Chapter 1
TOPSIS

1.1 Introduction

TOPSIS is an acronym that stands for 'Technique of Order Preference Similarity to the Ideal Solution' and is a pretty straightforward MCDA method. As the name implies, the method is based on finding an ideal and an anti-ideal solution and comparing the distance of each one of the alternatives to those. It was presented in [7, 14], and can be considered as one of the classical MCDA methods that has received a lot of attention from scholars and researchers. It has been successfully applied in various instances; for a comprehensive state-of-the-art literature review, refer to Behzadian et al. [3]. Table 1.1, adopted from that reference, presents the distribution of papers on TOPSIS by application areas.

The initial methodology was versatile enough to allow for various experiments and modifications; research has focused on the normalization procedure, the proper determination of the ideal and the anti-ideal solution, and the metric used for the calculation of the distances from the ideal and the anti-ideal solution. Studies also emerged using fuzzy [1, 11, 19, 34, 35, 39] or interval numbers [13, 17, 18] and it has been successfully extended to group decision making. This chapter will present the basic methodology with comments on the various possibilities in each step; a discussion on the rank reversal phenomenon will also take place. In all cases, there will be a detailed numerical example and an implementation in Python. Finally, the chapter will conclude with an extension of TOPSIS for group decision making with fuzzy triangular numbers.

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-319-91648-4_1) contains supplementary material, which is available to authorized users.

J. Papathanasiou, N. Ploskas, *Multiple Criteria Decision Aid*, Springer Optimization and Its Applications 136, https://doi.org/10.1007/978-3-319-91648-4_1

Table 1.1 Distribution of papers on TOPSIS by application areas [3]

Area	N	%
Supply chain management and logistics	74	27.5
Design, engineering, and manufacturing systems	62	23
Business and marketing management	33	12.3
Health, safety, and environment management	28	10.4
Human resources management	24	8.9
Energy management	14	5.2
Chemical engineering	7	2.6
Water resources management	7	2.6
Other topics	20	7.4
Total	269	100

1.2 Methodology

It is originally assumed that a typical decision matrix with m alternatives, $A_1, \ldots, A_m$, and n criteria, $C_1, \ldots, C_n$, is properly formulated first. Initially, each alternative is evaluated with respect to each of the n criteria separately. These evaluations form a decision matrix $X = \left(x_{ij}\right)_{m\times n}$. Let also $W = (w_1, \ldots, w_n)$ be the vector of the criteria weights, where $\sum_{j=1}^{n} w_j = 1$. The TOPSIS algorithm is comprised of the following six steps:

Step 1. Normalization of the Decision Matrix
In order to be able to compare different kinds of criteria the first step is to make them dimensionless, i.e., eliminate the units of the criteria. In the normalized decision matrix, the normalized values of each performance x_{ij} is calculated as

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{1.1}$$

Apart from this technique (called vector normalization), several alternatives exist; the linear normalization is as

$$r_{ij} = \frac{x_{ij}}{x_j^+}, \quad x_j^+ = \max_j \left(x_{ij}\right), \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.2}$$

if the criterion is to be maximized (benefit criteria) and

$$r_{ij} = \frac{x_{ij}}{x_j^-}, \quad x_j^- = \min_j \left(x_{ij}\right), \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.3}$$

if the criterion is to be minimized (cost criteria).

Shih et al. [31] present a summary of other variants of the linear normalization and other methods like the non-monotonic normalization. This is an important step than can greatly affect the final ranking; Pavlicic [29] has pondered upon the effects of normalization to the results of MCDA methods. More recently, Jahan and Edwards [16] published a survey on the influence of normalization techniques in ranking, focusing on the material selection process, but their research can be applied in other domains as well. On the other hand, Vafaei et al. [33] analyzed in detail the effects of the normalization process on a case study based on TOPSIS. The avoidance of the rank reversal phenomenon is also focused on this step (to be further discussed later in this chapter).

Step 2. Calculation of the Weighted Normalized Decision Matrix
The weights are the only subjective parameters (as opposed to other MCDA methodologies) taken into account in TOPSIS; the second step is about the multiplication of the normalized decision matrix with the weight associated with each of the criteria.

$$\sum_{j=1}^{n} w_j = 1, \quad j = 1, 2, \cdots, n \tag{1.4}$$

where w_j is the weight of the jth criterion and

$$v_{ij} = w_j r_{ij}, \quad i = 1, \cdots, m, \quad j = 1, \cdots, n \tag{1.5}$$

are the weighted normalized values.

Step 3. Determination of the Ideal (Zenith) and Anti-ideal (Nadir) Solutions
The simplest case is that the ideal and anti-ideal points are fixed by the decision maker, but this should be avoided as it would imply that the decision maker can actually make a credible elicitation of the two points and it would add more subjectivity to the procedure. Another alternative is that the ideal solution (A^*) is

$$A^* = \left\{v_1^*, v_2^*, \cdots, v_n^*\right\} = \left\{\left(\max_j v_{ij} | i \in I^{'}\right), \left(\min_j v_{ij} | i \in I^{''}\right)\right\}, \quad i = 1, 2, \cdots, m, \quad j = 1, \cdots, n \tag{1.6}$$

Meaning that the ideal solution comes from collecting the best performances from the normalized decision matrix. Respectively, the anti-ideal solution (A^-) is

$$A^- = \left\{v_1^-, v_2^-, \cdots, v_n^-\right\} = \left\{\left(\min_j v_{ij} | i \in I^{'}\right), \left(\max_j v_{ij} | i \in I^{''}\right)\right\}, \quad i = 1, 2, \cdots, m, \quad j = 1, \cdots, n \tag{1.7}$$

In this case and in accordance with the ideal solution, the anti-ideal solution is derived by the worst performances in the normalized decision matrix. $I^{'}$ is

associated with benefit criteria and I'' is associated with cost criteria. A third option (others are available in the literature as well) is to use absolute ideal and anti-ideal points, e.g., $A^* = (1, 1, \cdots, 1)$ and $A^- = (0, 0, \cdots, 0)$.

Step 4. Calculation of the Separation Measures
This step is about the calculation of the distances of each alternative from the ideal solution as

$$D_i^* = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^*\right)^2}, \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.8}$$

Similarly, the distances from the anti-ideal solution are calculated as

$$D_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}, \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.9}$$

The above is the most popular case and is the classical Euclidean distance. Other metrics have been adopted, e.g., the Hamming Distance [15] and the Manhattan and Chebyshev distances [31].

Step 5. Calculation of the Relative Closeness to the Ideal Solution
The relative closeness C_i^* is always between 0 and 1 and an alternative is best when it is closer to 1. It is calculated for each alternative and is defined as

$$C_i^* = \frac{D_i^-}{D_i^* + D_i^-}, \quad i = 1, 2, \cdots, m \tag{1.10}$$

Step 6. Ranking of the Preference Order
Finally, the alternatives are ranked from best (higher relative closeness value) to worst. The best alternative and the solution to the problem is on the top of the list.

1.2.1 Numerical Example

The facility location (or location-allocation) problem is a well-known and extensively studied problem in the operational research discipline (for comprehensive reviews, see [28, 30]). In this instance, let's assume that a firm is trying to identify the best site out of six possible choices in order to locate a production facility, taking at the same time into account a number of criteria, namely the investment costs, the employment needs, the social impact, and the environmental impact. The first two criteria are quantitative variables with deterministic values, while the other two are qualitative and rated in a typical Likert scale, ranging from 1 (worst) to 7 (best); all the criteria need to be maximized, since we consider the firm to be a public one.

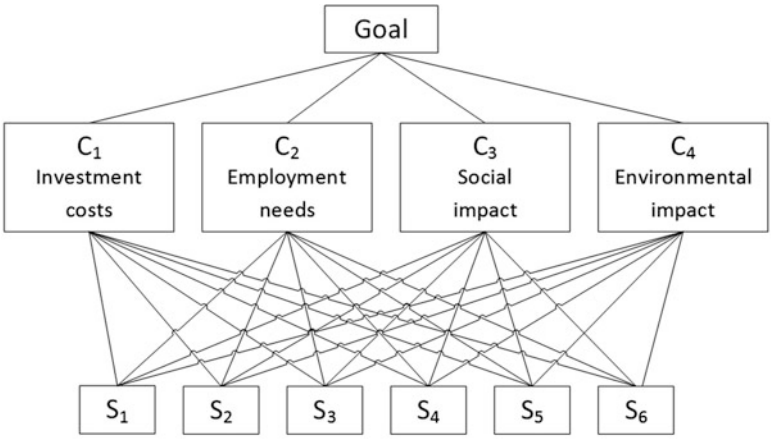

Fig. 1.1 The problem hierarchical structure

Table 1.2 Initial data

	Investment costs (million €)	Employment needs (hundred employees)	Social impact (1–7)	Environmental impact (1–7)
Weight	0.4	0.4	0.1	0.2
Site 1	8	7	2	1
Site 2	5	3	7	5
Site 3	7	5	6	4
Site 4	9	9	7	3
Site 5	11	10	3	7
Site 6	6	9	5	4

If it was private, it would have a different perspective and would most probably be keen on minimizing the first two criteria. The problem structure is shown in Figure 1.1. Table 1.2 includes the weights of each criterion and the performances of the alternatives over all criteria.

