Multi-branch tree を用いた予算配分法において評価者の与える評価値の揺らぎが組織全体の予算配分に与える影響に関する考察

― Python による揺らぎ抽出アルゴリズムの構築とシミュレーションによる分析評価―

田 島 博 之

1．はじめに

例えば、予算総額があらかじめ決まっている場合、その中で複数の対象物にその予算をどのように割り当てるかということは重要な研究トピックである。また、複数の評価者が複数の対象物のそれぞれにどのように予算を配分するかというトピックについても数多く存在する。例えば市町村の予算配分等がこれに当たる。各部署には様々なプロジェクトがあり、各部署はそのプロジェクトごとに予算を見積り、必要となる予算額を毎年予算委員会に提出する。予算委員会ではこれら提案された各部署からの要求を総合的に検討して評価することで全体の予算を決定しているのである。[1][2][3]

このような中で Multi-branch tree を用いた予算配分法（以後「T-method」という。）が提案されている[4][5]。この方法は、予算配分に係るコストを最小化しつつ効率的な予算配分を実現することを可能にしている。さらにそれは、部門長が虚偽申告をする誘因を持たない Incentive compatibility に関する証明もされており多くのメリットがある。

T-method においては評価者をツリー（木）構造のノード（節）の部分に階層的に配置し、リーフ（葉）の部分に評価対象物を配置する。評価者に属するブランチ（枝）に従属する評価対象物を一つ選択して評価し、これらの評価値を統合させることで、全体の予算配分額を決定させることができる。また T-method は、評価コストを最小化させるため、個々の評価者が担当する対象に対する相対評価を自由度なしにウェイティングをさせている。これは評価ウェイトの小さな変更が、予算配分額の全体に敏感に影響を与えるといった、評価の頑健性に問題を来す可能性を示している。そこで彼らの提案アルゴリズムの実用化には、個々の評価者が評価に意図せず与える誤差に対する評価が不可欠であるとともに、それを回避する冗長性を確保した評価アルゴリズムの提案とその有効性の数値的検証が必要となる。

このような問題意識から、本論ではツリー構造の中から階層別に一人の評価者を選出し、その評価者が評価すべき対象物を与えた値に対する評価が、全体の予算配分額に与える影響について、Monte Carlo method[6][7]を用いたシミュレーションによって分析を行う。また、一人の評価者が与える一つの相対評価値であっても、全体に与える影響はツリー構造上の位置や相対評価値そのものなど様々な要因で規定されることが予測されるため、Stepwise method[8][9]を用いて多重回帰モデルによる検証を行う。

本論の流れとしては、まず第 2 章において具体的な計算手法を示すことによって T-method を分かりやすく紹介する。次に第 3 章では、ある一人の評価者が与える評価ウェイトに注目して、これが全体の予算配分に与える影響がツリー構造によってどのように変化するのかを統計的に分析することを可能にするための評価関数を定義し、ツリー構造の上位に存在する評価者の与える評価値ほど予算全体に大きな影響を与えることを示した。そして第 4 章では、Stepwise method を用いた多重回帰モデルを構築し、評価物の数とツリー構造上のブランチの数と階層の高さの関係を示した。最後に、これらの結果を最終章の第 5 章においてまとめた。

2. T-method による Algorithm

本節では T-method について説明する。正確な解法の定義は文献[4]の第 2 節にあるが、ここでは本論を理解するために、単純な Sample を使って異なる説明を行う。

Figure1.では$e_i(i = 1,2,\dots,9)$が評価者を表し、$p_j(j = 1,2,\dots,10)$を評価対象物として表す。各評価者は、直下にある1本のブランチに対して、そのブランチ以下に繋がる評価対象物の中から1つを選択し、相対評価を行う。例えば、評価者e_1は左のブランチによって、e_2に従属している$\{p_1, p_2, p_3, p_4\}$から1つ。さらに右のブランチからは、e_3に従属している$\{p_5, p_6, p_7, p_8, p_9, p_{10}\}$から1つの評価対象物を選択し、相対評価値を与える。同様に評価者e_2は、左のブランチからe_4に従属する$\{p_1, p_2\}$から1つ、右のブランチからe_5に従属する$\{p_3, p_4\}$から1つを選択し、それぞれに相対評価値を与える。

