

Appendix: Revised Simos

A.1 Introduction

An important and challenging step in the solution of a decision making problem is the elicitation of weights. In the examples presented in the previous Chapters, the weights of the criteria were inputs to the problem. However, one of the most important steps that the decision maker performs during the solution of a decision making problem with an MCDA method is the assessment of the criteria weights. Various methods have been proposed for this task. These methods can be mainly categorized in two major classes, the direct assessment ones and the indirect ones. The indirect assessment methods have been used in most applications of MCDA methods due to their simplicity and realism. One of the most widely-used methods is the one proposed by Simos [2, 3]. This method is a typical indirect assessment method that has been widely-used in decision making problems since it is relatively easy for decision makers to express their preferences. The elicitation of the criteria weights is performed by asking decision makers to express the relative importance of the criteria, through the arrangement of criteria cards, from the least to the most important one. The method was later extended by Figuera and Roy [1] in order to address certain robustness issues of the original method. The revised Simos method is widely-used in decision making problems for estimating the criteria weights.

A.2 Methodology

The decision maker is given a set of cards. The name of each criterion is written on a card. The decision maker uses the cards in order to rank the criteria from the least important to the most important. The first criterion in the ranking is the least important and the last criterion is the most important. If some criteria have the same importance for the decision maker, he/she can place them together in the same

position. Therefore, a complete pre-order of the whole n criteria is obtained. The number of ranks is \tilde{n} , where $1 \leq \tilde{n} \leq n$ (since some of the cards can be placed in the same rank).

The decision maker has also a set of white cards. The importance of two successive criteria (or two successive subsets of ex aequo criteria in case two or more cards have been placed together) in the ranking can be more or less close. In order to depict this smaller or larger difference of the importance of successive criteria, the decision maker introduces white cards between two successive cards. The more the number of white cards between two successive criteria, the greater the difference between their importance. If no white card is placed between two successive ranks, then the difference between the weights of the criteria in these two successive ranks can be chosen as the unit, u , for measuring the intervals between weights. Hence, if one white card is placed between two successive ranks, then there is a difference of $2u$ between the weights of the criteria in these two successive ranks. Finally, the decision maker should state how many times the last criterion is more important than the first one. This ratio is denoted by the parameter z .

The revised Simos method consists of two steps:

1. Calculate the non-normalized weights $k = (k_1, k_2, \dots, k_{\tilde{n}})$: Let e_r' be the number of white cards between the ranks r and $r + 1$.

$$\begin{cases} e_r = e_r' + 1, \forall r = 1, 2, \dots, \tilde{n} - 1, e_1' = 0 \\ u = \frac{z-1}{\sum_{r=1}^{\tilde{n}-1} e_i} \end{cases} \quad (\text{A.1})$$

Next, we can calculate the non-normalized weights k as follows:

$$k_r = 1 + u \sum_{i=0}^{r-1} e_i, e_0 = 0 \quad (\text{A.2})$$

2. Calculate the normalized weights $k^* = (k_1^*, k_2^*, \dots, k_{\tilde{n}}^*)$: Let c_i be the number of cards in each ranking i , where $1 \leq i \leq \tilde{n}$. Then, the normalized weights k^* are calculated as follows:

$$k_r^* = \frac{100}{\sum_{i=1}^{\tilde{n}} c_i k_i} k_r \quad (\text{A.3})$$

Figuera and Roy [1] also presented a method to eliminate some of the decimal figures from the normalized weights.

A.2.1 Numerical Example

Let us consider a set of eight criteria: $\{a, b, c, d, e, f, g, h\}$. Let us also suppose that the decision maker groups together cards associated to the criteria having the same importance into four subsets of ex aequo, inserting also three white cards between some of the successive ranks. Table A.1 presents the ranking of these cards.

Let us suppose that $z = 6.5$, i.e., the decision maker states that the last subset is 6.5 times more important than the first one. We start by calculating vector e and parameter u according to Equation (A.1)

$$e = [1 \ 2 \ 3 \ 1]$$

$$u = 1.375$$

Then, we can calculate the non-normalized weights k according to Equation (A.2)

$$k = [1 \ 2.375 \ 5.125 \ 9.25]$$

Finally, we calculate the normalized weights k^* according to Equation (A.3)

$$k^* = [2.61437908 \ 6.20915033 \ 13.39869281 \ 24.18300654]$$

Therefore, the criteria weights are presented in Table A.2.