Initially, we calculate the normalized decision matrix using the vector normalization technique in Equation (1.1) (Table 1.3). Next, we multiply the normalized decision matrix with the weight associated with each of the criteria using Equation (1.5) (Table 1.4). Then, we determine the ideal and anti-ideal solutions using Equations (1.6) and (1.7) (Table 1.5). Finally, we calculate the distances of each alternative from the ideal and the anti-ideal solution using Equations (1.8) and (1.9) and we also calculate the relative closeness of each alternative using Equation (1.10) (Table 1.6). Figure 1.2 is a graphical representation of Table 1.6. The final ranking of the sites (from best to worst) is Site 5–Site 4–Site 6–Site 3–Site 1–Site 2.

Table 1.3 Normalized decision matrix

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	0.413	0.377	0.152	0.093
Site 2	0.258	0.162	0.534	0.464
Site 3	0.361	0.269	0.457	0.371
Site 4	0.464	0.485	0.534	0.279
Site 5	0.567	0.538	0.229	0.650
Site 6	0.309	0.485	0.381	0.371

Table 1.4 Weighted normalized decision matrix

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	0.165	0.113	0.015	0.019
Site 2	0.103	0.048	0.053	0.093
Site 3	0.144	0.081	0.046	0.074
Site 4	0.186	0.145	0.053	0.056
Site 5	0.227	0.162	0.023	0.130
Site 6	0.124	0.145	0.038	0.074

Table 1.5 Ideal and anti-ideal solutions

	Investment costs	Employment needs	Social impact	Environmental impact
Positive ideal solution A_j^*	0.227	0.161	0.053	0.130
Negative ideal solution A_j^-	0.103	0.049	0.015	0.019

Table 1.6 Final ranking for the facility location problem

Site	D_i^*	D_i^-	C_i^*
Site 1	0.141	0.089	0.387
Site 2	0.171	0.083	0.327
Site 3	0.128	0.082	0.389
Site 4	0.086	0.138	0.617
Site 5	0.030	0.201	0.870
Site 6	0.119	0.116	0.493

1.2.2 Python Implementation

The file *TOPSIS_example_1.py* includes a Python implementation that shows how easy it is to use Python in order to obtain the solution of this problem. The input variables are arrays *x* (alternatives performance, lines 9–10) and *weights* (as the name implies, line 13). Comments embedded in the code listing are above each specific step of the TOPSIS procedure. If we omit the blank and comment lines, then we have a solution to this problem in just 22 lines of code; this is just to show how powerful Python is. The normalization method used in lines 18–20 is

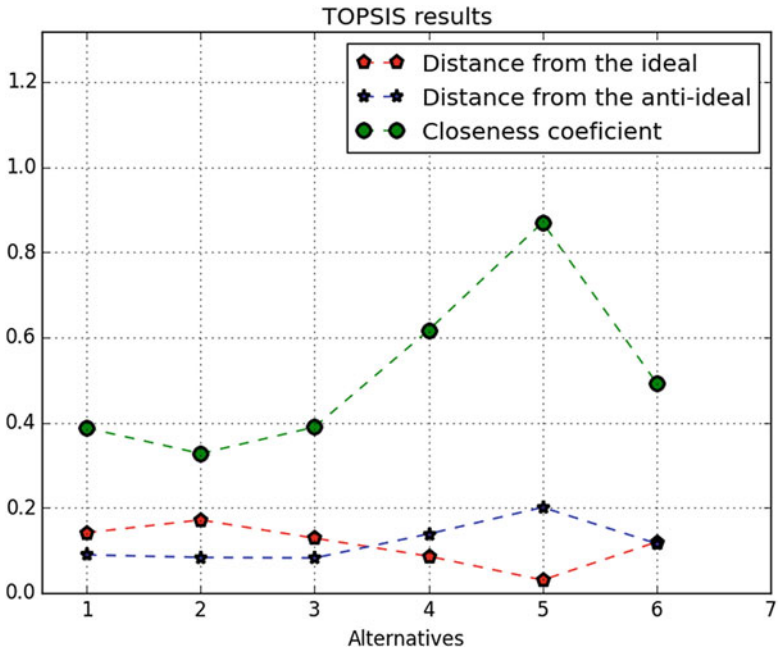

Fig. 1.2 Ideal solution, anti-ideal solution, and closeness coefficient for the facility location problem

the vector normalization and the determination of the ideal and anti-ideal solution is based on finding the best and worst performances (lines 28–31, arrays *pis* and *nis*, respectively) of the normalized matrix. We also use the Euclidean distance to measure the distance of the alternatives to the ideal and anti-ideal solution (lines 35–37 and 41–43, arrays *dpis* and *dnis*, respectively). Finally, the result (relative closeness coefficients) is calculated in lines 47–48 and printed in the console in line 49.

```
1.   # Filename: TOPSIS_example_1.py
2.   # Description: Application of TOPSIS in the first facility
3.   # location problem of Chapter 1
4.   # Authors: Papathanasiou, J. & Ploskas, N.
5.
6.   from numpy import *
7.
8.   # performances of the alternatives
9.   x = array([[8, 7, 2, 1], [5, 3, 7, 5], [7, 5, 6, 4],
10.       [9, 9, 7, 3], [11, 10, 3, 7], [6, 9, 5, 4]])
11.
12.  # weights of the criteria
13.  weights = array([0.4, 0.3, 0.1, 0.2])
14.
```

```
15.  # Step 1 (vector normalization): cumsum() produces the
16.  # cumulative sum of the values in the array and can also
17.  # be used with a second argument to indicate the axis to use
18.  col_sums = array(cumsum(x**2, 0))
19.  norm_x = array([[round(x[i, j] / sqrt(col_sums[x.shape[0]
20.      - 1, j]), 3) for j in range(4)] for i in range(6)])
21.
22.  # Step 2 (Multiply each evaluation by the associated weight):
23.  # wnx is the weighted normalized x matrix
24.  wnx = array([[round(weights[i] * norm_x[j, i], 3)
25.      for i in range(4)] for j in range(6)])
26.
27.  # Step 3 (positive and negative ideal solution)
28.  pis = array([amax(wnx[:, :1]), amax(wnx[:, 1:2]),
29.      amax(wnx[:, 2:3]), amax(wnx[:, 3:4])])
30.  nis = array([amin(wnx[:, :1]), amin(wnx[:, 1:2]),
31.      amin(wnx[:, 2:3]), amin(wnx[:, 3:4])])
32.
33.  # Step 4a: determine the distance to the positive ideal
34.  # solution (dpis)
35.  b1 = array([[(wnx[j, i] - pis[i])**2 for i in range(4)]
36.      for j in range(6)])
37.  dpis = sqrt(sum(b1, 1))
38.
39.  # Step 4b: determine the distance to the negative ideal
40.  # solution (dnis)
41.  b2 = array([[(wnx[j, i] - nis[i])**2 for i in range(4)]
42.      for j in range(6)])
43.  dnis = sqrt(sum(b2, 1))
44.
45.  # Step 5: calculate the relative closeness to the ideal
46.  # solution
47.  final_solution = array([round(dnis[i] / (dpis[i] + dnis[i]),
48.      3) for i in range(6)])
49.  print("Closeness coefficient = ", final_solution)
```

Although the code in file *TOPSIS_example_1.py* produces correct results, it is however rather simplistic and does not take full advantage of Python's capabilities. The file *TOPSIS.py* includes a Python implementation that uses the same input and produces the same results, it has however additional functionalities and allows for more experimentations. Apart from *numpy*, it uses the *matplotlib* library to plot the results and the *timeit* module to time the procedure (lines 5–7). Each step is implemented in its own function and the way to use the function, e.g., the input of the function, is embedded as a Python doc string.

More analytically:

- Step 1 (function *norm(x, y)*) is in lines 10–29. It includes two different normalization techniques, namely the vector (lines 15–20) and the linear (lines 21–29) normalization techniques. The inputs are the array with the alternatives performances (variable *x*) and the normalization method (variable *y*). Its output is the normalized decision matrix (variable *z*).

- Step 2 (function *mul_w(r, t)*) is in lines 32–40. The inputs are the criteria weights matrix (variable *r*) and the normalized matrix from Step 1 (variable *t*). Its output is the weighted normalized decision matrix (variable *z*).
- Step 3 (function *zenith_nadir(x, y)*) is in lines 43–61. The inputs are the weighted normalized decision matrix (parameter *x* from Step 2) and the method used to acquire the ideal and anti-ideal points (parameter *y*). For min/max is 'm' (lines 49–57) and anything else for absolute (for clarity, users can enter 'a', lines 58–61). Its outputs are the ideal and anti-ideal solutions for each criterion (variables *b* and *c*, respectively).
- Step 4 (function *distance(x, y, z)*) is in lines 65–76. As inputs, it requires the weighted normalized decision matrix (parameter *x* from Step 2) and the results of the zenith and nadir points from Step 3 (parameters *y* and *z*). Its outputs are the distances from the zenith and nadir points.
- Step 5 (function *topsis(matrix, weight, norm_m, id_sol, pl)*) is in lines 80–110. This function calls all the other functions and produces the final result. Variable *matrix* is the initial decision matrix, *weight* is the criteria weights matrix, *norm_m* is the normalization method choice, *id_sol* is the action used to define the ideal and the anti-ideal solution, and *pl* indicates whether or not to plot the results using *matplotlib* or not. Figure 1.2 was produced using the *matplotlib* library.