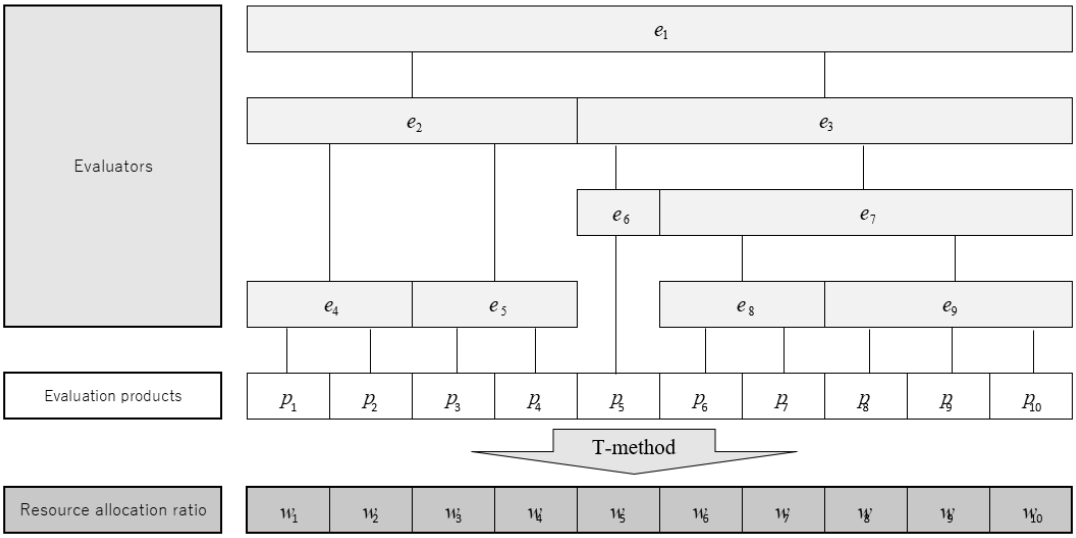

Figure1. Sample Tree

Figure1.に示されるツリー構造における各評価者と評価対象物の選択肢の関係を以下に示す。

$e_1: \{\{p_1, p_2, p_3, p_4\}, \{p_5, p_6, p_7, p_8, p_9, p_{10}\}\}$
$e_2: \{\{p_1, p_2\}, \{p_3, p_4\}\}$
$e_3: \{\{p_5\}, \{p_6, p_7, p_8, p_9, p_{10}\}\}$
$e_4: \{\{p_1\}, \{p_2\}\}$
$e_5: \{\{p_3\}, \{p_4\}\}$
$e_6: \{\{p_5\}\}$
$e_7: \{\{p_6, p_7\}, \{p_8, p_9, p_{10}\}\}$
$e_8: \{\{p_6\}, \{p_7\}\}$
$e_9: \{\{p_8\}, \{p_9\}, \{p_{10}\}\}$

評価者e_i が評価対象物p_jに対して与える相対評価値を$v_{(i,j)}$とする。上の選択肢から各評価者が選択した評価対象物を Table1.に示す。

Table 1. Evaluation table

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	Selected evaluation value
e_1	★								★		$v_{(1,1)}, v_{(1,9)}$
e_2	★		★								$v_{(2,1)}, v_{(2,3)}$
e_3					★	★					$v_{(3,5)}, v_{(3,7)}$
e_4	★	★									$v_{(4,1)}, v_{(4,2)}$
e_5			★	★							$v_{(5,3)}, v_{(5,4)}$
e_6					★						$v_{(6,5)}$
e_7						★		★			$v_{(7,6)}, v_{(7,8)}$
e_8						★	★				$v_{(8,6)}, v_{(8,7)}$
e_9								★	★	★	$v_{(9,8)}, v_{(9,9)}, v_{(9,10)}$

T-method

W	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}

Table1.では評価者が選択した評価物を表している。そこで、この評価物に対する評価値$v_{(i,j)}$に従って評価対象物$p_i(i = 1,2, \dots,10)$に対する、予算配分$W = (w_1, w_2, \dots, w_{10})$を求める演算を行う。

初めに、$p_i(i = 1,2, \dots,10)$に対して T-method のルール[1]に従い、それぞれの評価者が与える相対評価値$v_{(i,j)}$から以下のようにウェイト$s_i(i = 1,2, \dots,10)$を算出する。