Table A.1 Ranking of the criteria using cards

Rank	Subset of ex aequo
1	$\{b, d\}$
2	$\{c\}$
3	White card
4	$\{e, f, h\}$
5	White card
6	White card
7	$\{a, g\}$

Table A.2 Criteria weights

Criterion	Weight
a	24.1830065359
b	2.61437908497
c	6.2091503268
d	2.61437908497
e	13.3986928105
f	13.3986928105
g	24.1830065359
h	13.3986928105
Total	100

A.2.2 Python Implementation

The file *RevisedSimos.py* includes a Python implementation of the revised Simos method. The input variables are array *subsets* (the ranks of the criteria, lines 8–9), and *z* (the parameter *z*, line 12). In lines 15–22, the number of positions with non-white cards, and vector *c* are calculated. Then, we calculate parameter *u* (line 25). Next, we calculate vector *e* (lines 29–34). The non-normalized weights are computed in lines 37–41, while the normalized weights are computed in lines 44–46. Finally, the criteria weights are printed in lines 49–57.

```

1.  # Filename: RevisedSimos.py
2.  # Description: Revised Simos method
3.  # Authors: Papathanasiou, J. & Ploskas, N.
4.
5.  from numpy import *
6.
7.  # placement of cards ('w' represents a white card)
8.  subsets = array([[ 'b', 'd'], [ 'c'], [ 'w'],
9.                  [ 'e', 'f', 'h'], [ 'w'], [ 'w'], [ 'a', 'g']])
10.
11. # parameter z
12. z = 6.5
13.
14. # calculate number of cards, positions, and vector c
15. noOfcards = 0
16. positions = 0
17. c = []
18. for i in range(subsets.shape[0]):
19.     if subsets[i][0] != 'w':
20.         noOfcards = noOfcards + len(subsets[i][:])
21.         positions = positions + 1
22.         c.append(len(subsets[i][:]))
23.
24. # calculate u
25. U = round((z - 1) / positions, 6)
26.

```

```

27. # calculate vector e
28. e = ones(positions)
29. counter = -1
30. for i in range(subsets.shape[0]):
31.     if subsets[i][0] != 'w':
32.         counter = counter + 1
33.     else:
34.         e[counter] = e[counter] + 1
35.
36. # calculate the non-normalized weights k
37. k = ones(positions)
38. totalk = k[0] * c[0]
39. for i in range(1, positions):
40.     k[i] = 1 + U * sum(e[0:i])
41.     totalk = totalk + k[i] * c[i]
42.
43. # calculate the normalized weights
44. normalizedWeights = zeros(positions)
45. for i in range(0, positions):
46.     normalizedWeights[i] = (100 / totalk) * k[i]
47.
48. # print the criteria weights
49. counter = -1
50. for i in range(subsets.shape[0]):
51.     if subsets[i][0] != 'w':
52.         counter = counter + 1
53.     else:
54.         continue
55.     for j in range(len(subsets[i][:])):
56.         print("Weight of criterion ", subsets[i][j],
57.             " = ", normalizedWeights[counter])

```

References

1. Figueira, J., & Roy, B. (2002). Determining the weights of criteria in the ELECTRE type methods with a revised Simos' procedure. *European Journal of Operational Research*, 139(2), 317–326.
2. Simos, J. (1990). *Evaluer l'impact sur l'environnement: Une approche originale par l'analyse multicritère et la négociation*. Lausanne: Presses polytechniques et universitaires romandes.
3. Simos, J. (1990). *L'évaluation environnementale: Un processus cognitif négocié*. Ph.D. dissertation, DGF-EPFL, Lausanne, France.

Index

A

Analytic hierarchy process (AHP), 111
 consistency check, 111, 113
 eigenvector method, 112
 geometric mean method, 113
 implementation, 118
 local priority vector, 114
 normalized column sum method, 113
 numerical example, 114
 pairwise comparison matrix, 110
 priority vector, 112
 rank reversal, 113, 124
 ranking, 114
 reciprocal matrix, 110
 transitivity rule, 110

C

Center of area, 41, 43
Chebyshev Goal Programming, 158, 159
 implementation, 161
 methodology, 159
 numerical example, 159
Classical Goal Programming, 132, 133
 implementation, 143
 methodology, 133
 numerical example, 138
Commensurability, 146
Complete ranking, 63
Compromise solution, 33, 44
 consensus, 34, 44
 maximum group utility, 34, 44
 veto, 34, 44