Note that the code allows the inclusion of only benefit criteria. It is however easy for interested readers to make the appropriate changes to also allow cost criteria.

```
1.   # Filename: TOPSIS.py
2.   # Description: TOPSIS method
3.   # Authors: Papathanasiou, J. & Ploskas, N.
4.
5.   from numpy import *
6.   import matplotlib.pyplot as plt
7.   import timeit
8.
9.   # Step 1: normalize the decision matrix
10.  def norm(x, y):
11.      """ normalization function; x is the array with the
12.      performances and y is the normalization method.
13.      For vector input 'v' and for linear 'l'
14.      """
15.      if y == 'v':
16.          k = array(cumsum(x**2, 0))
17.          z = array([[round(x[i, j] / sqrt(k[x.shape[0] - 1,
18.              j]), 3) for j in range(x.shape[1])]
19.              for i in range(x.shape[0])])
20.          return z
21.      else:
22.          yy = []
23.          for i in range(x.shape[1]):
24.              yy.append(amax(x[:, i:i + 1]))
25.              k = array(yy)
26.          z = array([[round(x[i, j] / k[j], 3)
```

```
            for j in range(x.shape[1])]
            for i in range(x.shape[0])])
        return z

# Step 2: find the weighted normalized decision matrix
def mul_w(r, t):
    """ multiplication of each evaluation by the associate
    weight; r stands for the weights matrix and t for
    the normalized matrix resulting from norm()
    """
    z = array([[round(t[i, j] * r[j], 3)
        for j in range(t.shape[1])]
        for i in range(t.shape[0])])
    return z

# Step 3: calculate the ideal and anti-ideal solutions
def zenith_nadir(x, y):
    """ zenith and nadir virtual action function; x is the
    weighted normalized decision matrix and y is the
    action used. For min/max input 'm' and for absolute
    input enter 'a'
    """
    if y == 'm':
        bb = []
        cc = []
        for i in range(x.shape[1]):
            bb.append(amax(x[:, i:i + 1]))
            b = array(bb)
            cc.append(amin(x[:, i:i + 1]))
            c = array(cc)
        return (b, c)
    else:
        b = ones(x.shape[1])
        c = zeros(x.shape[1])
        return (b, c)

# Step 4: determine the distance to the ideal and anti-ideal
# solutions
def distance(x, y, z):
    """ calculate the distances to the ideal solution (di+)
    and the anti-ideal solution (di-); x is the result
    of mul_w() and y, z the results of zenith_nadir()
    """
    a = array([[(x[i, j] - y[j])**2
       for j in range(x.shape[1])]
       for i in range(x.shape[0])])
    b = array([[(x[i, j] - z[j])**2
       for j in range(x.shape[1])]
        for i in range(x.shape[0])])
    return (sqrt(sum(a, 1)), sqrt(sum(b, 1)))

# TOPSIS method: it calls the other functions and includes
# step 5
def topsis(matrix, weight, norm_m, id_sol, pl):
```

```
    """ matrix is the initial decision matrix, weight is
    the weights matrix, norm_m is the normalization
    method, id_sol is the action used, and pl is 'y'
    for plotting the results or any other string for
    not
    """
    z = mul_w(weight, norm(matrix, norm_m))
    s, f = zenith_nadir(z, id_sol)
    p, n = distance(z, s, f)
    final_s = array([n[i] / (p[i] + n[i])
       for i in range(p.shape[0])])
    if pl == 'y':
       q = [i + 1 for i in range(matrix.shape[0])]
       plt.plot(q, p, 'p--', color = 'red',
           markeredgewidth = 2, markersize = 8)
       plt.plot(q, n, '*--',  color = 'blue',
           markeredgewidth = 2, markersize = 8)
       plt.plot(q, final_s, 'o--', color = 'green',
            markeredgewidth = 2, markersize = 8)
       plt.title('TOPSIS results')
       plt.legend(['Distance from the ideal',
           'Distance from the anti-ideal',
           'Closeness coefficient'])
       plt.xticks(range(matrix.shape[0] + 2))
       plt.axis([0, matrix.shape[0] + 1, 0, 3])
       plt.xlabel('Alternatives')
       plt.legend()
       plt.grid(True)
       plt.show()
    return final_s

# performances of the alternatives
x = array([[8, 7, 2, 1], [5, 3, 7, 5], [7, 5, 6, 4],
           [9, 9, 7, 3], [11, 10, 3, 7], [6, 9, 5, 4]])

# weights of the criteria
w = array([0.4, 0.3, 0.1, 0.2])

# final results
start = timeit.default_timer()
topsis(x, w, 'v', 'm', 'n')
stop = timeit.default_timer()
print("time = ", stop - start)
print("Closeness coefficient = ",
   topsis(x, w, 'v', 'm', 'y'))
```

The results are shown in Table 1.7 and Figure 1.3 if we choose the vector normalization and the calculation of the best and worst performances for the ideal and anti-ideal solutions (case (a), Table 1.7 and Figure 1.3). If we choose the linear normalization and absolute ideal and anti-ideal points, the results are more or less the same for the top three sites (case (b), Table 1.7 and Figure 1.3). However, if we modify the data for the first criterion (investments costs) and change the performance of site 4 from 9 to 1 and of site 5 from 11 to 3, the results are quite

Table 1.7 Solution for the facility location problem using initial data

	Solution using vector normalization and min/max for the ideal and anti-ideal solution (case (a))			Solution using linear normalization and absolute for the ideal and anti-ideal solution (case (b))		
	D_i^*	D_i^-	C_i^*	D_i^*	D_i^-	C_i^*
Site 1	0.141	0.089	0.387	1.736	0.361	0.172
Site 2	0.171	0.083	0.327	1.744	0.268	0.133
Site 3	0.128	0.082	0.389	1.703	0.328	0.161
Site 4	0.086	0.138	0.617	1.622	0.444	0.215
Site 5	0.030	0.201	0.870	1.551	0.540	0.258
Site 6	0.119	0.116	0.493	1.671	0.372	0.182

Table 1.8 Solution for the facility location problem using modified data

	Solution using vector normalization and min/max for the ideal and anti-ideal solution (case (c))			Solution using linear normalization and absolute for the ideal and anti-ideal solution (case (d))		
	D_i^*	D_i^-	C_i^*	D_i^*	D_i^-	C_i^*
Site 1	0.127	0.216	0.630	1.694	0.454	0.211
Site 2	0.147	0.144	0.495	1.713	0.318	0.157
Site 3	0.102	0.190	0.650	1.663	0.407	0.197
Site 4	0.219	0.111	0.335	1.755	0.305	0.148
Site 5	0.151	0.168	0.527	1.664	0.393	0.191
Site 6	0.084	0.186	0.689	1.634	0.425	0.207

different and the ranking changes considerably between the versions of TOPSIS and for the same data set, as seen in Table 1.8 and Figure 1.3 (cases (c) and (d)). The reader is encouraged to experiment further with the various variants of TOPSIS and compare the results. An interesting note can be made on the running time of the TOPSIS algorithm; using the *timeit* module in lines 120–122 and calling function *topsis* without printing the results (the value of variable *pl* is set to *'n'*), the running time is (an average of ten runs) 0.00085 s on a Linux machine with an Intel Core i7 at 2.2 GHz CPU and 6 GB RAM. Running the same code with 1,000 sites and four criteria, the execution time is 0.1029 s.

1.2.3 Rank Reversal

TOPSIS is also not immune to a phenomenon called rank reversal. This is a phenomenon especially studied for other MCDA methods and will be mentioned again during the course of this book. It has produced much debate among the scientific community and continues to be considered controversial by many. In its simplest form, it occurs when an alternative is added to the initial list of alternatives and then the final ranking changes considerably from the original one; in some cases,

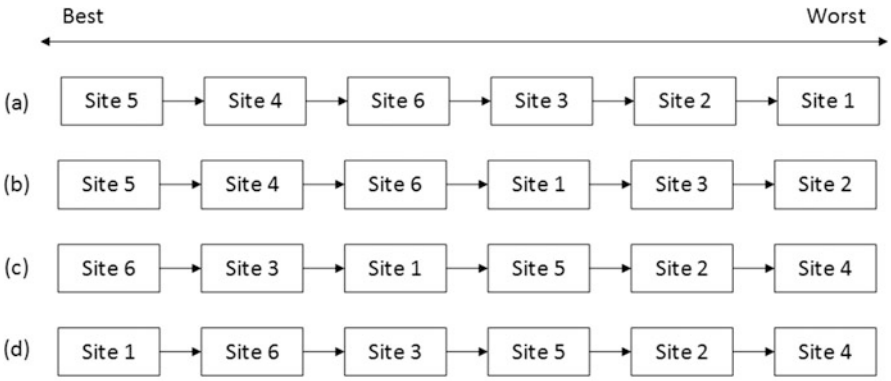

Fig. 1.3 Solutions for the facility location problem cases (a) and (b) with initial data and cases (c) and (d) with modified data

Table 1.9 Initial ranking in the rank reversal example

	$w_1 = 0.5$	$w_2 = 0.5$				
	C_1	C_2	D_i^*	D_i^-	C_i^*	Ranking
A_1	1	5	0.294	0.244	0.453	3
A_2	4	2	0.244	0.294	0.546	1
A_3	3	3	0.189	0.212	0.529	2

we observe that the final ranking is totally reversed. Wang and Luo [37] studied it for a number of methods, including TOPSIS, and concluded by saying that it might be, after all, a normal phenomenon. Yet, they do not provide a solution, while Garcia-Cascales and Lamata [12] propose modifications in the original algorithm in order to avoid it. In the next part of this section, we will recreate the example used by Garcia-Cascales and Lamata [12] to demonstrate the phenomenon.