$$p_1: s_1 = v_{(1,1)}$$
$$p_2: s_2 = \frac{v_{(1,1)} \cdot v_{(4,2)}}{v_{(4,1)}}$$
$$p_3: s_3 = \frac{v_{(1,1)} \cdot v_{(2,3)}}{v_{(2,1)}}$$
$$p_4: s_4 = \frac{v_{(1,1)} \cdot v_{(2,3)} \cdot v_{(5,4)}}{v_{(2,1)} \cdot v_{(5,3)}}$$
$$p_5: s_5 = \frac{v_{(1,9)} \cdot v_{(9,8)} \cdot v_{(7,6)} \cdot v_{(8,7)} \cdot v_{(3,5)}}{v_{(9,9)} \cdot v_{(7,8)} \cdot v_{(8,6)} \cdot v_{(3,7)}}$$
$$p_6: s_6 = \frac{v_{(1,9)} \cdot v_{(9,8)} \cdot v_{(7,6)}}{v_{(9,9)} \cdot v_{(7,8)}}$$
$$p_7: s_7 = \frac{v_{(1,9)} \cdot v_{(9,8)} \cdot v_{(7,6)} \cdot v_{(8,7)}}{v_{(9,9)} \cdot v_{(7,8)} \cdot v_{(8,6)}}$$
$$p_8: s_8 = \frac{v_{(1,9)} \cdot v_{(9,8)}}{v_{(9,9)}}$$
$$p_9: s_9 = v_{(1,9)}$$
$$p_{10}: s_{10} = \frac{v_{(1,9)} \cdot v_{(9,10)}}{v_{(9,9)}}$$

次に S=$\sum_{j=1}^{10} s_j$による正規化を行い以下に予算配分を決定する。

$$w_j = \frac{s_j}{S}(j = 1,2, \dots,10)$$

本章では T-method により、多階層からなる 9 人の評価者が相対評価を 9 回行うことで予算配分を行う例を紹介した。次章では新たなツリー構造を設定し、T-method における1人の評価者の評価値の揺らぎが、全体の予算配分の変化に与える影響についてシミュレーションを行う。

3. 一人の評価の変動が全体の予算配分に与える影響

本章ではT-method のツリーにおける階層数と1つのノードが持つブランチの数に応じて、「一人の評価者が与える相対評価が全体の予算配分にどの程度影響を与えるか」について統計的に分析する。ここでは、ツリーを$T(h,b)$と表す。hはツリーの階層数であり、bは1つのノードが持つブランチの数である。分析単純化のために全てのノードが持つブランチを同数と定める。T-method ではリーフ以外のノードに対して評価者を配置し、リーフには予算配分の対象物を配置する。このとき評価者の総数は$n=\sum_{i=1}^{h} b^{i-1}$である。また、評価対象物の総数は $m=b^{h-1}$である。評価者の集合を$E=\{e_i\}(i=1,2,\dots,n)$とする。

3.1 関数の定義

下位に評価者がいない評価者は、b個の評価対象物に対して相対評価値を与える。また、下位に評価者がいる場合は、下位の各評価者に属する評価対象物から1つを選択し合計b個の相対評価値を与える。なお評価対象物が複数存在する場合は、1番左下の評価対象物を評価するルールを与える。$T(h,b)$に対する第i番目の評価者が第j番目の評価対象物に対して与える相対評価値は一様分布関数$U(\alpha,\beta)=\{\delta|\alpha\le\delta\le\beta\}$を用いて$v_{(i,j)}\sim U(1.0,10.0)$と仮定する。以上を前提として評価者に対する評価値をランダムに与え予算配分を算出する。その後、一人の評価者を抽出し評価値を新たに与え再度予算配分を算出する。

初めに全ての評価者$E=\{e_i\}(i=1,2,\dots,n)$がルールに従い、それぞれb個の評価対象物に対して相対評価値を与える。また、全ての相対評価値を$V=\{v_{(i,j)}\}(1\le i\le n,1\le j\le b)$とする。

ここで揺らぎを与える評価者を各階層の一番左端から1人選択する。ここで選択した評価者を$e_k(k=1,2,\dots,n)$とするとき、e_kのp_jに対する相対評価値を$v_{(k,j)}(1\le j\le b)$と表す。

次に揺らぎを与える行為として、評価者e_kは、自身が選択した評価対象物p_jに対して再度新たな相対評価値$v'_{(k,j)}$を与える。この時、評価者e_kの与えた相対評価値の変量をxとして定義する。

$$x=\sqrt{\sum_{j=1}^{b}\left(v_{(k,j)}-v'_{(k,j)}\right)^2}$$

次に、Vを元に T-method から得られた予算配分を$W=(w_1,w_2,\dots,w_m)$とする。

また、評価者e_kが$v_{(k,j)}$から$v'_{(k,j)}$へと相対評価値を入れ替えたことによって全ての相対評価値もVからV'に変わる。V'を元に再度予算配分を行った結果を$W'=(w'_1,w'_2,\dots,w'_m)$とする。

