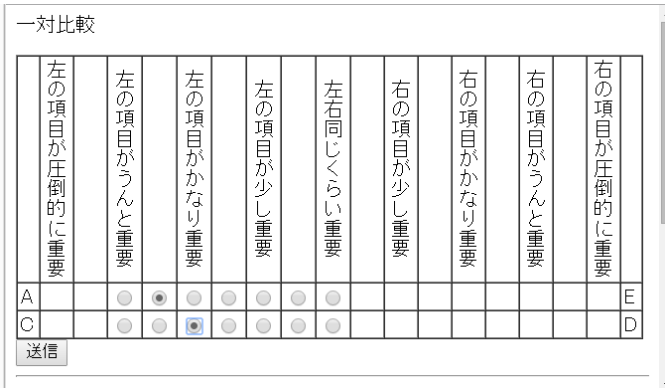

図 9: 順位差 3 の一対比較

(5) 順位差 3 の 1 位 (A) と 4 位 (E)，2 位 (C) と 5 位 (D) の一対比較を行う（図 9）．緩和な整合性制約により，ともに 1 位 (A) と 5 位 (D) の一対比較値 7 以下に制限される．それぞれ 6 と 5 を設定した．

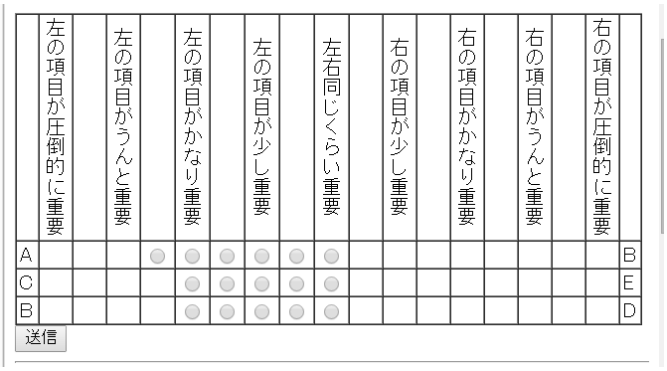

図 10: 順位差 2 の一対比較

(6) 順位差 2 の 1 位 (A) と 3 位 (B)，2 位 (C) と 4 位 (E)，3 位 (C) と 5 位 (E) の一対比較を行う（図 10）．緩和な整合性制約により，1 位 (A) と 3 位 (B) は，1 位 (A) と 4 位 (E) の一対比較値 6 以下に制限される．2 位 (C) と 4 位 (E) は，1 位 (A) と 4 位 (E)（一対比較値 6）と 2 位 (C) と 5 位 (D)（一対比較値 5）の両方に制約されるので，小さい方の 5 以下に制約される．3 位 (C) と 5 位 (E) は，2 位 (C) と 5 位 (D) の一対比較値 5 以下に制約される．

(7) 同様に順位差 1 の一対比較が行われる．

8. おわりに

ループの排除と緩和な整合性制約を満たす一対比較を行う手法を提案した．

ループの排除を行えば，確実に一対比較の整合性は高まる．また，緩和な整合性制約を満たすような修正も大きな効果がある．したがって，本手法はより正確な一対比較を行う手法である．

両方を満たす方法として，すでに一対比較が行われたものを修正する方法と，最初に要素間の順序を与え，緩和な整合性制約を満たす範囲で一対比較を行う方法を示した．しかし，順位が決定することは，おおよそのウエイトは決まってしまうので，本手法はより正確な評価値を求める手法である．

5 節では，一対比較をするとき，順位差が大きい項目間から一対比較をし，一対比較値の範囲（最大値を制限）を狭めていった．この手法とは逆に順位差が小さい項目間から一対比較を始め，一対比較値の範囲（最小値）を制限する方法もある．

参考文献

[1] T. L. Saaty: Bertoin, J. (1996), *The Analytic Hierarchy Process*, McGraw-Hill (1980).

[2] 田中浩光: AHP における一対比較値の緩和な整合性指標について (不確実性を含む意思決定の数理とその応用), 京都大学 数理解析研究所講究録, 1548, 122-129, 2007 (http://hdl.handle.net/2433/80827).

[3] 高萩栄一郎: 理想点，満足点を含めた一対比較について, 日本経営数学会誌, Vol.29, No.1, 43-59, 2008.

[4] 刀根薫: ゲーム感覚意思決定法 -AHP 入門, 日科技連, 1986.

[5] 中島信之: 一対比較行列の整合性の改善, 富大経済論集, 40-2, 391-406, 1994.

[6] P. T. Harker: *Alternative modes of questioning in the Analytic Hierarchy Process*, Math. Modelling, 9, 353-360, 1987.

高萩　栄一郎
専修大学商学部
〒 214-8580 川崎市多摩区東三田 2–1–1
E-mail: takahagi@isc.senshu-u.ac.jp