They consider three different candidates for a certain position, individuals A_1, A_2, and A_3. Each one of them completes two questionnaires with the same weight ($w_1 = 0.5$ and $w_2 = 0.5$). Therefore, the formulated multiple criteria problem has a couple of criteria and three alternatives. The initial input data and the final ranking after applying TOPSIS (using the vector normalization and the calculation of the best and worst performances for the ideal and anti-ideal solutions) are shown in Table 1.9; the ranking is $A_2 > A_3 > A_1$.

Let's add now another candidate for the position, A_4, whose responses to the questionnaires are evaluated to 5 and 1 for C_1 and C_2, respectively (Table 1.10). We observe that the ranking has changed; it is now $A_1 > A_3 > A_2 > A_4$. A_1, which was originally the worst, is now the best; the order is totally reversed! This does not seem logical in many decision problems and is the root of many heated debates.

As already mentioned, the normalization used in their example is the vector normalization, while they determine the ideal and anti-ideal solutions using the best and worst performances. To avoid this phenomenon, they apply the linear

Table 1.10 Ranking after the introduction of a new alternative in the rank reversal example

	$w_1 = 0.5$	$w_2 = 0.5$				
	C_1	C_2	D_i^*	D_i^-	C_i^*	Ranking
A_1	1	5	0.280	0.320	0.533	1
A_2	4	2	0.250	0.225	0.473	3
A_3	3	3	0.213	0.213	0.500	2
A_4	5	1	0.320	0.280	0.467	4

normalization and modify the definitions of the ideal and anti-ideal solutions with the introduction of two fictitious alternatives. From the example, if we refer to the set $S = [1, 2, 3, 4, 5]$, the values of A^* and A^- would be (5, 5) and (0, 0), respectively; these are the new fictitious alternatives and the rank reversal phenomenon can be avoided (until proven otherwise).

1.3 Fuzzy TOPSIS for Group Decision Making

1.3.1 Preliminaries: Elements of Fuzzy Numbers Theory

In this subsection, we will briefly review important elements from the fuzzy numbers theory, adopted mainly from [2, 23, 40, 41]. The aim of this section is not to make even a short introduction to fuzzy theory; just to present a few elements enough for the reader to keep up with the rest of this chapter. A fuzzy set is a class with a continuum of membership grades [40]; so, a fuzzy set A in a referential (universe of discourse) X is characterized by a membership function A, which associates with each element $x \in X$ a real number $A(x) \in [0, 1]$, having the interpretation $A(x)$ as the membership grade of x in the fuzzy set A.

Let's consider now a fuzzy subset of the real line u: $R \rightarrow [0, 1]$. u is a fuzzy number [2, 10], if it satisfies the following properties:

- u is normal, i.e., $\exists x_0 \in R$ with $u(x_0) = 1$.
- u is fuzzy convex, i.e., $u(tx+(1-t)y) \geq \min\{u(x), u(y)\}, \forall t \in [0, 1], x, y \in R$.
- u is upper semi-continuous on R, i.e., $\forall \epsilon > 0, \exists \delta > 0$ such that $u(x) - u(x_0) < \epsilon, |x - x_0| < \delta$.
- u is compactly supported, i.e., $cl\{x \in R; u(x) > 0\}$ is compact, where $cl(A)$ denotes the closure of the set A.

One of the most popular shapes of fuzzy numbers is the triangular fuzzy number that can be defined as a triplet $A = (\alpha_1, \alpha_2, \alpha_3)$, shown in Figure 1.4. The membership functions are as follows:

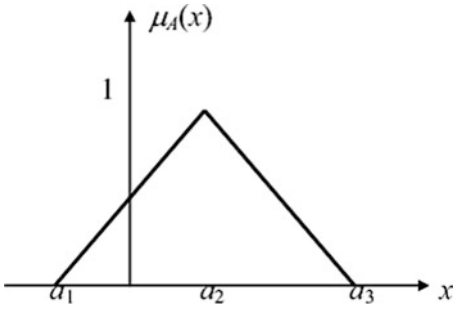

Fig. 1.4 Triangular fuzzy number $A = (\alpha_1, \alpha_2, \alpha_3)$, adopted from [23]

$$\mu_A(x) = \begin{cases} 0, & x < \alpha_1 \\ \frac{x-\alpha_1}{\alpha_2-\alpha_1}, & \alpha_1 \le x \le \alpha_2 \\ \frac{\alpha_3-x}{\alpha_3-\alpha_2}, & \alpha_2 \le x \le \alpha_3 \\ 0, & x > \alpha_3 \end{cases} \tag{1.11}$$

Assuming that both $A = (\alpha_1, \alpha_2, \alpha_3)$ and $B = (b_1, b_2, b_3)$ are triangular fuzzy numbers, then:

- the result from their addition or subtraction is also a triangular fuzzy number.
- the result from their multiplication or division is not a triangular fuzzy number (however, we can assume that the result of multiplication or division is a triangular fuzzy number as an approximation value).
- the max or min operation does not result to a triangular fuzzy number.

Focusing on the addition and the subtraction, they are as

$$A(+)B = (\alpha_1, \alpha_2, \alpha_3)(+)(b_1, b_2, b_3) = (\alpha_1 + b_1, \alpha_2 + b_2, \alpha_3 + b_3) \tag{1.12}$$

and

$$A(-)B = (\alpha_1, \alpha_2, \alpha_3)(-)(b_1, b_2, b_3) = (\alpha_1 - b_3, \alpha_2 - b_2, \alpha_3 - b_1) \tag{1.13}$$

A fuzzy vector is a certain vector that includes an element and has a value between 0 and 1. Bearing this in mind, a fuzzy matrix is a gathering of such vectors. The operations on given fuzzy matrices $A = (\alpha_{ij})$ and $B = (b_{ij})$ are

- sum

$$A + B = \max\left[\alpha_{ij}, b_{ij}\right] \tag{1.14}$$

- max product

$$A \cdot B = \max_k \left[\min\left(\alpha_{ik}, b_{kj}\right)\right] \tag{1.15}$$

- scalar product

$$\lambda A \tag{1.16}$$

where $0 \leq \lambda \leq 1$.

Chen [5] uses the vertex method to calculate the distance between two triangular fuzzy numbers $\widetilde{m} = (m_1, m_2, m_3)$ and $\widetilde{n} = (n_1, n_2, n_3)$ as

$$d(\widetilde{m}, \widetilde{n}) = \sqrt{\frac{1}{3}\left[(m_1 - n_1)^2 + (m_2 - n_2)^2 + (m_3 - n_3)^2\right]} \tag{1.17}$$

Finally, according to Zadeh [41], a linguistic variable is one whose values are words or sentences in a natural or artificial language; among others, he provides an example in the form of the linguistic variable 'Age' that can take the values young, not young, very young, quite young, old, not very old and not very young, etc., rather than 20, 21, 22, 23, $\cdots$. This kind of variables can well be represented by triangular fuzzy numbers. Lee [23] further denotes that a linguistic variable can be defined by the quintuple

$$\text{Linguistic variable} = (x, T(x), U, G, M) \tag{1.18}$$

where:

- x: name of variable
- $T(x)$: set of linguistic terms that can be a value of the variable
- U: set of universe of discourse, which defines the characteristics of the variable
- G: syntactic grammar that produces terms in $T(x)$
- M: semantic rules, which map terms in $T(x)$ to fuzzy sets in U

1.3.2 Fuzzy TOPSIS Methodology

This section presents a fuzzy extension of TOPSIS that is based on Chen's methodology [5], despite the fact that his work has received some criticism by Mahdavi [27]. Variations of fuzzy TOPSIS focus on the distance measurement, the determination of the ideal and anti-ideal points, and the use of other than triangular fuzzy numbers, like trapezoidal fuzzy numbers [24]. However, we use in this section Chen's work due to its relative simplicity as regards to the distance measures compared to the other fuzzy TOPSIS approaches. In addition to the fuzzy numbers (and to complicate things a little bit), TOPSIS is also extended into group decision making involving the opinions of a number of independent experts. Table 1.11, adopted from [20], presents a comparison of the various fuzzy TOPSIS methods proposed in the literature.