この時の予算配分の変化、つまりWからW'に対する揺らぎの変量を以下にyとして定義する。

$$y=\sqrt{\sum_{j=1}^{m}\left(w_j-w'_j\right)^2}$$

3.2 誤差量の推定

前節で定義したx,yに試行回数μをインデックスとしてあたえる。本節では前節に従い$T(3,3)$モデルにおいて、$(x_\mu,y_\mu)(\mu=1,2,\dots,10000)$を算出するシミュレーションを行う。なお、本研究における、シミュレーションは、Python（ver.3.8.5）に従ってコーディングを行っている。各モジュールは本論末尾の Appendix を参照されたい。なお、ここでは T-method における評価者は 1 番左下の評価対象物を選択するルールを与える。また、揺らぎを与える評価者は最上位層（Level1）、中位層（Level2）、下位層（Level3）の左端から選出する 3 パターンで行う。

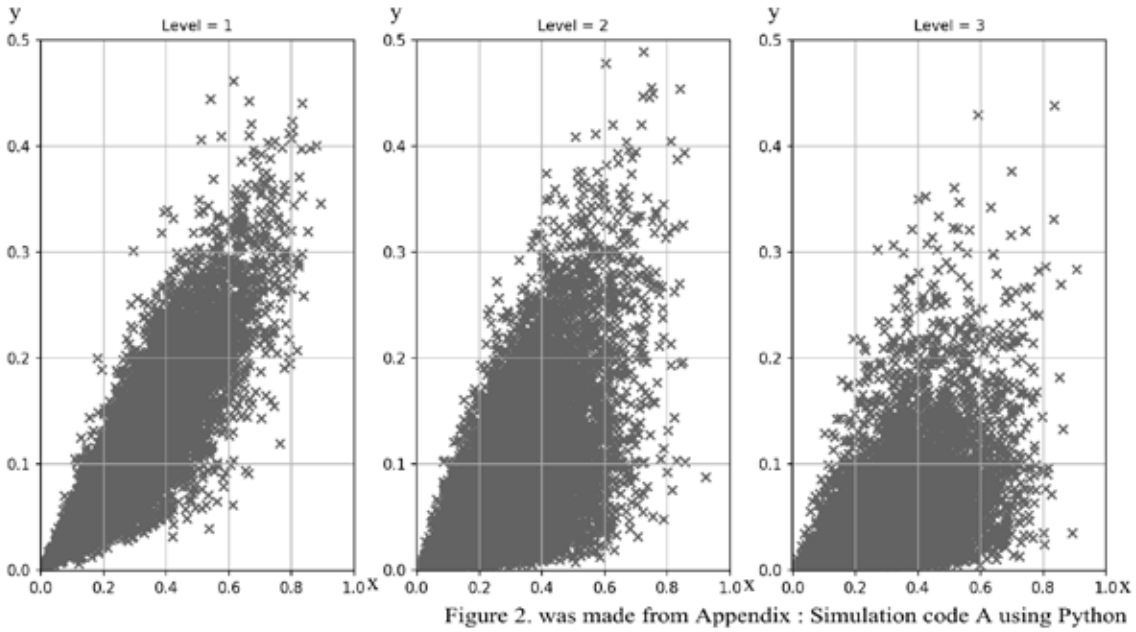

Figure 2. (x_μ, y_μ) **の散布図**

相関係数はいずれも p<0.01 の水準で有意であり、Level1 から順に 0.87、0.63、0.51 となる。Figure2.の Leve1 からは、xの変量が高いほどyの値も高くなっていることがわかる。その一方で、Level2 の中階層から Level3 の低階層に下がるにしたがって、xの変量が増加するにつれyの変量に与える影響が減少していることが読み取れる。