Consistency check, 111, 113
Crisp value, 41, 43
Criteria weights, 166

D

Defuzzify, 41, 43
Deviational variable, 132

E

Eigenvalue, 111
Eigenvector, 112
Euclidean normalization, 147

F

Fuzzy number, 14, 40
 center of area, 41, 43
 crisp value, 41, 43
 defuzzify, 41, 43
 trapezoidal, 41
 triangular, 14
Fuzzy TOPSIS, 17
 aggregation, 17
 distance measure, 19
 fuzzy decision matrix, 18
 fuzzy weights, 18
 ideal/anti-ideal solutions, 19
 implementation, 22
 normalization, 18
 numerical example, 20
 relative closeness, 19

Fuzzy VIKOR, 42
 aggregation, 42
 compromise solution, 44
 fuzzy decision matrix, 42
 fuzzy weights, 43
 implementation, 46
 maximum group utility, 44
 numerical example, 45
 ranking, 44

G

GAIA, 77
 Geometric mean, 113
 Global flow, 64
 Goal, 132
 Goal Programming, 132
 deviational variable, 132
 goal, 132
 hard constraint, 133
 soft constraint, 133

H

Hard constraint, 133

I

Ideal/anti-ideal solutions, 3, 19
 Indifference threshold, 61
 Inferiority flow, 93
 Inferiority matrix, 93
 Inflection point, 61

L

Lexicographic Goal Programming, 152
 implementation, 155
 methodology, 152
 numerical example, 152
 priority level, 152
 Linear normalization, 2, 18
 Linguistic variable, 16, 18, 42

M

Maximum group utility, 33, 44
 Minmax Goal Programming, 158

N

Net flow, 63
 n-flow, 96
 Non-preemptive Goal Programming, 146

Normalization, 2, 18, 146, 147
 euclidean normalization, 147
 linear normalization, 2, 18
 percentage normalization, 147
 vector normalization, 2
 zero-one normalization, 147

O

Outranking flow, 62

P

Pairwise comparison matrix, 110
 Partial ranking, 62
 Percentage normalization, 147
 Preemptive Goal Programming, 152
 Preference, 92
 degree, 59
 indices, 62
 threshold, 61
 Preference function, 59, 92
 Gaussian criterion, 62
 Level criterion, 61
 U-shape criterion, 61
 Usual criterion, 61
 V-shape criterion, 61
 V-shape with indifference criterion, 62
 Principal component analysis technique, 77
 Priority level, 152
 PROMETHEE, 58
 complete ranking, 63
 GAIA, 77
 Group Decision Support System, 78
 indifference threshold, 61
 inflection point, 61
 methodology, 64
 outranking flow, 62
 partial ranking, 62
 preference degree, 59
 preference function, 59
 preference indices, 62
 preference threshold, 61
 unicriterion flow, 62
 PROMETHEE I, 62
 PROMETHEE II, 62
 global flow, 64
 implementation, 71
 net flow, 63
 numerical example, 65
 PROMETHEE V, 80
 implementation, 86
 methodology, 80
 numerical example, 85

PuLP, 81

 example, 81–83

Pyomo, 134

 example, 134, 136

R

Rank reversal, 13, 113, 124

Reciprocal matrix, 110

Relative closeness, 4, 19

Revised Simos, 166

 criteria weights, 166

 implementation, 168

 numerical example, 167

r-flow, 96

S

SIR-SAW, 93

SIR-TOPSIS, 94

Soft constraint, 133

Superiority, 92

Superiority and inferiority ranking method
 (SIR)

 aggregation, 93

 implementation, 101

 inferiority flow, 93

 inferiority matrix, 93

 n-flow, 96

 numerical example, 98

 preference, 92

 preference function, 92

 r-flow, 96

 ranking, 96

 SIR-SAW, 93

 SIR-TOPSIS, 94

 superiority, 92

 superiority flow, 93

 superiority matrix, 92

Superiority flow, 93

Superiority matrix, 92

T

TOPSIS, 2

distance measure, 4

ideal/anti-ideal solutions, 3

implementation, 7

normalization, 2

numerical example, 5

rank reversal, 13

ranking, 4, 20

relative closeness, 4

Trade-offs, 34

Transitivity rule, 110

Trapezoidal fuzzy number, 40

Triangular fuzzy number, 14

U

Unicriterion flow, 62

V

Vector normalization, 2

Vertex method, 16

VIKOR, 33

 compromise solution, 33

 implementation, 38

 maximum group utility, 33

 numerical example, 35

 ranking, 33

 trade-offs, 34

 weight stability interval, 34

W

Weight stability interval, 34

Weighted Goal Programming, 146, 147

 commensurability, 146

 implementation, 149

 methodology, 147

 normalization, 146, 147

 numerical example, 148

 weight, 146

Z

Zero-one normalization, 147