Table 1.11 A comparison of fuzzy TOPSIS methods [20]

Source	Attribute weights	Type of fuzzy numbers	Ranking method	Normalization method
Chen and Hwang [7]	Fuzzy numbers	Trapezoidal	Lee and Li's [22] generalized mean method	Linear normalization
Liang [25]	Fuzzy numbers	Trapezoidal	Chen's [6] ranking with maximizing set and minimizing set	Linear normalization
Chen [5]	Fuzzy numbers	Triangular	Chen [5] proposes the vertex method	Linear normalization
Chu [8]	Fuzzy numbers	Triangular	Liou and Wang's [26] ranking method of total integral value with $\alpha = 1/2$	Modified manhattan distance
Tsaur et al. [32]	Crisp values	Triangular	Zhao and Govind's [43] center of area method	Vector normalization
Zhang and Lu [42]	Crisp values	Triangular	Chen's [5] vertex mode	Manhattan distance
Chu and Lin [9]	Fuzzy numbers	Triangular	Kaufmann and Gupta's [21] mean of the removals method	Linear normalization
Cha and Yung [4]	Crisp values	Triangular	Cha and Young [4] propose a fuzzy distance operator	Linear normalization
Yang and Hung [38]	Fuzzy numbers	Triangular	Chen's [5] vertex method	Normalized fuzzy linguistic ratings are used
Wang and Elhag [36]	Fuzzy numbers	Triangular	Chen's [5] vertex method	Linear normalization
Jahanshahloo et al. [18]	Crisp values	Interval data	Jahanshahloo et al. [18] propose a new column and ranking method	

In conjunction with the steps of the typical TOPSIS method presented earlier in this chapter, the steps of the fuzzy extension are:

Step 1. Form a Committee of Decision Makers, Then Identify the Evaluation Criteria

If we assume that the decision group has K persons, then the importance of the criteria and the ratings of the alternatives can be calculated as

$$\widetilde{x}_{ij} = \frac{1}{K}\left[\widetilde{x}_{ij}^{1}(+)\widetilde{x}_{ij}^{2}(+)\cdots(+)\widetilde{x}_{ij}^{K}\right] \tag{1.19}$$

$$\widetilde{w}_{j} = \frac{1}{K}\left[\widetilde{w}_{j}^{1}(+)\widetilde{w}_{j}^{2}(+)\cdots(+)\widetilde{w}_{j}^{K}\right] \tag{1.20}$$

where $\widetilde{x}_{ij}^{K}$ and $\widetilde{w}_{j}^{K}$ are the ratings and criteria weights of the Kth decision maker.

Step 2. Choose the Linguistic Variables
Choose the appropriate linguistic variables for the importance weight of the criteria and the linguistic ratings for alternatives with respect to the criteria.

Step 3. Perform Aggregations
Aggregate the weight of criteria to get the aggregated fuzzy weight $\widetilde{w}_j$ of criterion C_j, and pool the decision makers' opinions to get the aggregated fuzzy rating $\widetilde{x}_{ij}$ of alternative A_i under criterion C_j.

Step 4. Construct the Fuzzy Decision Matrix and the Normalized Fuzzy Decision Matrix
The fuzzy decision matrix is constructed as

$$\widetilde{D} = \begin{bmatrix} \widetilde{x}_{11} & \widetilde{x}_{12} & \cdots & \widetilde{x}_{1n} \\ \widetilde{x}_{21} & \widetilde{x}_{22} & \cdots & \widetilde{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{x}_{m1} & \widetilde{x}_{m2} & \cdots & \widetilde{x}_{mn} \end{bmatrix} \tag{1.21}$$

and the vector of the criteria weights as

$$\widetilde{W} = [\widetilde{w}_1, \widetilde{w}_2, \cdots, \widetilde{w}_n] \tag{1.22}$$

where $\widetilde{x}_{ij}$ and $\widetilde{w}_j, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n$ are linguistic variables according to Step 2. They can be described by the triangular fuzzy numbers $\widetilde{x}_{ij} = \left(a_{ij}, b_{ij}, c_{ij}\right)$ and $\widetilde{w}_j = \left(w_{j1}, w_{j2}, w_{j3}\right)$. For the normalization step, Chen uses the linear scale transformation in order to drop the units and make the criteria comparable; it is also important to preserve the property stating that the ranges of the normalized triangular fuzzy numbers belong to [0, 1]. The normalized fuzzy decision matrix denoted by $\widetilde{R}$ is

$$\widetilde{R} = \left[\widetilde{r}_{ij}\right]_{m \times n} \tag{1.23}$$

The set of benefit criteria is B and the set of cost criteria is C, therefore

$$\widetilde{r}_{ij} = \left(\frac{a_{ij}}{c_j^*}, \frac{b_{ij}}{c_j^*}, \frac{c_{ij}}{c_j^*}\right), \quad j \in B, \quad i = 1, 2, \cdots, m \tag{1.24}$$

$$\widetilde{r}_{ij} = \left(\frac{a_j^-}{c_{ij}}, \frac{a_j^-}{b_{ij}}, \frac{a_j^-}{a_{ij}}\right), \quad j \in C, \quad i = 1, 2, \cdots, m \tag{1.25}$$

$$c_j^* = \max_i c_{ij}, \text{ if } j \in B, \quad i = 1, 2, \cdots, m \tag{1.26}$$

$$a_j^- = \min_i a_{ij}, \text{ if } j \in C, \quad i = 1, 2, \cdots, m \tag{1.27}$$

Step 5. Construct the Fuzzy Weighted Normalized Decision Matrix
Then, the fuzzy weighted normalized decision matrix can be constructed as

$$\widetilde{V} = \left[\widetilde{v}_{ij}\right]_{m\times n}, \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.28}$$

where

$$\widetilde{v}_{ij} = \widetilde{r}_{ij}(\cdot)\widetilde{w}_j, \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.29}$$

The elements $\widetilde{v}_{ij}, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n$, are normalized positive triangular fuzzy numbers ranging from 0 to 1.

Step 6. Determine the Fuzzy Positive Ideal Solution (FPIS) and the Fuzzy Negative Ideal Solution (FNIS)
The fuzzy positive ideal solution (FPIS, A^*) and the fuzzy negative ideal solution (FNIS, A^-) are

$$A^* = \left(\widetilde{v}_1^*, \widetilde{v}_2^*, \cdots, \widetilde{v}_n^*\right) \tag{1.30}$$

$$A^- = \left(\widetilde{v}_1^-, \widetilde{v}_2^-, \cdots, \widetilde{v}_n^-\right) \tag{1.31}$$

where

$$\widetilde{v}_j^* = (1, 1, 1), \widetilde{v}_j^- = (0, 0, 0), \quad j = 1, 2, \cdots, n \tag{1.32}$$

Step 7. Calculate the Distance of Each Alternative from FPIS and FNIS
The distance of each of the alternatives from FPIS and FNIS can be calculated as

$$D_i^* = \sum_{j=1}^{n} d\left(\widetilde{v}_{ij}, \widetilde{v}_j^*\right), \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.33}$$

$$D_i^- = \sum_{j=1}^{n} d\left(\widetilde{v}_{ij}, \widetilde{v}_j^-\right), \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n \tag{1.34}$$

where d is the distance measurement between two fuzzy numbers (Equation (1.17)).

Step 8. Calculate the Closeness Coefficient of Each Alternative
The closeness coefficient of each alternative can be defined as

$$CC_i = \frac{D_i^-}{D_i^* + D_i^-}, \quad i = 1, 2, \cdots, m \tag{1.35}$$

Step 9. Rank the Alternatives
An alternative A_i is better than A_j if its closeness coefficient is closer to 1.

Therefore, the final ranking of the alternatives is defined by the value of the closeness coefficient.

1.3.3 Numerical Example

The problem in hand is the same as in the previous examples, the facility location problem; again there are four criteria with six alternative sites (Figure 1.1), but there are now three individual decision makers with different views to be taken into account. In this case, the importance weights of the qualitative criteria and the ratings are considered as linguistic variables expressed in positive triangular fuzzy numbers, as shown in Tables 1.12 and 1.13; they are also considered to be evaluated by decision makers that are experts on the field. These evaluations are in Tables 1.14 and 1.15.

Initially, we aggregate the weights of criteria to get the aggregated fuzzy weights and the ratings to calculate the fuzzy decision matrix using Equations (1.19) and (1.20) (Table 1.16). Next, we calculate the fuzzy normalized decision matrix using Equations (1.24)–(1.27) (Table 1.17). Then, we construct the fuzzy weighted

Table 1.12 Linguistic variables for the criteria

Linguistic variables for the importance weight of each criterion	
Very low (VL)	(0, 0, 0.1)
Low (L)	(0, 0.1, 0.3)
Medium low (ML)	(0.1, 0.3, 0.5)
Medium (M)	(0.3, 0.5, 0.7)
Medium high (MH)	(0.5, 0.7, 0.9)
High (H)	(0.7, 0.9, 1.0)
Very high (VH)	(0.9, 1.0, 1.0)

Table 1.13 Linguistic variables for the ratings

Linguistic variables for the ratings	
Very poor (VP)	(0, 0, 1)
Poor (P)	(0, 1, 3)
Medium poor (MP)	(1, 3, 5)
Fair (F)	(3, 5, 7)
Medium good (MG)	(5, 7, 9)
Good (G)	(7, 9, 10)
Very good (VG)	(9, 10, 10)

Table 1.14 The importance weight of the criteria for each decision maker

	D_1	D_2	D_3
Investment costs	H	VH	VH
Employment needs	M	H	VH
Social impact	M	MH	ML
Environmental impact	H	VH	MH