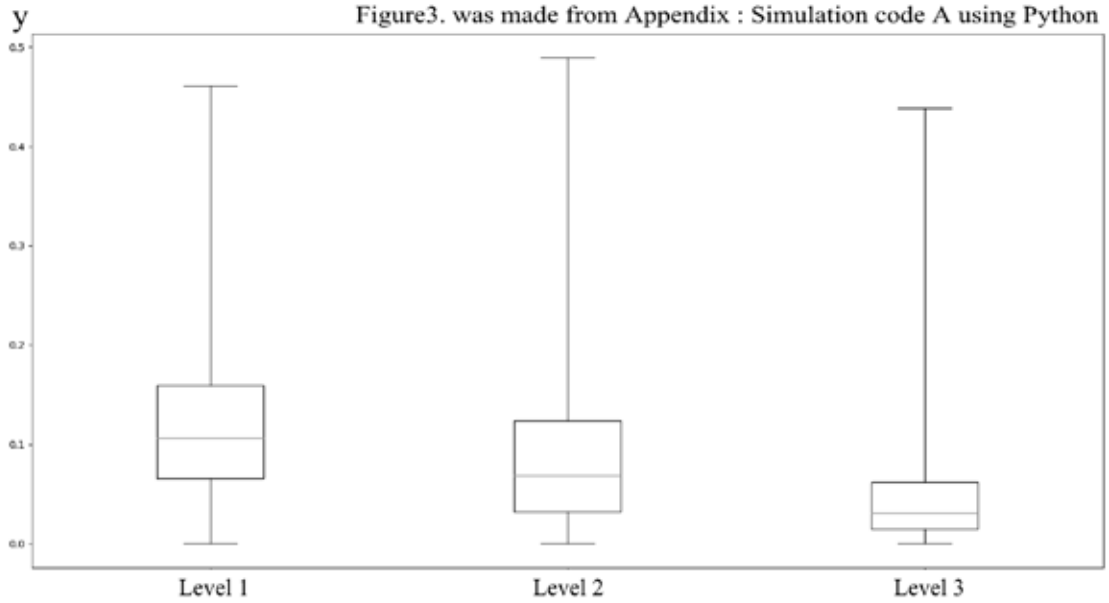

Figure 3. Box plot によるy**の分布**

さらに、yにおける分布の様相を詳しく調べるために Box plot を Figure3.に表す。Box plot では分布の最小値、第 1 四分位点、中央値、第 3 四分位点、最大値を見ることが出来る。いずれの Box plot においても上髭が長くあるが、中央値が Level1 で 0.1067、続いて Level2 が 0.0684、Level3 が 0.0308 となっており、第 1 四分位点と第 3 四分位点の位置からも、揺らぎを与える評価者の階層が

上位であるほど、変量xから多くの影響を受けyの分布が高くなっている。

このように階層が上位にある評価者の変化量xの値は変化量yに強い影響を与え、全体の予算配分の変化量を上げていることが明らかになった。

4. 全体の誤差量を最小にするツリー構造の推定

前章では一人の評価者の相対評価値を変化させたとき、評価者の階層が上位にあるほど(x, y)の相関は強くなり、変化量yの値が増加することが分かった。ここでは階層数を$2 \leq h \leq 6$、またブランチの数を$2 \leq b \leq 6$と範囲を限定した$T(h, b)$に対して、Stepwise method を用いて多重回帰モデルを構築する。これによって同定されるモデルは、評価対象物が$2^2 = 4$個から$6^6 = 46656$個までを対象とすることができる。$T(h, b)$において最上位層一番左の評価者の相対評価値を変化させ、変量yの値を計算する。 具体的な(h, b)の組み合わせは 25 通りあるので、 それぞれ $\mu = 1,2, \ldots,$ 40000 のシミュレーションを行い、40,000 回×25 通り=1,000,000 回の試行を行う。

これらの 1,000,000 組のデータをもとに、Stepwise method を用いて多重回帰モデルを構築する。

本モデルではh, b、交互作用項である$h \cdot b$からなる 3 つの因子を説明変数として用意した。

また、目的変数を変化量yとした。この条件のもとに、Stepwise method を用いて多重回帰モデルを求める。計算の結果を Table2.に示す。各因子のP値は、いずれも 0 となり 3 つの因子が、いずれも有効であることが示された。補正 R2 の値は 0.358、重相関 R の値は 0.641 と高い適合率が確かめられた。

Table 2. Stepwise method による多重回帰モデル

Regression statistics			Coefficient	Standard error	t	P-value
Multiple correlation R	0.641071211	Intercept	0.315280948	0.000560452	562.5471342	0
Multiple decision R2	0.410972298	h	-0.037943349	0.000132099	-287.2339759	0
Adjusted R-square R2	0.410970531	b	-0.028648106	0.000132096	-216.8731134	0
Standard error	0.062287664	h·b	0.001943538	3.11352E-05	62.42258611	0
Number of samples	1,000,000					

Table 2. was made from Appendix : Simulation code B using Python

Stepwise method によって得られた多重回帰モデルを次式に与える。

$$y = -0.0379h - 0.0286b + 0.00194h \cdot b + 0.315$$

(h, b)の値が既知であるとき、この重回帰式を使うことによって最上位層にいる一人の評価者の評価値データの揺らぎに対するyの変化量を予測することが可能となる。

次に、評価対象物の総数をm、ツリーの高さをhとしたとき、ブランチの数$b = m^{\frac{1}{h}}$と表せる。これを上式に代入することで評価対象物の個数mと、階層の高さhを変数とした関数$f(m, h)$とする。

$$f(m, h) = -0.0379h - 0.0286m^{\frac{1}{h}} + 0.00194h \cdot m^{\frac{1}{h}} + 0.315$$

これによって、評価対象物の個数がmと決定しているとき、$2 \leq h \leq 6$の範囲において$f(m, h)$を最小とするhの値を選択することによって、最上位層の評価者の揺らぎxが全体の予算配分に対して最も影響を及ぼさないツリー構造を決定することが可能となる。

5．まとめ

複数の対象物に対して予算を配分する評価の方法については、これまでは Analytic Hierarchy Process（AHP）[10]~[14]が優秀な方法として活用されてきた。また、AHP では複数の評価者による方法も研究されている[15]。いずれのモデルにおいても、評価者が評価対象物を一対比較することで評価対象物に対する評価値を算出することになる。しかしながら、AHP には従来から評価対象物の増加に伴い比較評価を行う回数が飛躍的に増加してしまうという問題があった。その点 T-method は、評価者をツリー構造に配置して各評価者が自身に属するブランチの数だけ評価すればよく、評価にかかるコストの低減に大いに貢献している。しかしながら、一人の与える相対評価の値が全体の予算配分にどの程度の影響を与えるかについての言及はされていなかった。

そこで本論では、独自の評価関数を 2 つ定義し、Monte Carlo method を用いたシミュレーションによる検証を行った。結果として、評価者の評価値の変動がモデルの上位階層にあるほど、予算全体に与える変動が大きいことを明らかにした。これは、T-method における評価すべきツリー構造における、それぞれの階層で評価者の評価値が適切に予算配分されるということであり、組織における上位者の責任の重さの数量化に成功したと言える。

次に、$2 \leq b \leq 6$、$2 \leq h \leq 6$となる範囲における$T(h,b)$から、最上位階層から一人の評価者を抽出し、その揺らぎを求めるシミュレーションを行った。ここで得られた 1,000,000 のデータセットから Stepwise method を用いて重相関の R 値が 0.641 となる多重回帰モデルを得た。 これによって、T-method を用いる際の最上位者の揺らぎから生じる全体の予算配分全体に与える影響を数量化して予測することを可能にした。さらに、回帰モデルを構築することによって、評価対象物の個数が増大したときに、T-method を用いる際の理想的なブランチ数と階層数を予測可能にした。

今後は、本研究を基盤とした冗長戦略を考察するとともに、T-method をより現実的なモデルにするために、シミュレーションに利用する乱数関数の選択や、事例に適したモデルに対する考察が必要になると考える。

謝辞

本論を執筆するにあたって、創価大学経営学部　岡田勇先生には、数多くの有益なご助言を賜りました。この場をお借りしまして、心より感謝の意を表します。

参考文献

[1] Melkers, J.; Willoughby, K. Models of Performance-Measurement Use in Local Governments: Understanding Budgeting, Communication, and Lasting Effects. *Publ. Adm. Rev.* **2005**, 65, 180-190.

[2] Mousavi-Nasab, S.; Safari, J.; Hafezalkotob, A. Resource allocation based on overall equipment effectiveness using cooperative game. *Int. J. Cybern. Syst. Manag. Sci.* **2019**, 49, 819-834.

[3] Nasseri, S.H.; Baghban1, A.; Mahdavi, I. A new approach for solving fuzzy multi-objective quadratic programming of water resource allocation problem. *J. Ind. Eng. Manag.* **2019**, 6, 78-102.

[4] Oyamaguchi, N.; Tajima, H.; Okada, I. Model of Multi-branch Trees for Efficient Resource Allocation. *Algorithms* **2020**, *13*, 55.

[5] Oyamaguchi, N.; Tajima, H.; Okada, I. Tournament Method Using a Tree Structure to Resolve Budget Conicts. In *Intelligent Decision Technologies 2019, Smart Innovation, Systems and Technologies, vol193*; Springer, Singapore, **2020**, 525-532.