Table 1.15 The ratings of the six sites by the three decision makers for the four criteria

Criteria	Candidate sites	Decision makers		
		D_1	D_2	D_3
Investment costs	Site 1	VG	G	MG
	Site 2	MP	F	F
	Site 3	MG	MP	F
	Site 4	MG	VG	VG
	Site 5	VP	P	G
	Site 6	F	G	G
Employment needs	Site 1	F	MG	MG
	Site 2	F	VG	G
	Site 3	MG	MG	VG
	Site 4	G	G	VG
	Site 5	P	VP	MP
	Site 6	F	MP	MG
Social impact	Site 1	P	P	MP
	Site 2	MG	VG	G
	Site 3	MP	F	F
	Site 4	MG	VG	G
	Site 5	G	G	VG
	Site 6	VG	MG	F
Environmental impact	Site 1	G	VG	G
	Site 2	MG	F	MP
	Site 3	MP	P	P
	Site 4	VP	F	P
	Site 5	G	MG	MG
	Site 6	P	MP	F

Table 1.16 Fuzzy decision matrix and fuzzy weights

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	(7.000, 8.667, 9.667)	(4.333, 6.333, 8.333)	(0.333, 1.667, 3.667)	(7.667, 9.333, 10.000)
Site 2	(2.333, 4.333, 6.333)	(6.333, 8.000, 9.000)	(7.000, 8.667, 9.667)	(3.000, 5.000, 7.000)
Site 3	(3.000, 5.000, 7.000)	(6.333, 8.000, 9.333)	(2.333, 4.333, 6.333)	(0.333, 1.667, 3.667)
Site 4	(7.667, 9.000, 9.667)	(7.667, 9.333, 10.000)	(7.000, 8.667, 9.667)	(1.000, 2.000, 3.667)
Site 5	(2.333, 3.333, 4.667)	(0.333, 1.333, 3.000)	(7.667, 9.333, 10.000)	(5.667, 7.667, 9.333)
Site 6	(5.667, 7.667, 9.000)	(3.000, 5.000, 7.000)	(5.667, 7.333, 8.667)	(1.333, 3.000, 5.000)
Weight	(0.833, 0.967, 1.000)	(0.633, 0.800, 0.900)	(0.300, 0.500, 0.700)	(0.700, 0.867, 0.967)

normalized decision matrix using Equations (1.28) and (1.29) (Table 1.18). Finally, we calculate the distance of each alternative from the fuzzy positive ideal and anti-ideal solutions, and calculate the closeness coefficient of each alternative (Table 1.19). Figure 1.5 is a graphical representation of Table 1.19. The final ranking of the sites (from best to worst) is Site 1–Site 4–Site 2–Site 6–Site 5–Site 3.

Table 1.17 Fuzzy normalized decision matrix

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	(0.700, 0.867, 0.967)	(0.433, 0.633, 0.833)	(0.033, 0.167, 0.367)	(0.767, 0.933, 1.000)
Site 2	(0.233, 0.433, 0.633)	(0.633, 0.800, 0.900)	(0.700, 0.867, 0.967)	(0.300, 0.500, 0.700)
Site 3	(0.300, 0.500, 0.700)	(0.633, 0.800, 0.933)	(0.233, 0.433, 0.633)	(0.033, 0.167, 0.367)
Site 4	(0.767, 0.900, 0.967)	(0.767, 0.933, 1.000)	(0.700, 0.867, 0.967)	(0.100, 0.200, 0.367)
Site 5	(0.233, 0.333, 0.467)	(0.033, 0.133, 0.300)	(0.767, 0.933, 1.000)	(0.567, 0.767, 0.933)
Site 6	(0.567, 0.767, 0.900)	(0.300, 0.500, 0.700)	(0.567, 0.733, 0.867)	(0.133, 0.300, 0.500)

Table 1.18 Fuzzy weighted normalized decision matrix

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	(0.700, 0.867, 0.967)	(0.433, 0.633, 0.833)	(0.033, 0.167, 0.367)	(0.767, 0.933, 1.000)
Site 2	(0.233, 0.433, 0.633)	(0.633, 0.800, 0.900)	(0.700, 0.867, 0.967)	(0.300, 0.500, 0.700)
Site 3	(0.300, 0.500, 0.700)	(0.633, 0.800, 0.933)	(0.233, 0.433, 0.633)	(0.033, 0.167, 0.367)
Site 4	(0.767, 0.900, 0.967)	(0.767, 0.933, 1.000)	(0.700, 0.867, 0.967)	(0.100, 0.200, 0.367)
Site 5	(0.233, 0.333, 0.467)	(0.033, 0.133, 0.300)	(0.767, 0.933, 1.000)	(0.567, 0.767, 0.933)
Site 6	(0.567, 0.767, 0.900)	(0.300, 0.500, 0.700)	(0.567, 0.733, 0.867)	(0.133, 0.300, 0.500)

Table 1.19 The final result

Alternative	D_i^*	D_i^-	Closeness coefficient	Final ranking
Site 1	1.965	2.305	0.540	1
Site 2	2.212	2.051	0.481	3
Site 3	2.577	1.673	0.394	6
Site 4	1.959	2.279	0.538	2
Site 5	2.526	1.704	0.403	5
Site 6	2.347	1.914	0.449	4

1.3.4 Python Implementation

The file *FuzzyTOPSIS.py* includes a Python implementation of the fuzzy TOPSIS method. Similar to the implementation of TOPSIS, it uses the *matplotlib* library to plot the results and the *timeit* module to time the procedure (lines 6–7). Each step is implemented in its own function and the way to use the functions, e.g., the input of the function, is embedded as a Python doc string.

More analytically:

- Steps 1 and 2 of the fuzzy TOPSIS procedure are in lines 168–198. Dictionaries *cw* (lines 168–171) and *r* (lines 175–177) correspond to Tables 1.12 and 1.13, respectively; they are the definitions of the linguistic variables used for the criteria weights and the ratings. Python list *cdw* (lines 180–181) is about the importance weight of the criteria (Table 1.14) and the ratings of the six candidate sites by the three decision makers are each in its own list, named *c1*, *c2*, $\cdots$, *c6* (lines 185–196, Table 1.15). These lists are concatenated into one list in line 198.
- Steps 3 and 4 are in lines 119–122. Arrays *fuzzy_weights* and *fuzzy_decision_matrix* are the output of function *cal(a, b, k)*. Variable *a* is the dictionary with the

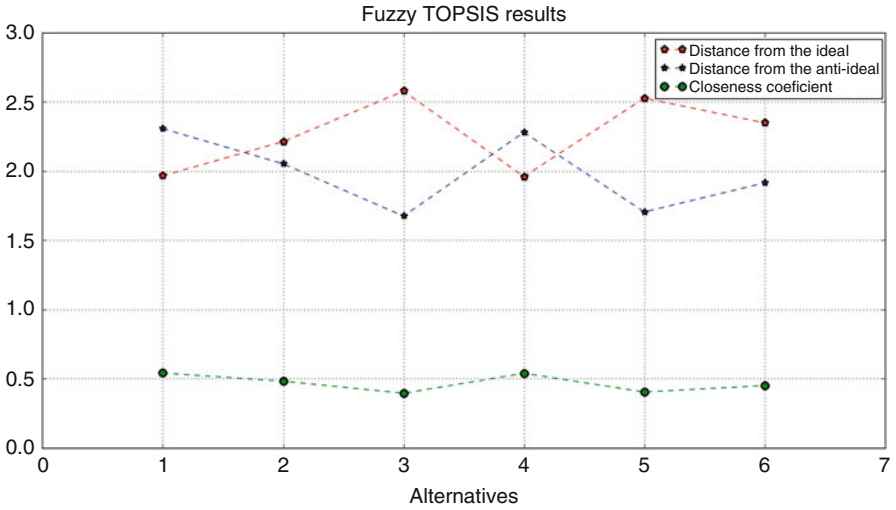

Fig. 1.5 The final result

linguistic variables for the criteria weights or the linguistic variables for the ratings, variable *b* is the matrix with the criteria weights or the ratings, and variable *k* is the number of the decision makers. The output is the fuzzy weights of the criteria or the fuzzy decision matrix (Table 1.16). Array *fuzzy_norm_decision_matrix* is the output of function *fndm(a, n, m)*, which uses as an input parameter the output of function *cal*. Variable *a* is the fuzzy decision matrix, variable *n* is the number of criteria, and variable *m* is the number of the alternatives. The output of function *fndm* is the fuzzy normalized decision matrix (Table 1.17).

- Step 5 is in lines 125–127. The fuzzy weighted normalized decision matrix is calculated by calling function *weighted_fndm(a, b, n, m)* that uses as input the results of the previous steps (Table 1.18). Variable *a* is the fuzzy normalized decision matrix, variable *b* is the criteria weights, variable *n* is the number of criteria, and variable *m* is the number of the alternatives. The output is the fuzzy weighted normalized decision matrix.
- Steps 6 and 7 are in lines 130–133. They make use of functions *func_dist_fpis(a, n, m)* and *func_dist_fnis(a, n, m)*. The first function is about finding the distance from the ideal solution and the second finding the distance from the anti-ideal solution (Table 1.19). Variable *a* is the fuzzy weighted normalized decision matrix, variable *n* is the number of criteria, and variable *m* is the number of the alternatives. The output is the ideal or anti-ideal solution of each criterion. Both functions use function *distance(a, b)* to calculate the distance between two fuzzy triangular numbers. Variables *a* and *b* are fuzzy triangular numbers and the output is their distance.
- Step 8 is in lines 136–139 and is about finding the final solution (Table 1.19 and Figure 1.5).