[6] Cameron.B Edward P, Daniel W, Simon L, Peter I. C, Philipp R,Stephen Tavener, Diego Perez, Spyridon Samothrakis and Simon Colton A Survey of Monte Carlo Tree Search Methods *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES*, VOL. 4, NO. 1, MARCH, **2012,**1-43

[7] Robert S.; Jian-Sheng W., Nonuniversal Critical Dynamics in Monte Carlo Simulation, *Physical Review Letters* 58(2),February **1987**,86-88

[8] Bruce T. Stepwise Regression and Stepwise Discriminant Analysis Need Not Apply here: A Guidelines *Editorial Educational and Psychological Measurement* Volume: 55 issue: 4, **1995,** 525-534

[9] Michael C.; A. Afifi, Comparison of Stopping Rules in Forward Stepwise Discriminant Analysis, *Journal of the American Statistical Association*, Vol. 74, No. 368. Dec., **1979**, pp. 777-785.

[10] Thomas L. Saaty Decision making with the analytic hierarchy process *Int. J. Services Sciences*, Vol. 1, No. 1, **2008,** 83-98

[11] JOSÉ A.,TERESA L. CONSISTENCY IN THE ANALYTIC HIERARCHY PROCESS:A NEW APPROACH, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* Vol. 14, No. 4, **2006**, 445-459

[12] Thomas L. Saaty How to make a decision:The Analytic Hierarchy Process, *European Journal of Operational Research 48*, **1990**, 9-26

[13] Jiří F.Aleš K. Judgment scales and consistency measure in AHP, *ScienceDirect Procedia Economics and Finance* 12 , **2014** , 164- 173

[14] Evangelos T.,Stuart H. M. USING THE ANALYTIC HIERARCHY PROCESS FOR DECISION MAKING IN ENGINEERING APPLICATIONS: SOME CHALLENGES,Inter'l, *Journal of Industrial Engineering: Applications and Practice*, Vol. 2, No. 1, **1995**, 35-44

[15] María Teresa Escobar & José María Moreno-jiménez Aggregation of Individual Preference Structures in Ahp-Group Decision Making Group *Decision and Negotiation* volume 16, **2007,** 287-301

Appendix : Simulation code A using Python (Python ver.3.8.5)

```
import numpy as np
import pandas as pd
import random as r
r.seed(2)
import matplotlib.pyplot as plt
import seaborn as sns

class Global():
    const = {
        'level' :     3, # No. levels of a tree
        'branch':     3, # No. branches in each assessor
        'N'      :10000, # No. trials
    }

def calError(org,rev):
    n = float(len(org))
    total = 0.
    for ele1,ele2 in zip(org,rev):
        total += (ele2 - ele1) ** 2
        tt=total**0.5
    return tt

def makeOnes():
    branch = Global.const['branch']
    tmp = []
    for b in range(branch):
        tmp.append(r.uniform(1.,10.))
    total = sum(tmp)
    vOne = []
    for ele in tmp:
        vOne.append(ele/total)
    return vOne
```

```
def makeData():
    level = Global.const['level']
    branch = Global.const['branch']
    vAll = []
    for l in range(level):
        assessors = branch ** l
        vLevel = []
        for assessor in range(assessors):
            vLevel.append(makeOnes())
        vAll.append(vLevel)
    return(vAll)

def makeRev(data,target):
    level = Global.const['level']
    branch = Global.const['branch']
    rev = []
    for ele1 in data:
        rev2 = []
        for ele2 in ele1:
            rev3 = []
            for ele3 in ele2:
                rev3.append(ele3)
            rev2.append(rev3)
        rev.append(rev2)
    rev[target][0] = makeOnes()
    alpha = calError(data[target][0],rev[target][0])
    return rev,alpha

def calWeight(value):
    level = Global.const['level']
    branch = Global.const['branch']
    num = branch ** level
    weight = [0. for i in range(num)]
    for b in range(branch):
        weight[b * (branch ** (level - 1))] = value[0][0][b]
```

```
    for l in range(1,level):
        for n in range(branch ** l):
            p = n * (branch ** (level - l))
            q = branch ** (level - l - 1)
            for pos in range(1,branch):
                weight[p+q*pos] = weight[p] * value[l][n][pos] / value[l][n][0]
    total = sum(weight)
    ans = []
    for ele in weight:
        ans.append(ele / total)
    return ans

########################

if __name__ == "__main__":
    N = Global.const['N']
    Level = Global.const['level']
    Branch = Global.const['branch']

    fig=plt.figure(figsize=(10,3), dpi=150)

    dataBP = []
    for level in range(Level):
        plt.subplot(1,Level,level+1)
        #plt.ylim(-.001,.5)
        #plt.xlim(-.001,1.0)

        X = []
        Y = []
        for i in range(N):
            orgData = makeData()
            revData, alpha = makeRev(orgData,level)
            X.append(alpha)
            Y.append(calError(calWeight(orgData),calWeight(revData)))
        dataBP.append(Y)
```

```
            coef = np.corrcoef(X, Y)    # Find the correlation coefficient
            print(coef)
            print("----------------------------------------")

            plt.scatter(X,Y,marker="x")
            plt.title('Level = ' + str(level+1),fontsize=10)
            plt.grid(True)
        plt.savefig('Fig1.png', format='png', dpi=150)
        plt.show()

        plt.boxplot(dataBP, labels=range(Level), whis="range")
        plt.savefig('Fig2a.png', format='png', dpi=150)
        plt.show()

        plt.violinplot(dataBP,showmeans=False,showmedians=True)
        plt.savefig('Fig2b.png', format='png', dpi=150)
        plt.show()
a=np.array(dataBP)
print(a)
print("----------------------------------------------------")
l1=np.median(a[0])
l2=np.median(a[1])
l3=np.median(a[2])
print(l1,l2,l3)
print("----------------------------------------------------")
f = open('test.txt', 'w')
for i in range(10000):
    strs=str(X[i])+'\t'+str(a[0,i])+'\t'+str(a[1,i])+'\t'+str(a[2][i])+'\n'
    f.writelines(strs)
f.close()
```