- Final results (function *f_topsis(a, b, c, d, n, m, k, pl)*) are calculated in lines 107–159. This function calls all the other functions and produces the final result. Variable *a* is the dictionary with the linguistic variables for the criteria weights, variable *b* is the matrix with the importance weights of the criteria, variable *c* is a dictionary with the linguistic variables for the ratings, variable *d* is the matrix with all the ratings, variable *n* is the number of criteria, variable *m* is the number of the alternatives, variable *k* is the number of the decision makers, and variable *pl* is whether to plot the results using *matplotlib* or not. Figure 1.5 was produced using the *matplotlib* library.

Similarly to the implementation of TOPSIS, the code of fuzzy TOPSIS allows the inclusion of only benefit criteria. It is however easy for interested readers to make the appropriate changes to also allow cost criteria.

Using the *timeit* module in lines 201–204 and calling function *f_topsis* without printing the results (the value of variable *pl* is set to *'n'*), the running time is (an average of ten runs) 0.0024 s on a Linux machine with an Intel Core i7 at 2.2 GHz CPU and 6 GB RAM. Running the same code with 1,000 sites, four criteria, and three decision makers, the execution time is 0.4015 s.

```
1.   # Filename: FuzzyTOPSIS.py
2.   # Description: Fuzzy TOPSIS method
3.   # Authors: Papathanasiou, J. & Ploskas, N.
4.
5.   from numpy import *
6.   import matplotlib.pyplot as plt
7.   import timeit
8.
9.   # Convert the linguistic variables for the criteria weights
10.  # or the ratings into fuzzy weights and fuzzy decision
11.  # matrix, respectively
12.  def cal(a, b, k):
13.      """ a is the dictionary with the linguistic variables
14.      for the criteria weights (or the linguistic
15.      variables for the ratings), b is the matrix with
16.      the criteria weights (or the ratings), and k is
17.      the number of the decision makers. The output is
18.      the fuzzy decision matrix or the fuzzy weights
19.      of the criteria
20.      """
21.      f = []
22.      for i in range(len(b)):
23.          c = []
24.          for z in range(3):
25.              x = 0
26.              for j in range (k):
27.                  x = x + a[b[i][j]][z]
28.              c.append(round(x / k, 3))
29.          f.append(c)
30.      return asarray(f)
31.
```

```
32.  # Calculate the fuzzy normalized decision matrix
33.  def fndm(a, n, m):
34.      """ a is the fuzzy decision matrix, n is the number of
35.      criteria, and m is the number of the alternatives.
36.      The output is the fuzzy normalized decision matrix
37.      """
38.      x = amax(a[:, 2:3])
39.      f = zeros((n * m, 3))
40.      for i in range(n * m):
41.          for j in range(3):
42.              f[i][j] = round(a[i][j] / x, 3)
43.      return f
44.
45.  # Calculate the fuzzy weighted normalized decision matrix
46.  def weighted_fndm(a, b, n, m):
47.      """ a is the fuzzy normalized decision matrix, b is the
48.      criteria weights, n is the number of criteria, and m
49.      is the number of the alternatives. The output is
50.      the fuzzy weighted normalized decision matrix
51.      """
52.      f = zeros((n * m, 3))
53.      z = 0
54.      for i in range(n * m):
55.          if i % len(b) == 0:
56.              z = 0
57.          else:
58.              z = z + 1
59.          for j in range(3):
60.              f[i][j] = round(a[i][j] * b[z][j], 3)
61.      return f
62.
63.  # Calculate the distance between two fuzzy triangular
64.  # numbers
65.  def distance(a, b):
66.      """ a and b are fuzzy triangular numbers. The output is
67.      their distance
68.      """
69.      return sqrt(1/3 * ((a[0] - b[0])**2 + (a[1] - b[1])**2
70.          + (a[2] - b[2])**2))
71.
72.  # Determine the fuzzy positive ideal solution (FPIS)
73.  def func_dist_fpis(a, n, m):
74.      """ a is the fuzzy weighted normalized decision matrix,
75.      n is the number of criteria, and m is the number of
76.      the alternatives. The output is the ideal
77.      solution of each criterion
78.      """
79.      fpis = ones((3, 1))
80.      dist_pis = zeros(m)
81.      p = 0
82.      for i in range(m):
83.          for j in range(n):
84.              dist_pis[i] = dist_pis[i] + distance(a[p + j],
85.                  fpis)
```

```
86.             p = p + n
87.         return dist_pis
88.
89.     # Determine the fuzzy negative ideal solution (FNIS)
90.     def func_dist_fnis(a, n, m):
91.         """ a is the fuzzy weighted normalized decision matrix,
92.         n is the number of criteria, and m is the number of
93.         the alternatives. The output is the anti-ideal
94.         solution of each criterion
95.         """
96.         fnis = zeros((3, 1))
97.         dist_nis = zeros(m)
98.         p = 0
99.         for i in range(m):
100.            for j in range(n):
101.                dist_nis[i] = dist_nis[i] + distance(a[p + j],
102.                    fnis)
103.            p = p + n
104.        return dist_nis
105.
106.    # Fuzzy TOPSIS method: it calls the other functions
107.    def f_topsis(a, b, c, d, n, m, k, pl):
108.        """ a is the dictionary with the linguistic variables
109.        for the criteria weights, b is the matrix with the
110.        importance weights of the criteria, c is a
111.        dictionary with the linguistic variables for the
112.        ratings, d is the matrix with all the ratings, n
113.        is the number of criteria, m is the number of the
114.        alternatives, and k is the number of the decision
115.        makers
116.        """
117.
118.        # Steps 3 and 4
119.        fuzzy_weights = cal(a, b, k)
120.        fuzzy_decision_matrix = cal(c, d, k)
121.        fuzzy_norm_decision_matrix = fndm(fuzzy_decision_matrix,
122.            n, m)
123.
124.        # Step 5
125.        weighted_fuzzy_norm_decision_matrix = \
126.            weighted_fndm(fuzzy_norm_decision_matrix,
127.            fuzzy_weights, n, m)
128.
129.        # Steps 6 and 7
130.        a_plus = func_dist_fpis(
131.           weighted_fuzzy_norm_decision_matrix, n, m)
132.        a_minus = func_dist_fnis(
133.           weighted_fuzzy_norm_decision_matrix, n, m)
134.
135.        # Step 8
136.        CC = [] # closeness coefficient
137.        for i in range(m):
138.            CC.append(round(a_minus[i] / (a_plus[i] +
139.                a_minus[i]), 3))
```

```
140.
141.     if pl == 'y':
142.         q = [i + 1 for i in range(m)]
143.         plt.plot(q, a_plus, 'p--', color = 'red',
144.             markeredgewidth = 2, markersize = 8)
145.         plt.plot(q, a_minus, '*--',  color = 'blue',
146.             markeredgewidth = 2, markersize = 8)
147.         plt.plot(q, CC, 'o--', color = 'green',
148.             markeredgewidth = 2, markersize = 8)
149.         plt.title('Fuzzy TOPSIS results')
150.         plt.legend(['Distance from the ideal',
151.             'Distance from the anti-ideal',
152.             'Closeness coeficient'])
153.         plt.xticks(range(m + 2))
154.         plt.axis([0, m + 1, 0, 3])
155.         plt.xlabel('Alternatives')
156.         plt.legend()
157.         plt.grid(True)
158.         plt.show()
159.     return CC
160.
161. m = 6 # the number of the alternatives
162. n = 4 # the number of the criteria
163. k = 3 # the number of the decision makers
164.
165. # Steps 1 and 2
166. # Define a dictionary with the linguistic variables for the
167. # criteria weights
168. cw = {'VL':[0, 0, 0.1], 'L':[0, 0.1, 0.3],
169.       'ML':[0.1, 0.3, 0.5], 'M':[0.3, 0.5, 0.7],
170.       'MH':[0.5, 0.7, 0.9], 'H':[0.7, 0.9, 1],
171.       'VH':[0.9, 1, 1]}
172.
173. # Define a dictionary with the linguistic variables for the
174. # ratings
175. r = {'VP':[0, 0, 1], 'P':[0, 1, 3], 'MP':[1, 3, 5],
176.      'F':[3, 5, 7], 'MG':[5, 7, 9], 'G':[7, 9, 10],
177.      'VG':[9, 10, 10]}
178.
179. # The matrix with the criteria weights
180. cdw = [['H', 'VH', 'VH'], ['M', 'H', 'VH'],
181.        ['M', 'MH', 'ML'], ['H', 'VH', 'MH']]
182.
183. # The ratings of the six candidate sites by the decision
184. # makers under all criteria
185. c1 = [['VG', 'G', 'MG'], ['F', 'MG', 'MG'],
186.       ['P', 'P', 'MP'], ['G', 'VG', 'G']]
187. c2 = [['MP', 'F', 'F'], ['F', 'VG', 'G'],
188.       ['MG', 'VG', 'G'], ['MG', 'F', 'MP']]
189. c3 = [['MG', 'MP', 'F'], ['MG', 'MG', 'VG'],
190.       ['MP', 'F', 'F'], ['MP', 'P', 'P']]
191. c4 = [['MG', 'VG', 'VG'], ['G', 'G', 'VG'],
192.       ['MG', 'VG', 'G'], ['VP', 'F', 'P']]
193. c5 = [['VP', 'P', 'G'], ['P', 'VP', 'MP'],
```

```
194.        ['G', 'G', 'VG'], ['G', 'MG', 'MG']]
195. c6 = [['F', 'G', 'G'], ['F', 'MP', 'MG'],
196.        ['VG', 'MG', 'F'], ['P', 'MP', 'F']]
197.
198. all_ratings = vstack((c1, c2, c3, c4, c5, c6))
199.
200. # final results
201. start = timeit.default_timer()
202. f_topsis(cw, cdw, r, all_ratings, n, m, k, 'n')
203. stop = timeit.default_timer()
204. print(stop - start)
205. print("Closeness coefficient = ",
206.     f_topsis(cw, cdw, r, all_ratings, n, m, k, 'y'))
```

References

1. Anisseh, M., Piri, F., Shahraki, M. R., & Agamohamadi, F. (2012). Fuzzy extension of TOPSIS model for group decision making under multiple criteria. *Artificial Intelligence Review, 38*(4), 325–338.
2. Bede, B. (2013). *Mathematics of fuzzy sets and fuzzy logic*. Berlin: Springer.
3. Behzadian, M., Otaghsara, S. K., Yazdani, M., & Ignatius, J. (2012). A state-of the-art survey of TOPSIS applications. *Expert Systems with Applications, 39*(17), 13051–13069.
4. Cha, Y., & Jung, M. (2003). Satisfaction assessment of multi-objective schedules using neural fuzzy methodology. *International Journal of Production Research, 41*(8), 1831–1849.
5. Chen, C. T. (2000). Extensions of the TOPSIS for group decision-making under fuzzy environment. *Fuzzy Sets and Systems, 114*, 1–9.
6. Chen, S. H. (1985). Ranking fuzzy numbers with maximizing set and minimizing set. *Fuzzy Sets and Systems, 17*(2), 113–129.
7. Chen, S. J., & Hwang, C. L. (1992). *Fuzzy multiple attribute decision making methods*. Berlin: Springer.
8. Chu, T. C. (2002). Facility location selection using fuzzy TOPSIS under group decisions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10*(6), 687–701.
9. Chu, T. C., & Lin, Y. C. (2003). A fuzzy TOPSIS method for robot selection. *The International Journal of Advanced Manufacturing Technology, 21*(4), 284–290.
10. Diamond, P., & Kloeden, P. (2000). Metric topology of fuzzy numbers and fuzzy analysis. In D. Dubois & H. Prade (Eds.), *Fundamentals of fuzzy sets* (pp. 583–641). New York: Springer US.
11. Dymova, L., Sevastjanov, P., & Tikhonenko, A. (2013). An approach to generalization of fuzzy TOPSIS method. *Information Sciences, 238*, 149–162.
12. Garcia-Cascales, M. S., & Lamata, M. T. (2012). On rank reversal and TOPSIS method. *Mathematical and Computer Modelling, 56*(5), 123–132.
13. Giove, S. (2002). Interval TOPSIS for multicriteria decision making. In *Italian Workshop on Neural Nets* (pp. 56–63). Berlin: Springer.
14. Hwang, C. L., & Yoon, K. (1981). *Multiple attribute decision making: Methods and applications*. Berlin: Springer
15. Izadikhah, M. (2009). Using the Hamming distance to extend TOPSIS in a fuzzy environment. *Journal of Computational and Applied Mathematics, 231*(1), 200–207.
16. Jahan, A., & Edwards, K. L. (2015). A state-of-the-art survey on the influence of normalization techniques in ranking: Improving the materials selection process in engineering design. *Materials & Design, 65*, 335–342.

17. Jahanshahloo, G. R., Lotfi, F. H., & Davoodi, A. R. (2009). Extension of TOPSIS for decision-making problems with interval data: interval efficiency. *Mathematical and Computer Modelling, 49*(5), 1137–1142.
18. Jahanshahloo, G. R., Lotfi, F. H., & Izadikhah, M. (2006). An algorithmic method to extend TOPSIS for decision-making problems with interval data. *Applied Mathematics and Computation, 175*(2), 1375–1384.
19. Jahanshahloo, G. R., Lotfi, F. H., & Izadikhah, M. (2006). Extension of the TOPSIS method for decision-making problems with fuzzy data. *Applied Mathematics and Computation, 181*(2), 1544–1551.
20. Kahraman, C., Kaya, I., Çevik, S., Ates, N. Y., & Gülbay, M. (2008). Fuzzy multi-criteria evaluation of industrial robotic systems using TOPSIS. In C. Kahramn (Ed.), *Fuzzy multi-criteria decision making* (pp. 159–186). New York: Springer US.
21. Kaufmann, A., & Gupta, M. M. (1988). *Fuzzy mathematical models in engineering and management science*. New York: Elsevier Science Inc.
22. Lee, E. S., & Li, R. J. (1988). Comparison of fuzzy numbers based on the probability measure of fuzzy events. *Computers & Mathematics with Applications, 15*(10), 887–896.
23. Lee, K. H. (2006). *First course on fuzzy theory and applications* (Vol. 27). Berlin: Springer Science & Business Media.
24. Li, X., & Chen, X. (2014). Extension of the TOPSIS method based on prospect theory and trapezoidal intuitionistic fuzzy numbers for group decision making. *Journal of Systems Science and Systems Engineering, 23*(2), 231–247.
25. Liang, G. S. (1999). Fuzzy MCDM based on ideal and anti-ideal concepts. *European Journal of Operational Research, 112*(3), 682–691.
26. Liou, T. S., & Wang, M. J. J. (1992). Ranking fuzzy numbers with integral value. *Fuzzy Sets and Systems, 50*(3), 247–255.
27. Mahdavi, I., Mahdavi-Amiri, N., Heidarzade, A., & Nourifar, R. (2008). Designing a model of fuzzy TOPSIS in multiple criteria decision making. *Applied Mathematics and Computation, 206*(2), 607–617.
28. Owen, S. H., & Daskin, M. S. (1998). Strategic facility location: A review. *European Journal of Operational Research, 111*(3), 423–447.
29. Pavlicic, D. (2001). Normalization affects the results of MADM methods. *Yugoslav Journal of Operations Research, 11*(2), 251–265.
30. ReVelle, C. S., & Eiselt, H. A. (2005). Location analysis: a synthesis and survey. *European Journal of Operational Research, 165*(1), 1–19.
31. Shih, H. S., Shyur, H. J., & Lee, E. S. (2007). An extension of TOPSIS for group decision making. *Mathematical and Computer Modelling, 45*(7), 801–813.
32. Tsaur, S. H., Chang, T. Y., & Yen, C. H. (2002). The evaluation of airline service quality by fuzzy MCDM. *Tourism Management, 23*(2), 107–115.
33. Vafaei, N., Ribeiro, R. A., & Camarinha-Matos, L. M. (2015). Importance of data normalization in decision making: Case study with TOPSIS method. In B. Delibasic, F. Dargam, P. Zarate, J.E. Hernendez, S. Liu, R. Ribeiro, I. Linden & J. Papathanasiou (Eds.), *ICDSST 2015 Proceedings - the 1st International Conference on Decision Support Systems Technologies, an EWG-DSS Conference*, Belgrade, Serbia.
34. Wang, T. C., & Lee, H. D. (2009). Developing a fuzzy TOPSIS approach based on subjective weights and objective weights. *Expert Systems with Applications, 36*(5), 8980–8985.
35. Wang, Y. J., & Lee, H. S. (2007). Generalizing TOPSIS for fuzzy multiple-criteria group decision-making. *Computers & Mathematics with Applications, 53*(11), 1762–1772.
36. Wang, Y. M., & Elhag, T. M. (2006). Fuzzy TOPSIS method based on alpha level sets with an application to bridge risk assessment. *Expert Systems with Applications, 31*(2), 309–319.
37. Wang, Y. M., & Luo, Y. (2009). On rank reversal in decision analysis. *Mathematical and Computer Modelling, 49*(5), 1221–1229.
38. Yang, T., & Hung, C. C. (2007). Multiple-attribute decision making methods for plant layout design problem. *Robotics and Computer-Integrated Manufacturing, 23*(1), 126–137.

39. Yue, Z. (2012). Extension of TOPSIS to determine weight of decision maker for group decision making problems with uncertain information. *Expert Systems with Applications, 39*(7), 6343–6350.
40. Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338–353.
41. Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning—I. *Information Sciences, 8*(3), 199–249.
42. Zhang, G., & Lu, J. (2003). An integrated group decision-making method dealing with fuzzy preferences for alternatives and individual judgements for selection criteria. *Group Decision and Negotiation, 12*(6), 501–515.
43. Zhao, R., & Govind, R. (1991). Algebraic characteristics of extended fuzzy numbers. *Information Sciences, 54*(1), 103–130.