Appendix : Simulation code B using Python (Python ver.3.8.5)

```
import pandas as pd
import numpy as np
import random as r
import matplotlib.pyplot as plt
r.seed(2)
level_min,level_max=2,6
branch_min,branch_max=2,6
number_trials=40000

f = open('dataBP_6x6(40000(seed2)).txt', 'w')
f.writelines('L_size'+'\t'+'B_size'+'\t'+'level'+'\t'+'dataBP[i]'+'\n')
print('L_size','\t','B_size','\t','level')
for level_size in range(level_min,level_max+1):
   const = {}
   const['N']=number_trials            # No. trials
   const['level']=level_size           # No. levels of a tree
   for branch_size in range(branch_min,branch_max+1):
      const['branch']=branch_size      # No. branches in each assessor

      def calError(org,rev):
         n = float(len(org))
         total = 0.
         for ele1,ele2 in zip(org,rev):
            total += (ele2 - ele1) ** 2
         tt=(total)**0.5/n
         return tt
      def makeOnes():
            branch = const['branch']
            tmp = []
            for b in range(branch):
                  tmp.append(r.uniform(1.,10.))
            total = sum(tmp)
            vOne = []
            for ele in tmp:
```

```
            vOne.append(ele/total)
        return vOne
    def makeData():
        level = const['level']
        branch = const['branch']
        vAll = []
        for l in range(level):
            assessors = branch ** l
            vLevel = []
            for assessor in range(assessors):
                vLevel.append(makeOnes())
            vAll.append(vLevel)
        return(vAll)

    def makeRev(data,target):
        level = const['level']
        branch = const['branch']
        rev = []
        for ele1 in data:
            rev2 = []
            for ele2 in ele1:
                rev3 = []
                for ele3 in ele2:
                    rev3.append(ele3)
                rev2.append(rev3)
            rev.append(rev2)
        rev[target][0] = makeOnes()
        alpha = calError(data[target][0],rev[target][0])
        return rev,alpha
    def calWeight(value):
        level = const['level']
        branch = const['branch']
        num = branch ** level
        weight = [0. for i in range(num)]
        for b in range(branch):
```

```
                weight[b * (branch ** (level - 1))] = value[0][0][b]
            for l in range(1,level):
                for n in range(branch ** l):
                    p = n * (branch ** (level - l))
                    q = branch ** (level - l - 1)
                    for pos in range(1,branch):
                      weight[p+q*pos] = weight[p] * value[l][n][pos] / value[l][n][0]
            total = sum(weight)
            ans = []
            for ele in weight:
                ans.append(ele / total)
            return ans
        N = const['N']
        Level = const['level']
        Branch = const['branch']
        dataBP = []
        for level in range(1):
          X = []
          Y = []
          for i in range(N):
            orgData = makeData()
            revData, alpha = makeRev(orgData,level)
            X.append(alpha)
            Y.append(calError(calWeight(orgData),calWeight(revData)))
          dataBP.append(Y)

        for i in range(1):
          print(level_size,'\t',branch_size,'\t',i+1,'\n',dataBP[i],'\n')
          print(level_size,'\t',branch_size,'\t',i+1,'\n')
          for j in range(number_trials):
            strs=str(level_size)+'\t'+str(branch_size)+'\t'+str(i+1)+'\t'+str(dataBP[i][j])+'\n'
            f.writelines(strs)
    f.close()
```